Davide Ancona,
Marco Maratea,
Viviana Mascardi (eds.)

Convegno Italiano di Logica Computazionale

Trentesima Edizione, CILC 2015
Genova, 1–3 Luglio 2015
Atti del Convegno

CILC 2015 Home Page:
http://cilc2015.dibris.unige.it/

Indirizzo degli editori

DIBRIS – Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università degli Studi di Genova

Davide Ancona, Via Dodecaneso, 35 – 16146 Genova – Italy,
`davide.ancona@unige.it`

Marco Maratea, Viale Causa, 13 – 16145 Genova – Italy,
`marco.maratea@unige.it`

Viviana Mascardi, Via Dodecaneso, 35 – 16146 Genova – Italy,
`viviana.mascardi@unige.it`

Prefazione

In occasione del suo trentennale, il convegno annuale del GULP (Gruppo ricercatori e Utenti Logic Programming) è ritornato a Genova nei giorni 1–3 luglio 2015. Il convegno è stato preceduto da una Scuola di Logica Computazionale nei giorni 29 e 30 giugno e nella mattina del 1 luglio.

Sin dal primo evento della serie, tenutosi proprio a Genova nel 1986, il convegno annuale del GULP ha rappresentato la principale occasione di incontro e scambio di idee ed esperienze tra utenti, ricercatori e sviluppatori che operano nel campo della logica computazionale. Nel corso degli anni il convegno ha allargato i propri orizzonti dal campo specifico della programmazione logica tradizionale a quelli più generali della programmazione dichiarativa, delle applicazioni in diversi settori limitrofi, quali l'Intelligenza Artificiale o i Database Deduttivi. Anche con l'edizione 2015 il GULP ha proseguito questa politica identificando, con il più generale termine di Logica Computazionale, l'intero variegato mondo della ricerca, di base e applicata, che direttamente o indirettamente utilizza o si confronta con le idee e le tecniche proprie della logica come strumento per la rappresentazione e il calcolo.

La trentesima edizione si è aperta con la presentazione delle tesi di dottorato vincitrici del Premio GULP 2014, *Integration of logic and probability in terminological and inductive reasoning* di Elena Bellodi e *Software verification and synthesis using constraints and program transformation* di Emanuele De Angelis, ed è stata arricchita dalla relazione invitata di Georg Gottlob su *A Framework for Data, Knowledge, and Reasoning: Datalog*$^{\pm}$ e dal panel *GULP 30 e lode!* che ha visto la partecipazione di molti dei pionieri della Logica Computazionale italiana tra i quali Giovanni Adorni, Stefania Costantini, Maurizio Martelli, Ugo Montanari, Eugenio Omodeo, Gianfranco Rossi.

Questo volume contiene i venti contributi originali, dei trenta selezionati dal Comitato di Programma per essere presentati al convegno. I dieci lavori presentati al convegno ma non inclusi negli atti sono

- *Modeling and verifying relational multiagent systems with data types* di Diego Calvanese, Giorgio Delzanno e Marco Montali,
- *Multi-agent-contexts systems for reasoning and acting in heterogeneous environments* di Stefania Costantini,
- *Integration of DALI agents and ASP modules: a case-study* di Stefania Costantini, Giovanni De Gasperis e Giulio Nazzicone,
- *Completing workflow traces using action languages* di Chiara Di Francescomarino, Chiara Ghidini, Sergio Tessaris e Itzel Vazquez Sandoval,
- *Semantics-based generation of verification conditions by program specialization* di Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi e Maurizio Proietti,

- *Proving Horn clause specifications of partial correctness of imperative programs* di Emanuele De Angelis, Fabio Fioravanti, Alberto Pettorossi e Maurizio Proietti,
- *Web ontology representation and reasoning via fragments of set theory* di Domenico Cantone, Cristiano Longo, Marianna Nicolosi-Asmundo e Daniele Francesco Santamaria,
- *Towards a rational closure for expressive description logics: the case of SHIQ* di Laura Giordano, Valentina Gliozzi e Nicola Olivetti,
- *Why CP portfolio solvers are (under)utilized? Issues and challenges* di Roberto Amadini, Maurizio Gabbrielli e Jacopo Mauro, e
- *A contextual framework for reasoning on events* di Loris Bozzato, Stefano Borgo, Alessio Palmero Aprosio, Marco Rospocher e Luciano Serafini.

I contributi degli articoli coprono aree di ricerca estremamente attuali e variegate che vanno dai fondamenti e risultati teorici alle esperienze pratiche, dai casi di studio alle applicazioni, dagli agenti e sistemi multiagente alle logiche descrittive e ontologie, passando attraverso i linguaggi, i framework e gli strumenti.

Ringraziamo sentitamente tutti coloro che hanno permesso la realizzazione di questa edizione: i componenti del Comitato di Programma, il Presidente del GULP Agostino Dovier, il segretario del GULP Marco Gavanelli, i docenti della scuola di Logica Computazionale (Alessandro Dal Palù, Agostino Dovier, Marco Gavanelli, Angelo Montanari, Enrico Pontelli, Francesco Ricca, Paolo Torroni), Georg Gottlob, i panelist, i chair delle sessioni tecniche, il Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi dell'Università degli Studi di Genova, gli sponsor, Daniela Briola, Andrea Corradi e Angelo Ferrando per il loro aiuto nell'organizzazione locale, ma soprattutto i partecipanti alla scuola e al convegno che con il loro entusiasmo hanno contribuito al successo di questa trentesima edizione del CILC.

30 Agosto 2015

Davide Ancona
Marco Maratea
Viviana Mascardi

Organizzazione scientifica e locale

Davide Ancona, Università degli Studi di Genova
Marco Maratea, Università degli Studi di Genova
Viviana Mascardi, Università degli Studi di Genova

Comitato di programma

Paolo Baldan, Università degli Studi di Padova
Matteo Baldoni, Università degli Studi di Torino
Elena Bellodi, Università degli Studi di Ferrara
Stefano Bistarelli, Università degli Studi di Perugia
Davide Bresolin, Università degli Studi di Bologna
Alberto Casagrande, Università degli Studi di Trieste
Iliano Cervesato, Carnegie Mellon University
Federico Chesani, Università degli Studi di Bologna
Simona Colucci, Politecnico di Bari
Agostino Cortesi, Università Ca' Foscari di Venezia
Stefania Costantini, Università degli Studi di L'Aquila
Emanuele De Angelis, Università degli Studi di Chieti-Pescara
Wolfgang Faber, University of Huddersfield
Moreno Falaschi, Università degli Studi di Siena
Stefano Ferilli, Università degli Studi di Bari
Fabio Fioravanti, Università degli Studi di Chieti-Pescara
Camillo Fiorentini, Università degli Studi di Milano
Andrea Formisano, Università degli Studi di Perugia
Chiara Ghidini, Fondazione Bruno Kessler
Laura Giordano, Università degli Studi del Piemonte Orientale
Valentina Gliozzi, Università degli Studi di Torino
Evelina Lamma, Università degli Studi di Ferrara
Francesca Alessandra Lisi, Università degli Studi di Bari Aldo Moro
Isabella Mastroeni, Università degli Studi di Verona
Alessandra Mileo, National University of Ireland, Galway
Marco Montali, Free University of Bozen-Bolzano
Alessandro Mosca, Free University of Bozen-Bolzano
Marianna Nicolosi-Asmundo, Università degli Studi di Catania
Nicola Olivetti, Aix-Marseille University
Fabio Patrizi, Università La Sapienza di Roma
Carla Piazza, Università degli Studi di Udine
Enrico Pontelli, New Mexico State University
Maurizio Proietti, IASI-CNR di Roma
Alessandro Provetti, Università degli Studi di Messina
Luca Pulina, Università degli Studi di Sassari
Elisa Quintarelli, Politecnico di Milano

Francesco Ricca, Università della Calabria
Gianfranco Rossi, Università degli Studi di Parma
Luigi Sauro, Università di Napoli Federico II
Umberto Straccia, ISTI-CNR di Pisa
Laura Titolo, University of Malaga

Revisori esterni

Mauro Ferrari
Andrea Pazienza
Iliana Petrova
Gian Luca Pozzato

Organi direttivi del GULP

Presidente
Agostino Dovier

Vicepresidente
Stefania Costantini

Segretario
Marco Gavanelli

Tesoriere
Roberto Fontana

Ex-presidenti
Roberto Barbuti
Giorgio Levi
Maurizio Gabbrielli
Maurizio Martelli
Gianfranco Rossi

Consiglio direttivo
Federico Chesani
Alessandro Dal Palù
Andrea Formisano
Fabio Fioravanti
Francesca Alessandra Lisi
Viviana Mascardi
Marco Montali
Alberto Pettorossi
Enrico Pontelli

Gian Luca Pozzato
Fabrizio Riguzzi
Francesca Rossi

Sponsor




Indice dei contenuti

Linguaggi e programmazione

Logiche descrittive e ontologie

Applicazioni, framework e strumenti

A framework for data, knowledge, and reasoning: Datalog$^{\pm}$

Georg Gottlob

Department of Computer Science, St John's College, Oxford, UK
georg.gottlob@cs.ox.ac.uk

Abstract. Datalog$^{\pm}$ is a family of logic programming languages for data manipulation, knowledge representation and reasoning. These languages extend Datalog with features such as existential quantifiers, equalities, and the falsum in rule heads and negation in rule bodies, and, at the same time, apply restrictions in order to to achieve decidability and tractability. This talk will start with a general overview of the Datalog$^{\pm}$ family and its main decidability paradigms and an explanation of how tractable classes can be achieved. Subsequently, some more specialized issues will be dealt with such as nonmonotonic negation and disjunction. We will also report about a special version of Datalog$^{\pm}$ suitable for ontological reasoning, reasoning with reverse-engineered UML class diagrams, and about the TriQ language that expresses SPARQL with entailment regimes.

A Diophantine representation of Wolstenholme's pseudoprimality*

Luca Vallata[1] and Eugenio G. Omodeo[2]

[1] Graduated from the University of Trieste,
email: luca.vallata@gmail.com
[2] Dipartimento di Matematica e Geoscienze / DMI, Università di Trieste,
Via Valerio 12/1, I-34127 – Trieste, Italy
email: eomodeo@units.it

Abstract. As a by-product of the negative solution of Hilbert's 10^{th} problem, various prime-generating polynomials were found. The best known *upper* bound for the number of variables in such a polynomial, to wit 10, was found by Yuri V. Matiyasevich in 1977.
We show that this bound could be lowered to 8 if the converse of Wolstenholme's theorem (1862) holds, as conjectured by James P. Jones. This potential improvement is achieved through a Diophantine representation of the set of all integers $p \geqslant 5$ that satisfy the congruence $\binom{2p}{p} \equiv 2 \mod p^3$. Our specification, in its turn, relies upon a terse polynomial representation of exponentiation due to Matiyasevich and Julia Robinson (1975), as further manipulated by Maxim Vsemirnov (1997).
We briefly address the issue of also determining a *lower* bound for the number of variables in a prime-representing polynomial, and discuss the autonomous significance of our result about Wostenholme's pseudoprimality, independently of Jones's conjecture.

Keywords. Diophantine representations, Hilbert's 10^{th} problem, DPRM theorem, Wolstenholme's theorem, Siegel's theorem on integral points.

Introduction

At the beginning of the 1960's, one decade after Martin Davis had set forth the 'daring hypothesis $\cdots$ that *every semidecidable set is Diophantine*' [Mat93, p. 99], it became clear that finding a proof of that conjecture would have entailed the possibility to construct a polynomial with integer coefficients whose positive values, as the variables run through all nonnegative integers, form the set of prime numbers.[3] The existence of such a prime-generating polynomial seemed, at the time, rather unlikely; in fact, Davis's conjecture was received with understandable skepticism.

* Work partially supported by the project FRA-UniTS (2014) *"Learning specifications and robustness in signal analysis (with a case study related to health care)."*

[3] Cf. [DMR76, Sec. 1]: "This corollary was deduced by Putnam in 1960 from the then conjectured Main Theorem and it was considered by some to be an argument against its plausibility."

With [Mat70], Yuri V. Matiyasevich positively settled Davis's conjecture and so provided a negative answer to Hilbert's 10^{th} problem [Hil00, p. 276]. Soon afterward, the same scholar obtained two polynomials representing primes and only primes, one in 24 and one in 21 variables [Mat71]; in [MR75], Matiyasevich and Julia Robinson brought the number of variables down to 14; then other researchers succeeded in bringing it further down, to 12 (cf. [JSWW76]). The record number, 10 as of today, was achieved by Matiyasevich in 1977: in fact, [Mat81] produces a prime-generating polynomial in 10 variables, of degree 15905 (reducible to 13201 (13983?) or to 11281 [Mat81, p. 44], or even to 10001 [Vse97, p. 3204]).

Although methods have significantly evolved over time, the rigmarole for getting prime-representing polynomials usually results from the combination of ideas already present in [Rob52] (see Fig. 1) with a Diophantine polynomial specification of exponentiation, such as the masterpiece proposed in [MR75] (see Fig. 2), which Maxim A. Vsemirnov refined somewhat in [Vse97].

$$a = \binom{r}{j} \;\leftrightarrow\; a = \left\lfloor \frac{(u+1)^r}{u^j} \right\rfloor \% \, u \;\;\&\;\; u = 2^r + 1$$

$$j! = \left\lfloor \frac{r^j}{\binom{r}{j}} \right\rfloor \quad \text{for any } r > (2\,j)^{j+1}$$

$$\neg\, \exists\, x\,,\; y\, \big(p = (x+2)\,(y+2) \vee p = 0 \vee p = 1\big)$$
$$\leftrightarrow\; \exists\, q\,,\; u\,,\; v\, \big(p = q + 2 \;\&\; p\,u - (q+1)!\,v = 1\big)$$

Fig. 1. Binomial coefficient, factorial, and "p is a prime" are existentially definable by means of exponential Diophantine equations, cf. [Rob52, pp. 446–447]. Throughout, '%' designates the integer remainder operation.

Ameliorations along this pipeline are possible: e.g., Wilson's theorem enables one to state that p is a prime number through the formula $\exists\, q\,,\; u\, \big(p = q + 2 \;\&\; p\,u - (q+1)! = 1\big)$; and an improved exponential Diophantine representation of the binomial coefficient can be obtained through the theorem

$$\binom{r}{j} = \left\lfloor \frac{(u+1)^r}{u^j} \right\rfloor \% \, u \quad \text{for } r > 0\,,\; j > 0\,, \text{ and } u > r^j\,,$$

as remarked in [MR75, pp. 544–545]. However, a more decisive enhancement in the formulation of a prime-generating polynomial would ensue if one could remove factorial from the pipeline and could avoid exploiting the binomial coefficient in its full strength.

Joseph Wolstenholme proved the congruence $\binom{2\,p-1}{p-1} \equiv 1 \mod p^3$ for all prime numbers $p > 3$ in 1862 [Wol62]; and it was conjectured by James P. Jones (cf. [Rib04, p. 23] and [McI95, p. 381]) that, conversely, every integer $p > 3$ satisfying the said congruence is prime. If true, this conjecture would ease our

$$Q = \square \;\leftrightarrow_{\text{Def}}\; Q = h^2 \quad \text{for some } h \in \mathbb{N}\,,$$

$$X \mid Y \;\leftrightarrow_{\text{Def}}\; Y = \pm h\, X \text{ for some } h \in \mathbb{N}\,.$$

A1	$D\,F\,I = \square\,,\;\; F \mid H - C\,,\;\; B \leqslant C$	**E1**	$(M^2-1)\,L^2 + 1 = \square$
A2	$D \leftrightharpoons (A^2-1)\,C^2 + 1$	**E2**	$L^2 - 4\,(C - L\,y)^2\,x\,y\,n > 0$
A3	$E \leftrightharpoons 2\,(i+1)\,C^2\,D$	**E3**	$M \leftrightharpoons 4\,n\,(y+1) + x + 2$
A4	$F \leftrightharpoons (A^2-1)\,E^2 + 1$	**E4**	$L \leftrightharpoons n + 1 + \ell\,(M-1)$
A5	$G \leftrightharpoons (F-A)\,F + A$	**E5**	$A \leftrightharpoons M\,x$
A6	$H \leftrightharpoons B + 2\,j\,C$	**E6**	$B \leftrightharpoons n + 1$
A7	$I \leftrightharpoons (G^2-1)\,H^2 + 1$	**E7**	$C \leftrightharpoons k + B$

Fig. 2. Polynomial specification of the triadic relation $x^n = y$. Besides the parameters x, y, n, this involves four existential variables (also ranging over $\mathbb{N}$): i, j, k, ℓ; a fifth unknown is implicit in the constraint **A1** stating that the product DFI must be a perfect square with F dividing $H - C$. The notation '$V \leftrightharpoons P$' defines V to be an alias for the (integer-valued) polynomial P; hence all uppercase letters can be eliminated, e.g. in the order: M, B; A, C, L; H, D; E; F; G; I. By themselves, **A1**–**A7** form a polynomial specification of the relation $\psi_A(B) = C$ defined by the recurrence $\psi_A(0) = 0, \psi_A(1) = 1$, and $\psi_A(h+2) = 2A\psi_A(h+1) - \psi_A(h)$, if one takes A, B, C as parameters subject to the preconditions $A > 1$, $B > 0$, $C > 0$.

present task, enabling us to express primality without factorial and in terms of the *central* binomial coefficient $\binom{2p}{p}$.

After recalling, in Sec. 1 the basic definitions and techniques we need, in Sec. 2 we produce a Diophantine polynomial generator in 8 variables for the numbers meeting the just mentioned 'Wolstenholme's pseudoprimality' criterion. In Sec. 3, we give clues about the proof that the proposed polynomial operates properly. In the conclusions, we briefly discuss the autonomous significance of our specification independently of Jones's conjecture, and address the issue of determining a lower bound for the number of variables in a polynomial representation of primality.

1 Main definitions and presupposed notions

Let us recall here the notion of *Diophantine representation* of a relation $\mathcal{R}$, which historically played an essential role in the study of Hilbert's 10^{th} problem:

Definition 1. *A relation $\mathcal{R}$ among n natural numbers is said to be* Diophantine *if one can precisely characterize which are the n-tuples $\langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \rangle$ constituting $\mathcal{R}$ through a bi-implication of the form*

$$\mathcal{R}(a_1, \ldots, a_n) \leftrightarrow \exists\, x_1 \cdots \exists\, x_m \left(D(\overbrace{\underbrace{a_1, \ldots, a_n}_{\text{parameters}}, \underbrace{x_1, \ldots, x_m}_{\text{unknowns}}}^{\text{variables}}) = 0 \right)$$

which musto be true under the replacement $a_1 \mapsto \boldsymbol{a}_1, \ldots, a_n \mapsto \boldsymbol{a}_n$, where D is a polynomial with coefficients in $\mathbb{Z}$ whose variables are seen as ranging over $\mathbb{N}$.

In the common case when $n = 1$ one calls such an $\mathcal{R}$ a *Diophantine set*, and one readily gets from the defining D the polynomial $(x_0+1)\big(1-D^2(x_0,\ldots,x_m)\big)-1$, whose non-negative values (under replacement of the variables $x_0,\ldots,x_m$ by natural numbers $\boldsymbol{x}_0,\ldots,\boldsymbol{x}_m$) are precisely the elements of $\mathcal{R}$.

For example, classical results on the so-called Pell equation tell us that the equation $x^2 - d\,(y+1)^2 - 1 = 0$ in the parameter d and in the unknowns x, y makes a Diophantine representation of the set

$$\mathcal{R} = \{\,0\,\} \;\cup\; \{\,d \in \mathbb{N} \mid d \text{ is } \textit{not} \text{ a perfect square}\,\}\,;$$

therefore the non-negative values of the polynomial

$$(z+1)\,\big(\,1 - (x^2 - z\,(y+1)^2 - 1)^2\,\big) - 1\,,$$

as x, y, z range over $\mathbb{N}$, will form this $\mathcal{R}$.

The Pell equation of the special form $x^2 = (a^2-1)\,y^2+1$ enters extensively in the ongoing; thus we find it convenient to denote its right-hand side as $\mathsf{Pell}(a, y)$. We adopt $\mathsf{Pell}(S, T)$ as an analogous syntactic abbreviation also in the case when S and T are Diophantine polynomials, as shown in Fig. 6 (top).

As is well-known (see, e.g., [Dav73]), the solutions to the said equation $x^2 = \mathsf{Pell}(\boldsymbol{a}, y)$ when $\boldsymbol{a} \geqslant 2$ form a doubly recurrent infinite sequence

$$\langle 1\,,\,0\rangle,\ \langle \boldsymbol{a}\,,\,1\rangle,\ \langle 2\,\boldsymbol{a}^2 - 1\,,\,2\,\boldsymbol{a}\rangle,\ \langle 4\,\boldsymbol{a}^3 - 3\,\boldsymbol{a}\,,\,4\,\boldsymbol{a}^2 - 1\rangle,\ \ldots$$

of pairs whose first and second components constitute the respective increasing progressions $\chi_{\boldsymbol{a}}(0), \chi_{\boldsymbol{a}}(1),\ \chi_{\boldsymbol{a}}(2), \ldots$ and $\psi_{\boldsymbol{a}}(0), \psi_{\boldsymbol{a}}(1), \ldots$ shown in Fig. 3 (the latter was formerly introduced in the caption of Fig. 2). Figures 4, 5 recapitulate important properties enjoyed by these sequences.

$\chi_{\boldsymbol{a}}(0) = 1$	$\chi_{\boldsymbol{a}}(1) = \boldsymbol{a}$	$\chi_{\boldsymbol{a}}(h+2) = 2\,\boldsymbol{a}\,\chi_{\boldsymbol{a}}(h+1) - \chi_{\boldsymbol{a}}(h)$
$\psi_{\boldsymbol{a}}(0) = 0$	$\psi_{\boldsymbol{a}}(1) = 1$	$\psi_{\boldsymbol{a}}(h+2) = 2\,\boldsymbol{a}\,\psi_{\boldsymbol{a}}(h+1) - \psi_{\boldsymbol{a}}(h)$

Fig. 3. Recurrent specification of the solutions $\boldsymbol{x} = \chi_{\boldsymbol{a}}(b)$, $\boldsymbol{y} = \psi_{\boldsymbol{a}}(b)$ of Pell's equation $x^2 - (\boldsymbol{a}^2 - 1)\,y^2 = 1$. (These make sense even for $\boldsymbol{a} = 1$.)

2 How to represent Wolstenholme's pseudoprimality via a Diophantine polynomial

To be better aligned with [Vse97], let us now agree that the variables appearing in our Diophantine constraints must range over *positive* (instead of non-negative) integers. A refined polynomial specification of the components which occupy odd positions b in the progression $\psi_{\boldsymbol{a}}(b) = c$ discussed above is shown in Fig. 6

$$
\begin{aligned}
n < \boldsymbol{a}^n \leqslant \chi_{\boldsymbol{a}}(n) \leqslant \frac{\chi_{\boldsymbol{a}}(n+1)}{\boldsymbol{a}} &< \begin{cases} \chi_{\boldsymbol{a}}(n+1)\,, \\ (2\,\boldsymbol{a})^n + 1\,; \end{cases} \\
n \ \leqslant \psi_{\boldsymbol{a}}(n) < \frac{\psi_{\boldsymbol{a}}(n+1)}{\boldsymbol{a}} &< \quad \psi_{\boldsymbol{a}}(n+1)\,; \\
\psi_{\boldsymbol{a}}(n) &< \begin{cases} \frac{1}{2}\,\chi_{\boldsymbol{a}}(n+1)\,, \\ \frac{1}{2}\,\chi_{\boldsymbol{a}}(n) \quad \text{if } \boldsymbol{a} > 2\,; \end{cases} \\
(2\,\boldsymbol{a}-1)^n \leqslant \psi_{\boldsymbol{a}}(n+1) &\leqslant (2\,\boldsymbol{a})^n\,.
\end{aligned}
$$

Fig. 4. Noteworthy inequalities holding for the progressions $\chi_{\boldsymbol{a}}, \psi_{\boldsymbol{a}}$ ($\boldsymbol{a} \geqslant 2$).

0.	$\chi_{\boldsymbol{a}}(n) - \psi_{\boldsymbol{a}}(n)\,(\boldsymbol{a}-\ell) \ \equiv \ \ell^n \mod (\,2\,\boldsymbol{a}\,\ell - \ell^2 - 1\,)$;
1.	$\psi_{\boldsymbol{a}}(n) \equiv n \mod (\boldsymbol{a}-1)$ and $\psi_{\boldsymbol{a}}(n) \equiv n \mod 2$;
2.	$p \equiv q \mod r$ implies $\begin{cases} \chi_p(n) \equiv \chi_q(n) \mod r\,, \\ \psi_p(n) \equiv \psi_q(n) \mod r\,; \end{cases}$ $r \mid (p-1)$ implies $\psi_p(n) \equiv n \mod r$;
3.	$\psi_{\boldsymbol{a}}(n) \mid \psi_{\boldsymbol{a}}(n\,k)$;
4.	$\psi_{\boldsymbol{a}}(n) \mid \psi_{\boldsymbol{a}}(\ell)$ iff $n \mid \ell$;
5.	$\psi_{\boldsymbol{a}}(m\,r) \equiv r\,\chi_{\boldsymbol{a}}^{r-1}(m)\,\psi_{\boldsymbol{a}}(m) \mod \psi_{\boldsymbol{a}}^3(m)$;
6.	$\psi_{\boldsymbol{a}}(n) \mid \ell$ if $\psi_{\boldsymbol{a}}^2(n) \mid \psi_{\boldsymbol{a}}(\ell)$;
7.	$\psi_{\boldsymbol{a}}^2(n) \mid \psi_{\boldsymbol{a}}(n\,\psi_{\boldsymbol{a}}(n))$;
8.	$\psi_{\boldsymbol{a}}(i) \equiv \psi_{\boldsymbol{a}}(j) \mod \chi_{\boldsymbol{a}}(m)$ implies $(\,i \equiv j \vee i \equiv -j\,) \mod (\,2\,m\,)$.

Fig. 5. Noteworthy congruences holding for the progressions $\chi_{\boldsymbol{a}}, \psi_{\boldsymbol{a}}$. Here it is assumed that $n \geqslant 0$, $k \geqslant 0$, $\ell \geqslant 0$ and that $p > 0$, $q > 0$, $r > 0$, $m > 0$.

(right) and in Fig. 7 (left); in Fig. 7 (right) we extend it into an alike specification, to be discussed next, of Wolstenholme's pseudoprimality. In addition to the 6 unknowns z, w, s, h, i, j which appear explicitly in this system of Diophantine constraints, additional unknowns enter into play due to the presence of the constructs '$\square$', '$>$', '$|$', and of a congruence. Eliminating such abbreviations seems, at first glance, to call for five extra variables; a single, 7-th unknown suffices, though, thanks to the following proposition:

Theorem 1 (Relation-combining theorem, [MR75, pp. 525–527]). *To each q in $\mathbb{N}$ there corresponds a polynomial M_q with coefficients in $\mathbb{Z}$ such that, for all integers $X_1, \ldots, X_q$, J, R, V with $J \neq 0$, the conditions*

$$X_1 = \square\,, \ldots, X_q = \square\,, J \mid R\,, V > 0$$

$$\boxed{\mathsf{Pell}(\mathrm{S},\,\mathrm{T}) \;=_{\mathrm{Def}}\; (\mathrm{S}^2-1)\,\mathrm{T}^2+1}$$

$$\mathsf{Pell}(A\,,\,C)\;F\;I=\square\,,\;\; F\mid B+2\,j\,C-C\,,\;\; B\leqslant C$$
$$D\leftrightharpoons\mathsf{Pell}(A\,,\,C)$$

$F\leftrightharpoons\mathsf{Pell}\big(A\,,\,2\,(i+1)\,C^2\,D\big)$	$F\leftrightharpoons\mathsf{Pell}\big(A\,,\,2\,i\,C^2\,D\big)$
$I\leftrightharpoons\mathsf{Pell}\big((F-A)\,F+A\,,\,B+2\,j\,C\big)$	$I\leftrightharpoons\mathsf{Pell}\big((A+1)\,F-A\,,\,B+2\,j\,C\big)$

Fig. 6. Polynomial specifications of the relation $\psi_A(B)=C$ (see Fig. 2). When conjoined with the constraints in the middle, the two constraints appearing on the left form an abridged formulation of the specification **A1**–**A7** recalled above from [MR75, pp. 532–533]: in this case, the unknowns i,j etc. range over $\mathbb{N}$ and the parameters A,B,C are assumed to satisfy $A>1$, $B>0$, $C>0$. Likewise, the two constraints on the right must be combined with the ones in the middle to get an abridged version of the specification of [Vse97, pp. 3203–3204]: in this case, variables range over $\mathbb{N}\setminus\{0\}$ and the assumed preconditions are $A>1$, $B>1$, and $B\equiv 1 \mod 2$; a lower overall degree results from $(A+1)\,F-A$ having superseded $(F-A)\,F+A$.

are all met if and only if the equation $M_q(X_1,\ldots,X_q,J,R,V,m)=0$ *admits solutions for some value* $\boldsymbol{m}$ *in* $\mathbb{N}$ *of the variable* m. ⊣

This theorem is exploitable in the case at hand, with $q=2$, once the two divisibility conditions (one of which is hidden inside the congruence $3\,w\,C\equiv 2\,(w^2-1) \mod Q$) are combined together by resorting to the double implication

$$d_1\mid z_1\wedge d_2\mid z_2\;\leftrightarrow\; d_1\,d_2\mid z_1d_2+z_2d_1$$

which holds when d_1,d_2,z_1,z_2 are positive integers and d_1,d_2 are co-prime. All in all, we will be able to fold our constraints into a single Diophantine polynomial equation $\mathcal{W}(k,x_1,\ldots,x_7)=0$ over $\mathbb{N}$ whose degree is 5488 (as will be assessed at the end of Sec. 3) and which admits solutions in the 7 unknowns precisely for those integer values of k which exceed 4 and which also satisfy Wolstenholme's congruence $\binom{2\,k}{k}\equiv 2 \mod k^3$.

In order to get rid of the precondition $k\geqslant 5$ (Fig. 7, right), it suffices to strenghten the inequality $K^2-4\,(C-K\,Y)^2>0$ into $(k-1)\,(k-2)\,(k-3)\,(k-4)\,\big(K^2-4\,(C-K\,Y)^2\big)>0$ before resorting to Thm. 1. Accordingly, denoting by $\bar{\mathcal{W}}(k,x_1,\ldots,x_7)$ the polynomial equation that results after this preparatory retouch, our conjectured prime-generating polynomial is:

$$x_0\,\big(1-(x_0-2)^2\,(x_0-3)^2\,\bar{\mathcal{W}}^2(x_0,x_1,\ldots,x_7)\big)\,.$$

3 Correctness of our representation of Wolstenholme's pseudoprimality

The specification of Wolstenholme's pseudoprimality which we are proposing stems from *ad hoc* modifications to [Jon82, Lemma 2.25, pp. 556–557]; hence, by bringing into our present discourse the main ingredients entering the proof thereof, we will easily get our main claim, which is:

$$\mathsf{Pell}(\mathrm{S},\mathrm{T}) \;=_{\mathrm{Def}}\; (\mathrm{S}^2-1)\,\mathrm{T}^2+1$$

$\mathsf{Pell}(A\,,\,C)\;F\;I=\square \;\wedge\; F \mid B+2\,j\,C-C$ $D \leftrightharpoons \mathsf{Pell}(A\,,\,C)$ $I \leftrightharpoons \mathsf{Pell}\big((A+1)\,F-A\,,\,B+2\,j\,C\big)$ $F \leftrightharpoons \mathsf{Pell}\big(A\,,\,2\,i\,C^2\,D\,Q\big)$	
	$K^2-4\,(C-K\,Y)^2>0$
	$\mathsf{Pell}(P\,,\,K)=\square$
	$3\,w\,C \equiv 2\,(w^2-1) \mod Q$
	$M \leftrightharpoons k\,Y$
	$Y \leftrightharpoons k^3\,s+2$
	$P \leftrightharpoons 2\,M^2\,U$
	$Q \leftrightharpoons 4\,A-5$
	$U \leftrightharpoons k^3\,w$
	$K \leftrightharpoons k+1+h\,(P-1)$
	$A \leftrightharpoons M\,(U+1)$
	$B \leftrightharpoons 2\,k+1$
$B \leqslant C$	$C \leftrightharpoons B+z$
Domain: $\mathbb{N}\setminus\{0\}$ Unknowns: i,j,m Parameters: A,B,C Precond.: $A>1\,,\;B>1\,,\;2\nmid B\,,\;C>1$ Specifies: $\psi_A(B)=C$ Sources: [MR75], [Vse97]	Domain: $\mathbb{N}\setminus\{0\}$ Unknowns: z,w,s,h,i,j,m Parameters: k Precondition: $k\geqslant 5$ Specifies: $\binom{2\,k}{k} \equiv 2 \mod k^3$ Sources: [Jon82, Lemma 2.25], L. Vallata's laurea thesis

Fig. 7. Polynomial specification of Wolstenholme's pseudoprimality.

Theorem 2. *Let $\mathcal{W}(k,z,w,s,h,i,j,m)=0$ be the Diophantine polynomial equation resulting from the system in Fig. 7, right, via Thm. 1. Then the integer values $\boldsymbol{k}\geqslant 5$ for which the congruence $\binom{2\,\boldsymbol{k}}{\boldsymbol{k}} \equiv 2 \mod \boldsymbol{k}^3$ holds are precisely the ones for which the equation $\mathcal{W}(\boldsymbol{k},z,w,s,h,i,j,m)=0$—where $\boldsymbol{k}$ has superseded the variable k—can be solved relative to the unknowns z,w,s,h,i,j,m.* ⊣

First, we need an economical—as for the number of variables involved—representation of the triadic relation $\psi_A(B)=C$. We resort to a slight variant of the one which [Vse97, Lemma 8] proposed for an even number B, because an odd B better fits our present aims.

Lemma 1. *Let A, B, C, Q be integers with $A > 1$, $B > 1$, $C > 1$, B odd, and $Q > 0$. The relationship $\psi_A(B) = C$ holds if and only if there exist i, j such that*

$$\begin{cases} DFI = \square & \text{(P1)} \\ F \mid H - C & \text{(P2)} \\ B \leqslant C & \text{(P3)} \end{cases} \qquad \begin{aligned} D &\leftrightharpoons \mathsf{Pell}(A\,,\,C) && \text{(P4)} \\ E &\leftrightharpoons 2\,i\,C^2\,D\,Q && \text{(P5)} \\ F &\leftrightharpoons \mathsf{Pell}(A\,,\,E) && \text{(P6)} \\ G &\leftrightharpoons (A+1)\,F - A && \text{(P7)} \\ H &\leftrightharpoons B + 2\,j\,C && \text{(P8)} \\ I &\leftrightharpoons \mathsf{Pell}(G\,,\,H) && \text{(P9)} \end{aligned}$$

Proof: Minor modifications to the proof of [Vse97, Lemma 8, pp. 3203–3204] (see also Remark 2 therein) yield the claim of this lemma. In its turn, that proof mimicked the proof of [MR75, Theorem 4, pp. 532–533]. □

Second, we need a Diophantine representation of exponentiation:

Lemma 2. *The relationship $S^B = Y$ holds for integers S, B, Y with $S > 0$ if and only if there exist integers A, C such that*

1. $S < A$,
2. $Y^3 < A$,
3. $S^{3B} < A$,
4. $\psi_A(B) = C$,
5. $(S^2 - 1)\,Y\,C \;\equiv\; S\,(Y^2 - 1) \mod (2\,A\,S - S^2 - 1)$.

Proof: See [Jon79, Lemma 2.8, pp. 213–214], where this result is credited to Julia Robinson. A key congruence in Jones's proof just cited is

$$(\,\ell^2 - 1\,)\,\ell^n\,\psi_{\boldsymbol{a}}(n) \equiv \ell\,(\,\ell^{2\,n} - 1\,) \mod (\,2\,\boldsymbol{a}\,\ell - \ell^2 - 1\,)\,,$$

which follows easily from Fig. 5 (0), in light of the fact that $\boldsymbol{x} = \chi_{\boldsymbol{a}}(n)$, $\boldsymbol{y} = \psi_{\boldsymbol{a}}(n)$ solves the equation $x^2 = (\boldsymbol{a}^2 - 1)\,y^2 + 1$. Making use of the easy implication

$$\boldsymbol{a} \leqslant 2\,\boldsymbol{a}\,\ell - \ell^2 - 1 \;\text{ if }\; 0 < \ell < \boldsymbol{a}\,,$$

Jones gets another key ingredient for the proof:

If $0 < \ell < \boldsymbol{a}$, $y^3 < \boldsymbol{a}$, and $z^3 < \boldsymbol{a}$ then, taken together, the congruences

$$\begin{aligned} (\,\ell^2 - 1\,)\,y\,\psi &\equiv \ell\,(\,y^2 - 1\,) \mod (2\,\boldsymbol{a}\,\ell - \ell^2 - 1), \\ (\,\ell^2 - 1\,)\,z\,\psi &\equiv \ell\,(\,z^2 - 1\,) \mod (2\,\boldsymbol{a}\,\ell - \ell^2 - 1) \end{aligned}$$

imply that $y = z$, for any number ψ.

The desired conclusion follows without difficulty. □

In the light of Lemma 1 and Lemma 2, minimal clues about the proof of Theorem 2 should suffice to the reader: we will limit ourselves to indicating the modifications which the statement of the above-cited Lemma 2.25 of [Jon82] should undergo, so that its proof can then be adapted to our case without any

substantial changes. Some variables of the cited lemma must be replaced by ours according to the rewritings: $B' \rightsquigarrow B$, $\phi \rightsquigarrow z$, $W \rightsquigarrow w$, $R \rightsquigarrow k$, and $N \rightsquigarrow k$ (notice that we are thus enforcing the equality $R = N$). Moreover, one should: remove condition (B11) $W = b\,w$ of the cited lemma; replace its conditions (B9) $U = N^2\,w$ and (B10) $Y = N^2\,s$ by ours, namely $U = k^3\,w$ and $Y = k^3\,s + 2$; add our condition $Q = 4\,A - 5$.

Degree of the polynomial through which we have represented Wolstenholme's pseudoprimality

To end, let us calculate the degree of the polynomial $\mathcal{W}(k, z, w, s, h, i, j, m)$ discussed above. To more easily get the degrees of the polynomials involved in the right-hand specification of Fig. 7, we add a few more abbreviations to it: $H \leftrightharpoons B + 2\,j\,C$, $E \leftrightharpoons 2\,i\,C^2\,D\,Q$, and $G \leftrightharpoons (A+1)\,F - A$; then we get the degree map:

$$B/1,\ \ U/4,\ Y/4;\ \ C/1,\ \ M/5;\ H/2,\ \ A/9,\ \ \ P/14;$$
$$D/20,\ Q/9,\ K/15;\ E/32;\ F/82;\ G/91;\ I/186.$$

To complete the assessment of the degree of $\mathcal{W}$, we need to make the polynomial M_q of Thm. 1 rather explicit: according to [MR75],

$$M_q(\,X_1\,,\ldots,X_q\,,\,J\,,\,R\,,\,V\,,\,m\,) \quad =_{\text{Def}} \quad \prod_{\sigma\in\{0,1\}^{\{1,\ldots,q\}}} \Big(J^2\,m + R^2 - J^2\,(\,2\,V-1\,)\,\Big(R^2 + W^q + \sum_{j=1}^{q}(-1)^{\sigma(j)}\sqrt{X_j}\,W^{j-1}\Big)\Big)\,,$$

where

$$W \leftrightharpoons 1\ +\ \sum_{i=1}^{q} X_i^2\,.$$

In the case at hand,

$$\mathcal{W}(k, z, w, s, h, i, j, m)\ =_{\text{Def}}\ M_2(\,X_1\,,\,X_2\,,\,J\,,\,R\,,\,V\,,\,m),$$

where $X_1 \leftrightharpoons DFI$ and $X_2 \leftrightharpoons \mathsf{Pell}(P\,,\,K)$; hence $q = 2$ and $W \leftrightharpoons 1 + (D\,F\,I)^2 + \big((P^2-1)\,K^2+1\big)^2$. The polynomial V which we using in a statement $V > 0$ is $V \leftrightharpoons K^2 - 4(C - KY)^2$. The polynomials J, R of which we are stating that $J \mid R$, result from combination of the two conditions $F \mid H - C$ and $3\,w\,C \equiv 2\,(w^2-1)$ mod Q: hence $J \leftrightharpoons F\,Q$ and $R \leftrightharpoons (H - C)\,Q + \big(2\,(w^2-1) - 3\,w\,C\big)\,F$. The polynomials just introduced have degrees:

$$W/576,\ V/38,\ J/91,\ R/84$$

and, consequently, $\mathcal{W}$ has the degree

$$\deg M_2 = 4\ \deg\big(\,J^2\,(\,2\,V-1\,)\,W^2\,\big) = 4 \cdot 1372 = 5488\,.$$

Conclusions and future work

After explaining what it means for a relation $\varrho(x_1, \ldots, x_n)$ to be *Diophantine in a set* $\mathcal{S}$, Julia Robinson proved in [Rob69] that every recursively enumerable set is Diophantine in *any* infinite set of primes. We do not know whether Jones's conjectured converse of Wolstenholme's theorem will be proved, hence we cannot refer Robinson's result just recalled to the set $\mathbb{W}$ of all integers $k \geqslant 5$ such that $\binom{2\,k}{k} \equiv 2 \mod k^3$, and we feel that it would add to the autonomous significance of our polynomial representation of $\mathbb{W}$ if we succeeded in showing that every recursively enumerable set is Diophantine in $\mathbb{W}$.

Albeit subject to Jones's conjecture, the result presented in this paper suggests a new estimate for the *rank* (= least possible number of unknowns in a Diophantine representation) of the set of primes, shifting it down from 9 to 7. Although this was to be expected (cf. [Mat93, p. 56]), we could not find this result published anywhere.

We would also like to determine a non-trivial *lower* bound for the rank of primality. Pietro Corvaja gave us clues that the lower bound 2 can be obtained through direct application of Siegel's theorem on integral points (see [Sie29][4]).

It is a bit deceiving that we could not benefit from the celebrated [AKS04] for the aims of this paper; an explanation might be that the complexity of prime-number recognition has to do with bounds that one can place on the *sizes* of the unknowns in a Diophantine representation of primality rather than on the number of those unknowns.

Acknowledgements

As hinted at above, we had pleasant and profitable exchanges of ideas with prof. Pietro Corvaja (University of Udine).

References

N.B. Yuri V. Matiyasevich's name was transliterated variously in his publications in English; in this bibliography, the authors have preferred conformity with the spellings found in the originals to uniformity of writing.

AKS04. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, June 2004.

CZ02. Pietro Corvaja and Umberto Zannier. A subspace theorem approach to integral points on curves. *C. R. Acad. Sci. Paris, Ser. I*, 334(4):267–271, 2002.

Dav58. Martin Davis. *Computability and Unsolvability*. McGraw-Hill, New York, 1958. Reprinted with an additional appendix, Dover 1983.

Dav73. Martin Davis. Hilbert's tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, 1973. Reprinted with corrections in the Dover edition of *Computability and Unsolvability* [Dav58].

[4] A proof of Siegel's theorem along new lines can be found in [CZ02].

DMR76. Martin Davis, Yuri Matijasevič, and Julia Robinson. Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution. In *Mathematical Developments Arising From Hilbert Problems*, volume 28 of *Proceedings of Symposia in Pure Mathematics*, pages 323–378, Providence, RI, 1976. American Mathematical Society. Reprinted in [Rob96].

Hil00. David Hilbert. Mathematische Probleme. Vortrag, gehalten auf dem internationalen Mathematiker-Kongreß zu Paris 1900. *Nachrichten von der Königliche Gesellschaft der Wissenschaften zu Göttingen*, pages 253–297, 1900.

Jon79. James P. Jones. Diophantine representation of Mersenne and Fermat primes. *Acta Arithmetica*, XXXV(3):209–221, 1979.

Jon82. James P. Jones. Universal Diophantine equation. *The Journal of Symbolic Logic*, 47(3):549–571, 1982.

JSWW76. James P. Jones, Daihachiro Sato, Hideo Wada, and Douglas Wiens. Diophantine representation of the set of prime numbers. *American Mathematical Monthly*, 83(6):449–464, 1976.

Mat70. Ju. V. Matijasevič. Diofantovost' perechislimykh mnozhestv. *Doklady Akademii Nauk SSSR*, 191(2):279–282, 1970. (Russian). (Translated as Ju. V. Matijasevič. Enumerable sets are Diophantine. *Soviet Mathematics. Doklady*, 11(2):354-358, 1970.).

Mat71. Ju. V. Matijasevič. Diophantine representation of the set of prime numbers. *Soviet Mathematics. Doklady*, 12(1):249–254, 1971.

Mat81. Yu. V. Matijasevič. Primes are nonnegative values of a polynomial in 10 variables. *Journal of Soviet Mathematics*, 15(1):33–44, 1981.

Mat93. Yuri Vladimirovich Matiyasevich. *Hilbert's tenth problem*. The MIT Press, Cambridge (MA) and London, 1993.

McI95. Richard J. McIntosh. On the converse of Wolstenholme's theorem. *Acta Arithmetica*, LXXI(4):381–389, 1995.

MR75. Yuri Matijasevič and Julia Robinson. Reduction of an arbitrary diophantine equation to one in 13 unknowns. *Acta Arithmetica*, XXVII:521–553, 1975.

Rib04. Paulo Ribenboim. *The little book of bigger primes*. Springer, 2nd edition, 2004.

Rob52. Julia Robinson. Existential definability in arithmetic. *Transactions of the American Mathematical Society*, 72(3):437–449, 1952.

Rob69. Julia Robinson. Unsolvable Diophantine problems. *Proc. Amer. Math. Soc.*, 22(2):534–538, 1969.

Rob96. Julia Robinson. *The collected works of Julia Robinson*. Number 6 in Collected Works. American Mathematical Society, Providence, RI, 1996. ISBN 0-8218-0575-4. With an introduction by Constance Reid. Edited and with a foreword by Solomon Feferman. xliv+338 pp.

Sie29. Karl Ludwig Siegel. *Über einige Anwendungen diophantischer Approximationen*. Abhandlungen der Preussischer Akademie der Wissenschaften, 1. 1929. An English translation by Clemens Fuchs is available in [Zan14].

Vse97. Maxim Aleksandrovich Vsemirnov. Infinite sets of primes, admitting Diophantine representations in eight variables. *Journal of Mathematical Sciences*, 87(1):3200–3208, 1997.

Wol62. Joseph Wolstenholme. On certain properties of prime numbers. *The Quarterly Journal of Pure and Applied Mathematics*, 5:35–39, 1862.

Zan14. Umberto Zannier, editor. *On some applications of Diophantine approximations*. Edizioni della Normale. Scuola Normale Superiore, 2014.

A natural sequent calculus for Lewis' logic of counterfactuals

Nicola Olivetti[1] and Gian Luca Pozzato[2]

[1] Dip. Informatica - Universitá di Torino - Italy
gianluca.pozzato@unito.it
[2] Aix Marseille Univ. - CNRS, ENSAM, Univ. de Toulon - LSIS UMR 7296, Marseille - France
nicola.olivetti@univ-amu.fr

Abstract. The logic $\mathbb{V}$ is the basic logic of counterfactuals in the family of Lewis' systems. It is characterized by the whole class of so-called sphere models. We propose a new sequent calculus for this logic. Our calculus takes as primitive Lewis' connective of comparative plausibility $\preceq$: a formula $A \preceq B$ intuitively means that A is at least as plausible as B, so that a conditional $A \Rightarrow B$ can be defined as A is impossible or $A \wedge \neg B$ is less plausible than A. As a difference with previous attempts, our calculus is standard in the sense that each connective is handled by a finite number of rules with a fixed and finite number of premises. Moreover our calculus is "internal", in the sense that each sequent can be directly translated into a formula of the language. The peculiarity of our calculus is that sequents contain a special kind of structures, called *blocks*, which encode a finite combination of $\preceq$. We show that the calculus is terminating, whence it provides a decision procedure for the logic $\mathbb{V}$.

1 Introduction

In the recent history of conditional logics the work by Lewis [16] has a prominent place (among others [5, 18, 13, 11]). He proposed a formalization of conditional logics in order to represent a kind of hypothetical reasoning (if A were the case then B), that cannot be captured by classical logic with material implication. More precisely, the original motivation by Lewis was to formalize *counterfactual sentences*, i.e. conditionals of the form "if A were the case then B would be the case", where A is false. But independently from counterfactual reasoning, conditional logics have found then an interest also in several fields of artificial intelligence and knowledge representation. Just to mention a few: they have been used to reason about prototypical properties [8] and to model belief change [11, 9]. Moreover, conditional logics can provide an axiomatic foundation of nonmonotonic reasoning [4, 12], here a conditional $A \Rightarrow B$ is read as "in normal circumstances if A then B". Finally, a kind of (multi)-conditional logics [2, 3] have been used to formalize epistemic change in a multi-agent setting and in epistemic "games", each conditional operator expresses the "conditional beliefs" of an agent.

In this paper we concentrate on the logic $\mathbb{V}$ of counterfactual reasoning studied by Lewis. This logic is characterized by possible world models structured by a system of spheres. Intuitively, each world is equipped with a set of nested sets of worlds: inner sets represent "most plausible worlds" from the point of view of the given world and worlds

belonging only to outer sets represent less plausible worlds. In other words, each sphere represent a degree of plausibility. The (rough) intuition involving the truth condition of a counterfactual $A \Rightarrow B$ at a world x is that B is true at the most plausible worlds where A is true, whenever there are worlds satisfying A. But Lewis is reluctant to assume that *most plausible worlds A exist* (whenever there are A-worlds), for philosophical reasons. He calls this assumption the Limit Assumption and he formulates his semantics in more general terms which do need this assumption (see below). The sphere semantics is the strongest semantics for conditional logics, in the sense that it characterizes only a subset of relatively strong systems; there are weaker (and more abstract) semantics such as the selection function semantics which characterize a wider range of systems [18].

From the point of view of proof-theory and automated deduction, conditional logics do not have a state of the art comparable with, say, the one of modal logics, where there are well-established alternative calculi, whose proof-theoretical and computational properties are well-understood. This is partially due to the mentioned lack of a unifying semantics. Similarly to modal logics and other extensions/alternative to classical logics two types of calculi have been studied: *external* calculi which make use of labels and relations on them to import the semantics into the syntax, and *internal* calculi which stay within the language, so that a "configuration" (sequent, tableaux node...) can be directly interpreted as a formula of the language. Limiting our account to Lewis' counterfactual logics, some external calculi have been proposed in [10] which presents modular labeled calculi for preferential logic PCL and its extensions, this family includes all counterfactual logics by Lewis. Internal calculi have been proposed by Gent [7] and by de Swart [6] for Lewis' logic $\mathbb{VC}$ and neighbours. These calculi manipulate sets of formulas and provide a decision procedure, although they comprise an *infinite set of rules and rules with a variable number of premises.* Finally in [15] the authors provide internal calculi for Lewis' conditional logic $\mathbb{V}$ and some extensions. Their calculi are formulated for a language comprising the comparative plausibility connective, the strong and the weak conditional operator. Both conditional operators can be defined in terms of the *comparative similarity* connective. These calculi are actually an extension of Gent's and de Swart's ones and they comprise an infinite set of rules with a variable number of premises. We mention also a seminal work by Lamarre [13] who proposed a tableaux calculus for Lewis' logic, but it is actually a model building procedure rather than a calculus made of deductive rules.

In this paper we tackle the problem of providing a standard proof-theory for Lewis' logic $\mathbb{V}$ in the form of internal calculi. By "standard" we mean that we aim to obtain analytic sequent calculi where each connective is handled by a finite number of rules with a fixed and finite number of premises. As a preliminary result, we propose a new internal calculus for Lewis' logic $\mathbb{V}$. This is the most general logic of Lewis' family and it is complete with respect the whole class of sphere models (moreover, its unnested fragment essentially coincide with KLM rational logic **R** [14]). Our calculus takes as primitive Lewis' comparative plausibility connective $\preceq$: a formula $A \preceq B$ means, intuitively, that A is at least as plausible as B, so that a conditional $A \Rightarrow B$ can be defined as A is impossible or $A \wedge \neg B$ is less plausible than A[3]. As a difference with previous

[3] This definition avoids the Limit Assumption, in the sense that it works also for models where *at least* a sphere containing A worlds does not necessarily exist.

attempts, our calculus comprises *structured* sequents containing *blocks*, where a block is a new syntactic structure encoding a finite combination of $\preceq$. In other words, we introduce a new modal operator (but still definable in the logic) which encodes finite combinations of $\preceq$. This is the main ingredient to obtain a standard and internal calculus for $\mathbb{V}$. We show that the calculus is terminating whence it provides a decision procedure. In further research we shall study its complexity and we shall study how to extend it to stronger logics of Lewis' family.

2 Lewis' logic $\mathbb{V}$

We consider a propositional language $\mathcal{L}$ generated from a set of propositional variables $Varprop$ and boolean connectives plus two special connectives $\preceq$ (comparative plausibility) and $\Rightarrow$ (conditional). A formula $A \preceq B$ is read as "A is at least as plausible as B". The semantics is defined in terms of sphere models, we take the definition by Lewis without the limit assumption.

Definition 1. *A model $\mathcal{M}$ has the form $\langle W, \$, [\,] \rangle$, where W is a non-empty set whose elements are called worlds, $[\,] : Varprop \longrightarrow Pow(W)$ is the propositional evaluation, and $\$: W \longrightarrow Pow(Pow(W))$. We write $\$_x$ for the value of the function $\$$ for $x \in W$, and we denote the elements of $\$_x$ by $\alpha, \beta ....$ Models have the following property: $\forall \alpha, \beta \in \$_x \; \alpha \subseteq \beta \vee \beta \subseteq \alpha$.*

The truth definition is the usual one for boolean cases, for the additional connectives we have: (i) $x \in [A \preceq B]$ iff $\forall \alpha \in \$_x$ if $\alpha \cap [B] \neq \emptyset$ then $\alpha \cap [A] \neq \emptyset$ (ii) $x \in [A \Rightarrow B]$ iff either $\forall \alpha \in \$_x \; \alpha \cap [A] = \emptyset$ or there is $\alpha \in \$_x$, such that $\alpha \cap [A] \neq \emptyset$ and $\alpha \cap [A \wedge \neg B] = \emptyset$.

The semantic notions, satisfiability and validity are defined as usual.

For the ease of reading we introduce the following conventions and abbreviations: we write $x \models A$, where the model is understood instead of $x \in [A]$. Moreover given $\alpha \in \$_x$, we use the following notations:

$\alpha \models^{\forall} A$ if $\alpha \subseteq [A]$, i.e. $\forall y \in \alpha$ it holds $y \models A$
$\alpha \models^{\exists} A$ if $\alpha \cap [A] \neq \emptyset$, i.e. $\exists y \in \alpha$ such that $y \models A$

Observe that with this notation, the truths conditions for $\preceq$ and $\Rightarrow$ become:

- $x \models A \preceq B$ iff $\forall \alpha \in \$_x$ either $\alpha \models^{\forall} \neg B$ or $\alpha \models^{\exists} A$
- $x \models A \Rightarrow B$ iff either $\forall \alpha \in \$_x \; \alpha \models^{\forall} \neg A$ or there is $\beta \in \$_x$ such that $\beta \models^{\exists} A$ and $\beta \models^{\forall} A \rightarrow B$.

It can be observed that the two connectives $\preceq$ and $\Rightarrow$ are interdefinable, in particular:

$$A \Rightarrow B \equiv (\bot \preceq A) \vee \neg(A \wedge \neg B \preceq A)$$

Also the $\preceq$ connective can be defined in terms of the conditional $\Rightarrow$ as follows:

$$A \preceq B \equiv (A \vee B) \Rightarrow \bot \vee \neg((A \vee B) \Rightarrow \neg A)$$

The logic $\mathbb{V}$ can be axiomatized taking as primitive the connective $\preceq$ and the axioms are the following [16]:

- classical axioms and rules
- if $B \to (A_1 \vee \ldots \vee A_n)$ then $(A_1 \preceq B) \vee \ldots \vee (A_n \preceq B)$
- $(A \preceq B) \vee (B \preceq A)$
- $(A \preceq B) \wedge (B \preceq C) \to (A \preceq C)$
- $A \Rightarrow B \equiv (\bot \preceq A) \vee \neg(A \wedge \neg B \preceq A)$

3 An internal sequent calculus for $\mathbb{V}$

In this section we present $\mathcal{I}^{\mathsf{V}}$, a structured calculus for Lewis' conditional logic $\mathbb{V}$ introduced in the previous section. In addition to ordinary formulas, sequents contains also *blocks* of the form:

$$[A_1, \ldots, A_m \triangleleft B_1, \ldots, B_n]$$

where each A_i, B_j are formulas. The interpretation is the following:

$$x \models [A_1, \ldots, A_m \triangleleft B_1, \ldots, B_n]$$

if and only if, $\forall \alpha \in \$_x$:

- either $\alpha \models^{\forall} \neg B_j$ for some j, or
- $\alpha \models^{\exists} A_i$ for some i.

A *sequent* Γ is a multiset $G_1, \ldots G_k$, where each G_i is either a formula or a block. A sequent $\Gamma = G_1, \ldots G_k$, is valid if for every model $\mathcal{M} = \langle W, \$, [\,] \rangle$, for every world $x \in W$, it holds that $x \models G_1 \vee \ldots \vee G_k$. The calculus $\mathcal{I}^{\mathsf{V}}$ comprises the following axiom and rules:

- Standard Axioms: $(i)\ \Gamma, \top \quad (ii)\ \Gamma, \neg\bot \quad (iii)\ \Gamma, P, \neg P$
- Standard external rules of sequent calculi for boolean connectives
- $(\preceq^{+})$

$$\frac{\Gamma, [A \triangleleft B]}{\Gamma, A \preceq B}\ (\preceq +)$$

- $(\preceq^{-})$

$$\frac{\Gamma, \neg(A \preceq B), [B, \Sigma \triangleleft \Pi] \quad \Gamma, \neg(A \preceq B), [\Sigma \triangleleft \Pi, A]}{\Gamma, \neg(A \preceq B), [\Sigma \triangleleft \Pi]}\ (\preceq^{-})$$

- $(\Rightarrow^{+})$

$$\frac{\Gamma, [\bot \triangleleft A], \neg(A \wedge \neg B \preceq A)}{\Gamma, A \Rightarrow B}\ (\Rightarrow^{+})$$

- $(\Rightarrow^{-})$

$$\frac{\Gamma, \neg(\bot \preceq A) \qquad \Gamma, [A \wedge \neg B \triangleleft A]}{\Gamma, \neg(A \Rightarrow B)}\ (\Rightarrow^{-})$$

- (Communication)

$$\frac{\Gamma, [\Sigma_1 \triangleleft \Pi_1, \Pi_2], [\Sigma_1, \Sigma_2 \triangleleft \Pi_2] \quad \Gamma, [\Sigma_2 \triangleleft \Pi_1, \Pi_2], [\Sigma_1, \Sigma_2 \triangleleft \Pi_1]}{\Gamma, [\Sigma_1 \triangleleft \Pi_1], [\Sigma_2 \triangleleft \Pi_2]}\ (Com)$$

– $(Jump)$

$$\frac{\neg B_i, \Sigma}{\Gamma, [\Sigma \lhd B_1, \ldots, B_n]}\ (Jump)$$

Some remark on the rules: the rule $(\preceq^+)$ just introduces the block structure, showing that $\lhd$ is a generalization of $\preceq$; $(\preceq^-)$ prescribes case analysis and contribute to expand the blocks; the rules $(\Rightarrow^+)$ and $(\Rightarrow^-)$ just apply the definition of $\Rightarrow$ in terms of $\preceq$. The (Com) rule is directly motivated by the *nesting* of spheres, which means a *linear order* on sphere inclusion; this rule is very similar to the homonymous one used in *hypersequent* calculi for handling truth in linearly ordered structures [1, 17].

As usual, given a formula $G \in \mathcal{L}$, in order to check whether G is valid we look for a derivation of G in the calculus $\mathcal{I}^{\mathsf{V}}$. Given a sequent Γ, we say that Γ is derivable in $\mathcal{I}^{\mathsf{V}}$ if it admits a *derivation*. A derivation of Γ is a tree where: the root is Γ; a leaf is an instance of standard axioms; a non-leaf node is (an instance of) the conclusion of a rule having (an instance of) the premises of the rule as parents.

Here below we show a derivation of $(A \preceq B) \vee (B \preceq A)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg A, A}{[A \lhd B, A], [A, B \lhd A]}\,(Jump) \quad \dfrac{\neg B, B}{[B \lhd B, A], [A, B \lhd B]}\,(Jump)}{[A \lhd B], [B \lhd A]}\,(Com)}{[A \lhd B], B \preceq A}\,(\preceq^+)}{A \preceq B, B \preceq A}\,(\preceq^+)}{(A \preceq B) \vee (B \preceq A)}\,(\vee^+)$$

It can be shown that the calculus $\mathcal{I}^{\mathsf{V}}$ is sound, complete and terminating if rules are applied without redundancy[4]:

Theorem 1. *Given a sequent Γ, Γ is derivable if and only if it is valid. Given a sequent Γ, any non-redundant derivation-tree of Γ is finite.*

4 Conclusions

In this paper we begin a proof-theoretical investigation of Lewis' logics of counterfactuals characterized by the sphere-model semantics. We have presented a simple, analytic calculus $\mathcal{I}^{\mathsf{V}}$ for logic $\mathbb{V}$, the most general logic characterized by the sphere-model semantics. The calculus is *standard*, that is to say it contains a finite a number of rules with a fixed number of premisses and *internal* in the sense that each sequent denotes a formula of $\mathbb{V}$. The novel ingredient of $\mathcal{I}^{\mathsf{V}}$ is that sequents are structured objects containing blocks, where a block is a structure or a sort of n-ary modality encoding a finite combination of formulas with the connective $\preceq$. The calculus $\mathcal{I}^{\mathsf{V}}$ ensures termination, and therefore it provides a decision procedure for $\mathbb{V}$.

[4] Detailed proofs are confined in the accompanying report [19].

In future research, we aim at extending our approach to all the other conditional logics of the Lewis' family, in particular we aim at focusing on the logics $\mathbb{VT}$, $\mathbb{VW}$ and $\mathbb{VC}$. Moreover, we shall study the complexity of the calculus $\mathcal{I}^{\mathsf{V}}$ with the hope of obtaining optimal calculi.

References

1. A. Avron. The method of hypersequents in the proof theory of propositional non-classical logics. In Wilfrid Hodges, Martin Hyland, Charles Steinhorn, and John Truss, editors, *Logic: from foundations to applications.*, pages 1–32. Oxford University Press, New York, 1996.
2. Alexandru Baltag and Sonja Smets. The logic of conditional doxastic actions. *Texts in Logic and Games, Special Issue on New Perspectives on Games and Interaction*, 4:9–31, 2008.
3. Oliver Board. Dynamic interactive epistemology. *Games and Economic Behavior*, 49(1):49–80, 2004.
4. C. Boutilier. Conditional logics of normality: a modal approach. *Artificial Intelligence*, 68(1):87–154, 1994.
5. B. F. Chellas. Basic conditional logics. *Journal of Philosophical Logic*, 4:133–153, 1975.
6. H. C. M. de Swart. A gentzen- or beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics vc and vcs. *Journal of Symbolic Logic*, 48(1):1–20, 1983.
7. I. P. Gent. A sequent or tableaux-style system for lewis's counterfactual logic vc. *Notre Dame Journal of Formal Logic*, 33(3):369–382, 1992.
8. M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, 1986.
9. Laura Giordano, Valentina Gliozzi, and Nicola Olivetti. Weak AGM postulates and strong ramsey test: A logical formalization. *Artificial Intelligence*, 168(1-2):1–37, 2005.
10. Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Camilla Schwind. Tableau calculus for preference-based conditional logics: Pcl and its extensions. *ACM Trans. Comput. Logic*, 10(3), 2009.
11. G. Grahne. Updates and counterfactuals. *Journal of Logic and Computation*, 8(1):87–117, 1998.
12. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.
13. P. Lamarre. Etude des raisonnements non-monotones: Apports des logiques des conditionnels et des logiques modales. *PhD thesis, Université Paul Sabatier, Toulouse*, 1992.
14. D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.
15. Bjoern Lellmann and Dirk Pattinson. Sequent Systems for Lewis' Conditional Logics. In Jérôme Mengin Luis Fariñas del Cerro, Andreas Herzig, editor, *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012*, volume 7519 of *Lecture Notes in Artificial Intelligence (LNAI)*, page to appear, Toulouse, France, Sptember 2012. Springer-Verlag.
16. D. Lewis. Counterfactuals. *Basil Blackwell Ltd*, 1973.
17. George Metcalfe, Nicola Olivetti, and Dov Gabbay. *Proof Theory for Fuzzy Logics*. Springer, 2010.
18. D. Nute. Topics in conditional logic. *Reidel, Dordrecht*, 1980.
19. N. Olivetti and G. L. Pozzato. A sequent calculus for Lewis logic V: preliminary results. Technical Report –, Dipartimento di Informatica, Università degli Studi di Torino, Italy, July 2015.

Infinite Derivations as Failures

Andrea Corradi and Federico Frassetto

DIBRIS, Università di Genova, Italy
name . surname@dibris.unige.it

Abstract. When operating on cyclic data, programmers have to take care in assuring that their programs will terminate; in our opinion, this is a task for the interpreter. We present a Prolog meta-interpreter that checks for the presence of cyclic computations at runtime and returns a failure if this is the case, thus allowing inductive predicates to properly deal with cyclic terms.

Keywords: Logic Programming, Non-Termination, Inductive Semantics

1 Introduction

Sometimes, non-termination is the desired behavior of a program; most of the time, it is not. Nevertheless, in a programming language in which all programs terminate, there are always-terminating programs that cannot be written in it [11]. Non-termination is a necessary evil, a powerful feature that we *need*, but we do not actually *want* most of the time.

No work has been done to apply inductive semantics to the complete Herbrand model of Coinductive Logic Programming [8]. With the usual operational semantics, structural-recursive inductive predicates diverge when operating on infinite terms. Since infinite derivations do not belong to the inductive semantics [3], we propose a new operational semantics where infinite regular derivations of inductive predicate fail. While this implies a performance overhead, sometimes it is what a programmer would have done by hand, but faster and less error-prone.

In section 2, we illustrate briefly the notions of (co-)inductive model and give the declarative and operational semantics of co-logic programming. In section 3, we present our contribution, both as operational semantics and as Prolog code, and show some examples of use. In section 4, we compare our meta-interpreter with a standard Prolog interpreter. In section 5, we compare our work with the state of the art, draw conclusions and state possible future work.

2 About Inductive and Coinductive Semantics

Let the Herbrand universe $\mathcal{H}$ be a set of terms. We denote a *clause* as $A \leftarrow B_1, \ldots, B_n$ where $A \in \mathcal{H}$ and $\forall\, i \in \{1 \ldots n\}\ .\ B_i \in \mathcal{H}$. A *logic program* P is a couple (F, C), where F is a set of predicate symbols and C is a set of clauses. The *Herbrand base of* P, written $\mathcal{H}_P$, is the set of all the ground terms of the

form $p(\bar{t})$, where $p \in F$ and $\bar{t}$ is a tuple of terms. An interpretation of P is an element of $\mathcal{P}(\mathcal{H}_P)$, where $\mathcal{P}$ denotes the power-set constructor. A logic program P induces a *one-step-inference function* $\mathcal{I}_P$.

$$\mathcal{I}_P \colon \mathcal{P}(\mathcal{H}_P) \to \mathcal{P}(\mathcal{H}_P)$$
$$S \mapsto \{A \in \mathcal{H}_P \mid (A \leftarrow B_1, \ldots, B_n) \in P \wedge \forall i \in \{1 \ldots n\} \,.\, B_i \in S\}$$

A *model* of a logic program P is a fix-point of $\mathcal{I}_P$, that is, an interpretation S such that $\mathcal{I}_P(S) = S$.

Consider the logic program $\{p(1,2) \leftarrow \texttt{true}\} \cup \{p(X,X) \leftarrow p(X,X) \mid X \in \mathcal{H}\}$: written below in Prolog syntax.

```
p(1,2).
p(X,X) :- p(X,X).
```

both $\{p(1,2)\}$ and $\{p(1,2)\} \cup \{p(X,X) \in \mathcal{H}_\mathcal{P} \mid X \in \mathcal{H}_\mathcal{P}\}$ are models. Due to the Knaster–Tarski theorem [10], any logic program has a smallest model, called *inductive*, and a biggest one, called *coinductive*.

The usual Logic Programming (LP) semantics is inductive, therefore $\mathcal{H}_P$ contains only atoms inferred by finite derivations [3]; using the coinductive interpretation, we obtain the complete Herbrand base that contains also atoms inferred by infinite derivations. Simon et al. [7,8] have extended LP to co-Logic Programming (co-LP), which allows the programmer to choose between inductive and coinductive semantics for each predicate. The Prolog interpreters SWI-Prolog and YAP support co-LP.

We show the co-LP operational semantics as given by Ancona and Dovier[1] [2]. An *hypothetical goal* G is a couple $\langle E \,\square\, L \rangle$, where E is a set of equations and L is a list of pairs (A, S), where A is an atom and S is a set of atoms that represents the ancestors of A in the call stack. Given a program $\mathcal{P}$ and two hypothetical goals $G = \langle E \,\square\, (p(s_1, \ldots, s_n), S_1), (A_2, S_2), \ldots, (A_u, S_u) \rangle$ and G', we define the rewriting rule $G \vdash_{\text{co}} G'$ by cases as follows:

1. if p is a coinductive predicate and there is an atom $p(t_1, \ldots, t_n)$ in S_1 such that $E' = E \cup \{s_1 = t_1, \ldots, s_n = t_n\}$ is solvable, then $G' = \langle E' \,\square\, (A_2, S_2), \ldots, (A_u, S_u) \rangle$;
2. if there is a clause $p(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m$ that is a renaming of a clause in $\mathcal{P}$ with fresh variables and $E' = E \cup \{s_1 = t_1, \ldots, s_n = t_n\}$ is solvable, then $G' = \langle E' \,\square\, (B_1, S'), \ldots, (B_m, S'), (A_2, S_2), \ldots, (A_u, S_u) \rangle$ where $S' = S_1 \cup \{p(s_1, \ldots, s_n)\}$.

3 Finite failure

Let us consider the problem of checking whether an element appears in a list.

```
member(E,[E|_]).
member(E,[_|L]) :- member(E,L).
```

[1] The original semantics allows expanding any subgoal. For simplicity, our semantics always expands the first.

This naïve Prolog definition does not terminate if the second parameter is a cyclic list not containing `E`. This is not the desired behavior: we would like `member` to either succeed or fail in a finite amount of time. Simply applying coinduction to `member` leads to an erroneous semantics: for example, `L=[1|L], member(2,L)` succeeds even if `L` does not contain `2`, because the second call unifies with the first.

The definition to let it work on infinite lists leads to a more convoluted predicate `member2` that relies on the extra-logical cut operator `!`.

```
member2(E,L) :- member2([],E,L).
member2(H,E,L) :- member(L,H), !, fail.
member2(_,E,[E|_]).
member2(H,E,L) :- L=[_|T], member2([L|H],E,T).
```

The predicate we are trying to define is inherently inductive [7]. Our aim is to allow inductive predicates to work correctly on the complete Herbrand base. Since infinite derivations are not computable, the resolution procedure fails when it finds one.

3.1 Operational semantics

We give the operational semantics where the inductive predicates fail when the derivation diverges. We define the rewriting rule $\Vdash_{\overline{\text{co}}}$ in a similar way to rule $\vdash_{\overline{\text{co}}}$ in section 2, except for the second point:

2. if p is not inductive or if there is not an atom $p(t_1, \ldots, t_n)$ in S_1 such that $E' = E \cup \{s_1 = t_1, \ldots, s_n = t_n\}$ is solvable, use the second point of the definition of $\vdash_{\overline{\text{co}}}$.

In the second point if the p is coinductive we can behaves as $\vdash_{\overline{\text{co}}}$; if p is inductive we can proceed with the resolution only if it is not already in the call stack, $\Vdash_{\overline{\text{co}}}$ behaves as $\vdash_{\overline{\text{co}}}$. If it is in the call stack than atom could not be resolved and the resolution will fails.

3.2 Meta-interpreter

The implementation of a meta-interpreter follows the implementation of [1,7,8] but it returns a failure when it finds a cycle during the resolution inductive predicate. It is sound and complete with respect to $\Vdash_{\overline{\text{co}}}$.

```
cosld(G) :- solve([],G).
solve(H,(G1,G2)) :- !, solve(H,G1), solve(H,G2).
solve(_,A) :- built_in(A), !, A.
solve(H,A) :- member(A,H), !, coinductive(A).
solve(H,A) :- clause(A,As), solve([A|H],As).
coinductive(1).
```

The first argument of `solve` is the list of already processed atoms, used to avoid infinite computation; the second is the goal to resolve. The predicate `solve` has four clauses:

1. resolution distributes on conjunction as usual, with the same `H` in both calls, because it depends only on the ancestors;
2. resolution for `built_in` predicates is the default one;
3. if the atom `A` is in the hypotheses, the resolution succeeds if `A` is coinductive and fails otherwise;
4. if none of the above applies, the resolution proceeds normally.

Here we can avoid keeping the coinductive hypotheses for every atom in the goals, because we exploit the Prolog call stack to have the same result.

To mark a predicate p as coinductive, the source file must contain the fact `coinductive(`p`(_,...,_))`.. The declaration `coinductive(1)` assures that the `coinductive` predicate is defined even if there are no coinductive predicates.

This meta-interpreter keeps track of every non-`built_in` atom. We can improve efficiency by requiring to explicitly mark the inductive predicates and using the standard resolution procedure for the unmarked ones.

3.3 More Examples

Infinite trees. Let us represent a tree as `t(E,Ts)`, where `E` is the element of the node and `Ts` is a finite list of sub-trees.

The predicate `member_tree(E,T)` searches a tree `T` for a node with element `E`.

```
member_tree(E,t(E, _)).
member_tree(E,t(_,Ts)) :- member(T,Ts), member_tree(E,T).
```

When using a standard interpreter, similarly to `member` and infinite lists, the predicate `member_tree` is not guaranteed to terminate with an infinite tree; for example, `T1=t(1,[T1,T2]), T2=t(2,[T2,T3]), T3=t(3,[T3]), member_tree(3,T1)` loops forever. Using our meta-interpreter, the goal succeeds correctly: during the resolution of the first recursive call `member_tree(3,T)`, the call stack contains the atom `member_tree(3,T1)`; `T` unifies with `T1`, so the goal fails; then, backtracking takes place, `T` unifies with `T2` and the resolution continues in similar way until reaching `member_tree(3,T3)`.

Graphs. Finding a path between two nodes is a common operation on graphs. We encode the previous tree as a graph represented by an adjacency list, obtaining `[1-[1,2],2-[2,3],3-[1,3]]`.

The predicate `path(N1,N2,G,P)` searches for a path `P` from the node `N1` to the node `N2` in the graph `G`.

```
path(N1,N1,G,[N1]) :- neighbors(N1,G,_).
path(N1,N2,G,[N1|P]) :- neighbors(N1,G,Ns), member(N3,Ns),
                        path(N3,N2,G,P).
neighbors(N, [N-Ns|_], Ns).
neighbors(N1, [N2-_|G], Ns) :- N1 \= N2, neighbors(N1, G, Ns).
```

4 Comparison with ISO Prolog

Our meta-interpreter and a standard Prolog interpreter are not complete with respect to the inductive interpretation and not comparable to each other. Let us look at the following predicate.

```
p(3).
p(A) :- p(B).
```

In the inductive interpretation of the predicate `p`, `p(`x`)` holds for any x. This is the behavior of the standard interpreter; but, in our meta-interpreter, `p(`x`)` fails for any $x \neq 3$, because `A` and `B` always unify.

Consider the predicate `member`, but with the order of the clauses reversed.

```
member(E,[_|L]) :- member(L).
member(E,[E|_]).
```

In this case, the query `L=[1|L], member(1,L)` should succeed, because `(1,L)` is in the inductive interpretation. The query succeeds with our meta-interpreter, but does not terminate with a standard Prolog interpreter.

5 Conclusion

In this work we show an operational semantics that allow inductive predicates to behave correctly on the complete Herbrand base; moreover we show how it is possible with a simple meta-interpreter to have inductive predicates that fail when they have a infinite, but rational, proof.

A meta-interpreter by Ancona [1] executes a user-specified predicate when it finds a cycle: it can simulate ours by executing `fail`. Moura obtained similar results [5]. This system is more expressive, but the relation with the standard Prolog semantics is not clear.

Tabling [6,9,4] uses a table to store the sub-goals encountered during the evaluation and their answers. When it finds the same sub-goal again, it uses the information from the table; this allows to increase performance and ensure termination. In tabling, when a goal is found in the table but its answer is not yet available, instead of failing as we do, it *suspends* the current evaluation and try another clause. When the resolution of this second clause finish providing an answer it *resume* the first one and continue its evaluations using the the answer. We do not have such behavior, and in this respect, our semantics is similar to the classical Prolog one.

As future work, we plan to prove soundness with respect to the inductive semantics and to investigate whether completeness is attainable. Unification is not ideal for checking whether the resolution procedure is in a loop, because it succeeds too often. We are looking for a relation that better suits our needs.

References

1. Ancona, D.: Regular corecursion in Prolog. Computer Languages, Systems and Structures 39(4), 142–162 (Dec 2013)

2. Ancona, D., Dovier, A.: A theoretical perspective of coinductive logic programming. Tech. rep., University of Genova and University of Udine (2015)
3. Leroy, X., Grall, H.: Coinductive big-step operational semantics. Information and Computation 207(2), 284–304 (Feb 2009)
4. Mantadelis, T., Rocha, R., Moura, P.: Tabling, rational terms, and coinduction finally together! Theory and Practice of Logic Programming 14(Special Issue 4-5), 429–443 (Jul 2014)
5. Moura, P.: A portable and efficient implementation of coinductive logic programming. In: Sagonas, K. (ed.) Pratical Aspects of Declarative Languages. Lecture Notes in Computer Science, vol. 7752, pp. 77–92. Springer (Jan 2013)
6. Rocha, R., Silva, F., Santos Costa, V.: On applying or-parallelism and tabling to logic programs. Theory and Practice of Logic Programming 5(1-2), 161–205 (Mar 2005)
7. Simon, L., Mallya, A., Bansal, A., Gupta, G.: Co-logic programming: Extending logic programming with coinduction. Lecture Notes in Computer Science, vol. 4596, pp. 472–483. Springer (2007)
8. Simon, L.E.: Extending logic programming with coinduction. Ph.D. thesis, University of Texas at Dallas (Aug 2006)
9. Swift, T., Warren, D.S.: XSB: Extending Prolog with tabled logic programming. Theory and Practice of Logic Programming 12(Special Issue 1-2), 157–187 (Jan 2012)
10. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5(2), 285–309 (1955)
11. Turner, D.A.: Total functional programming. Journal of Universal Computer Science 10(7), 751–768 (Jul 2004)

On the first-order rewritability of conjunctive queries over binary guarded existential rules (extended abstract)

Cristina Civili, Riccardo Rosati

Dipartimento di Ingegneria informatica, automatica e gestionale
Sapienza Università di Roma

Abstract. We study conjunctive query answering and first-order rewritability of conjunctive queries for binary guarded existential rules. In particular, we prove that the problem of establishing whether a given set of binary guarded existential rules is such that all conjunctive queries admit a first-order rewriting is decidable, and present a technique for solving this problem. These results have a important practical impact, since they make it possible to identify those sets of binary guarded existential rules for which it is possible to answer every conjunctive query through query rewriting and standard evaluation of a first-order query (actually, a union of conjunctive queries) over a relational database system.

1 Introduction

Ontology-based query answering [1, 13, 6], now a consolidated topic in knowledge representation and database theory, studies the problem of answering expressive queries posed to a knowledge base that represents information both at the intensional and at the extensional level. The knowledge base is interpreted under the open-world assumption, which constitutes the main difference between this form of query answering and the standard query answering over relational databases. *Description Logics* or, more recently, *existential rules* are mostly used as the formalism to express knowledge bases, and *conjunctive queries (CQs)* are the query language usually considered in this setting. In this paper we focus on the framework existential rules, a.k.a. Datalog+/-, which extends Datalog with existential variables in the head of rules [6, 2].

Query rewriting is a well-known approach to ontology-based query answering (see, e.g., [8, 15, 11, 12]). In this approach, the knowledge base is divided into an *intensional* component, called the *ontology* (which in this paper is a set of existential rules), and an *extensional* component, which is usually a relational database instance. A query (CQ) posed to the knowledge base is first rewritten into a new query using the ontology only; the reformulated query constitutes a so-called *perfect* rewriting of the initial query, in the sense that its evaluation over the database produces exactly the *certain answers* to the query, i.e., the answers that hold in all the models of the knowledge base. This modularized strategy has several benefits, especially when the ontology $\mathcal{O}$ and the given query q enjoy the property called *first-order rewritability* (or *FO-rewritability*): this property holds if and only if there exists a *FO-rewriting* of q for $\mathcal{O}$, i.e., a first-order query that constitutes a perfect rewriting of the given query q with respect to the

ontology $\mathcal{O}$. FO-rewritability has an important practial implication, since it allows for solving the ontology-based query answering problem through standard evaluation of an SQL query over a relational database [8].

Most of the existing research related to FO-rewritability has tried to identify ontology languages that enjoy such a property: i.e., languages such that the FO-rewritability holds for every ontology $\mathcal{O}$ in this language and for every CQ q [8, 1, 7, 2, 9, 10]. For non-FO-rewritable ontology languages, the work has mostly focused on the identification of the FO-rewritability property for *single* pairs constituted by an ontology $\mathcal{O}$ and a query q. In particular, [3] has studied this problem when $\mathcal{O}$ is expressed in a fragment of existential rules and when q is a conjunctive query; while [5] have considered the case when $\mathcal{O}$ is expressed in Horn Description Logics and q is a query retrieving all the instances of a predicate.

In this paper we study a problem that is in the middle between the FO-rewritability of an ontology language and the FO-rewritability of a a specific query for a specific set of existential rules. We are interested in the problem of deciding whether a given set of existential rules $\mathcal{R}$ is such that *all* CQs over such an ontology admit a FO-rewriting for $\mathcal{R}$. In particular, we consider the class of *binary* and *guarded* existential rules (denoted by *BGER*s) and prove that such a task is decidable. To this aim, we prove the following crucial property: if a set of existential rules $\mathcal{R}$ is such that all *atomic queries* (that is, the conjunctive queries composed of a single atom) admit a FO-rewriting for $\mathcal{R}$, then *all conjunctive queries* admit a FO-rewriting for $\mathcal{R}$. So, to decide the FO-rewritability of all CQs for a set *BGER*s, it suffices to decide the FO-rewritability of atomic queries for such a set of existential rules.

We then use the results of [3], which shows that deciding the FO-rewritability of an atomic query for a set of *BGER*s is decidable: since the number of relevant atomic queries for a finite set of existential rules is finite, this result immediately implies the decidability of the problem of deciding CQ-FO-rewritability of a set of *BGER*s. This result has an important practical impact, since it proves the possibility of identifying those sets of *BGER*s for which it is always possible to answer a conjunctive query through query rewriting and standard evaluation of a first-order query (actually, a union of conjunctive queries) over a relational database system.

2 Deciding CQ-FO-rewritability of BGERs

An *existential rule* ρ over a signature Σ is a first-order logic expression of the form $\forall \bar{\mathbf{x}} \forall \bar{\mathbf{y}}\ \Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rightarrow \exists \bar{\mathbf{z}}\ \alpha(\bar{\mathbf{x}}, \bar{\mathbf{z}})$ where $\alpha(\bar{\mathbf{x}}, \bar{\mathbf{z}})$ is an atom, called the head of ρ ($head(\rho)$), $\Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a conjunction of atoms, called the body of ρ ($body(\rho)$), and $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}}$ are sequence of variables. An atom is an expression of the form $R(t_1, \ldots, t_n)$ where R is a predicate (or relation name) in Σ and $t_1, \ldots, t_n$ are called terms. We refer to the number of terms in R as the *arity* of R. We only consider variables as terms.

We will use a simplified notation for existential rules in which we omit the universal quantifiers of the body and we replace the conjunction symbol with a comma (e.g. $\rho : P(x, y), S(y, z) \rightarrow \exists w\ T(x, w)$).

A *binary* existential rule is an existential rule where all the atoms have at most an arity of 2. A *guarded* existential rule is an existential rule such that there exists an atom

in its body that contains all the universally quantified variables of the rule; such an atom is called a guard. In this work we focus on *binary guarded existential rules (BGER)*, i.e., sets of existential rules that are both binary and guarded.

Notice that the TGD mentioned above is binary, but not a guarded, while the TGD $\rho : P(x, y), S(y, x) \rightarrow \exists w\ T(x, w))$ is both binary and guarded.

A database over a signature Σ is a set of ground atoms. Given a signature Σ, a set of existential rules $\mathcal{R}$ over Σ, and a database D over Σ, a model for $\langle \mathcal{R}, D \rangle$ is a first-order interpretation that satisfies all formulas in $\mathcal{R} \cup D$.

Concerning queries, we consider *conjunctive queries* (CQs) over a signature Σ, that are represented through expressions of the form $q(\bar{\mathbf{x}}) \leftarrow \exists \bar{\mathbf{y}} \Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, where $\bar{\mathbf{x}}$ are the distinguished variables of the query, $\bar{\mathbf{y}}$ are the existentially quantified variables of the query, and $\Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is a conjunction of atoms of the form $R(t_1, \ldots, t_n)$, where $R \in \Sigma$ and $t_1, \ldots, t_n$ are variables in $\bar{\mathbf{x}}$ or $\bar{\mathbf{y}}$. A boolean conjunctive query (BCQ) is a CQ with no distinguished variables. We also consider *atomic queries*, i.e., CQs of the above form where $\Phi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is constituted of a single atom.

The certain answers to q over $\langle \mathcal{R}, D \rangle$ (notation $cert(q, \langle \mathcal{R}, D \rangle)$ are all the tuples $\bar{a}$ of constants such that $\mathcal{I} \models q(\bar{a})$ for every interpretation $\mathcal{I}$ of $\langle \mathcal{R}, D \rangle$, where $q(\bar{a})$ is the BCQ obtained by replacing $\bar{\mathbf{x}}$ with $\bar{a}$ in q. If q is a BCQ, the certain answer to q over $\langle \mathcal{R}, D \rangle$ is true if $\mathcal{I} \models q$ for every interpretation $\mathcal{I}$ of $\langle \mathcal{R}, D \rangle$ (notation $\langle \mathcal{R}, D \rangle \models q$), and false otherwise (notation $\langle \mathcal{R}, D \rangle \not\models q$). Then, let q be a CQ, and let q' be a first-order query, we say that q' is a *perfect rewriting* of q w.r.t. a set of existential rules $\mathcal{R}$ if, for each database D, $cert(q, \langle \mathcal{R}, D \rangle) = cert(q', \langle \emptyset, D \rangle)$. Moreover, we say that q is first-order rewritable (or *FO-rewritable*) for $\mathcal{R}$ if there exists a FO query q', such that q' is a perfect rewriting of q w.r.t. $\mathcal{R}$.

Definition 1. *Let $\mathcal{R}$ be a set of BGERs. We say that $\mathcal{R}$ is* CQ-FO-rewritable *if every CQ over $\mathcal{R}$ is FO-rewritable for $\mathcal{R}$. Moreover, we say that $\mathcal{R}$ is* atom-FO-rewritable *if every atomic query over $\mathcal{R}$ is FO-rewritable for $\mathcal{R}$.*

We now present the main result of this paper.

Theorem 1. *Let $\mathcal{R}$ be a set of BGERs. If $\mathcal{R}$ is atom-FO-rewritable then $\mathcal{R}$ is CQ-FO-rewritable.*

To prove the theorem, we make use of a conjunctive query rewriting technique for *BGER*s. Specifically, we make use of the general technique presented in [12] for rewriting conjunctive queries over existential rules, which can also be applied to *BGER*s. Such a technique generates a perfect rewriting in the form of a union of CQs, in particular a set of non-redundant CQs (i.e., no CQ is contained into another CQ): such a set may be finite (which implies that q is FO-rewritable for $\mathcal{R}$) or infinite. We classify the rewriting steps of this technique (that perform a form of resolution between a CQ and an inclusion axiom of the set of existential rules) into two categories: those that involve only "descendants" of a single atom of the initial query, and call such resolution steps *single-ancestor rewriting steps*, and those that involve descendants of at least two atoms of the initial query, and call such resolution steps *multiple-ancestor rewriting steps*.

We are then able to prove the following lemma, which states that, if all atomic queries are FO-rewritable for a set of existential rules $\mathcal{R}$, then the application of single-

ancestor rewriting steps only cannot lead to generating an unbounded number of rewritings of a CQ.

Lemma 1. *Let $\mathcal{R}$ be a set of BGERs such that $\mathcal{R}$ is atom-FO-rewritable and let q be a conjunctive query. If the rewriting of query q for $\mathcal{R}$ only applies single-ancestor rewriting steps, then it generates a finite set of CQs.*

The above lemma allows us to prove Theorem 1. Indeed, from such a lemma, it follows that, under the hypothesis that $\mathcal{R}$ is atom-FO-rewritable, if a CQ is not FO-rewritable for $\mathcal{R}$, then it must be possible to perform an infinite sequence of rewriting steps containing infinite multiple-ancestor rewriting steps (and generating non-redundant CQs). However, we show that, due to the restricted form of *BGER*s, this is impossible: roughly, the reason is that every multiple-ancestor step either eliminates a variable occurring in the initial query or generates an isolated subquery (i.e., a subquery not connected by existential variables to the rest of the query).

Example 1. Let $\mathcal{R}$ be the following set of *BGER*s:

$$\begin{aligned} P(x,y), R(y,x) &\rightarrow \exists z\ S(y,z) \\ R(y,x) &\rightarrow P(x,y) \\ S(x,y), S(y,x) &\rightarrow \exists z\ R(x,z) \end{aligned}$$

$\mathcal{R}$ is atom-FO-rewritable, since it can be verified that all the atomic queries over the signature of $\mathcal{R}$ are FO-rewritable. Thus, by Theorem 1, $\mathcal{R}$ is also CQ-FO-rewritable.

As an example, we provide a perfect rewriting of the query $q() \leftarrow S(x,y)$:

$$\begin{array}{ll} q() \leftarrow S(x,y) & q() \leftarrow R(z_0,x) \\ q() \leftarrow P(x,z_0), R(z_0,x) & q() \leftarrow S(z_0,z_1), S(z_1,z_0) \end{array}$$

In [3], it has been proved that the FO-rewritability of an answer-guarded conjunctive query over a set of *BGER*s is decidable. In particular, the following property directly follows from Theorem 11 in [3]:

Proposition 1. *For every set $\mathcal{R}$ of BGERs, and for every atomic conjunctive query q, one can effectively find a GN-Datalog program that computes the certain answers to q.*

GN-Datalog programs are a subclass of Datalog programs that enjoys the following property:

Proposition 2 ([4], Corollary 8.9). *For GN-Datalog programs, boundedness over finite instances is decidable and coincides with boundedness over unrestricted instances.*

From the above propositions, and from Theorem 1, it is possible to derive the following technique for deciding the FO-rewritability of a set of *BGER*s $\mathcal{R}$ over a finite signature Σ:

1. compute the set $\mathcal{Q}$ of all possible atomic queries over Σ;
2. for each atomic query $q \in \mathcal{Q}$, find the GN-Datalog program P that computes the certain answers to q over $\mathcal{R}$ and add it to $\mathcal{P}_{\mathcal{R}}$;

3. if there exists an unbounded program $P \in \mathcal{P}_{\mathcal{R}}$, then return false, otherwise return true.

Based on the above technique, we are able to show the following property.

Theorem 2. *Let $\mathcal{R}$ be a set of BGERs. Establishing the CQ-FO-rewritability of $\mathcal{R}$ is decidable.*

3 Conclusions

The work that is the closest to the present one is [14], which shows a property analogous to Theorem 1 for the description logic $\mathcal{ELI}$, which corresponds to a subclass of binary guarded existential rules.

We are currently working at extending the results presented in this paper to more expressive existential rules. Also, we would like to study the possibility of optimizing the technique for deciding the FO-rewritability of atomic queries for the case of *BGER*s.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338).

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
2. J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9–10):1620–1654, 2011.
3. V. Bárány, M. Benedikt, and B. ten Cate. Rewriting guarded negation queries. In *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013*, pages 98–110, 2013.
4. V. Bárány, B. ten Cate, and M. Otto. Queries with guarded negation. *Proc. of the 38th Int. Conf. on Very Large Data Bases (VLDB 2012)*, 5(11):1328–1339, July 2012.
5. M. Bienvenu, C. Lutz, and F. Wolter. First-order rewritability of atomic queries in horn description logics. In *Proc. of the 23st Int. Joint Conf. on Artificial Intelligence (IJCAI 2013)*, pages 754–760. AAAI Press, 2013.
6. A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *Semantic Web J.*, 14:57–83, 2012.
7. A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
9. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artificial Intelligence*, 195:335–360, 2013.
10. C. Civili and R. Rosati. A broad class of first-order rewritable tuple-generating dependencies. In *Proc. of the 2nd Datalog 2.0 Workshop*, 2012.
11. G. Gottlob, G. Orsi, and A. Pieris. Ontological queries: Rewriting and optimization. In *Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE 2011)*, pages 2–13, 2011.

12. M. König, M. Leclere, M.-L. Mugnier, and M. Thomazo. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web J.*, 2013.
13. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 247–257, 2010.
14. C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In *Proc. of the 24th Int. Workshop on Description Logic (DL 2011)*, 2011.
15. H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Applied Logic*, 8(2):186–209, 2010.

Computational Thinking for Beginners: A Successful Experience using Prolog

Silvio Beux, Daniela Briola, Andrea Corradi, Giorgio Delzanno, Angelo Ferrando, Federico Frassetto, Giovanna Guerrini, Viviana Mascardi, Marco Oreggia, Francesca Pozzi*, Alessandro Solimando, and Armando Tacchella

DIBRIS, University of Genoa, Italy – CNR, Italy

Abstract. We discuss a logic-based methodology that we adopted to teach basic concepts of programming to high school students with a scientific profile and very basic knowledge of computer science. For our experiments we combined lectures on inductive reasoning with Prolog, practice on natural language processing and ontologies, and evaluations based on questionnaires before and after the workshop.

1 Introduction

The idea that thinking might be understood as a form of computation, as recently suggested by one of the main experts in knowledge representation and reasoning in artificial intelligence [11], is extremely fascinating. In his book, H. Levesque shows how to support students to make the connection between thinking and computing by learning to write computer programs in Prolog for a variety of tasks that require thought, including understanding natural language.

Taking inspiration from recent experiments during a workshop for high school students organized by the "Guidance, Promotion and Tutoring Committee" of the Computer Science Degrees at the University of Genova[1], we present a proposal for a condensed course for uninitiated aimed at introducing basic and advanced programming concepts using declarative languages like Prolog, following Levesque's "thinking as computation" metaphor. Our work is inspired to seminal proposals by Kowalski [7,8,9] and to more recent works such as the "Prolog programming: a do-it-yourself course for beginners" by Kristina Striegnitz[2].

Although Prolog as a programming language for novices has been heavily criticized in the past because of the misconceptions it may generate [13], many resources for teaching Prolog to beginners can be found on the web. Among them, we can mention implementations like Visual Prolog for Tyros[3], Strawberry

* Dr. Francesca Pozzi is affiliated with CNR; all the other authors are affiliated with DIBRIS.

[1] `http://informatica.dibris.unige.it/`.

[2] `http://cs.union.edu/~striegnk/courses/esslli04prolog/`.

[3] `http://www.visual-prolog.com/download/73/books/tyros/tyros73.pdf`.

Prolog[4], Pretty Prolog[5], the book Learn Prolog Now! [3], also available online[6], as well as many tutorials. This abundance of teaching material witnesses that the Prolog community is extremely lively, and convinced us that it was worth teaching Prolog to students with no programming skills. This decision was also motivated by previous attempts with Scratch and its spin-offs Byob and SNAP![7], that – albeit suitable for allowing beginners to write an almost complex program in a few hours – were perceived as not enough professional.

The course requires about 12 hours and it is thought as a crash course for high school students with different profiles. Its template can be instantiated in many different ways, provided that no previous programming skills are assumed. Our experience with the course was carried out during a workshop involving 39 high school students over three days. The course has a leitmotif — which, in our case, was ontology-driven sentiment analysis — and it is organized in modules as follows.

- **Module 1**: a preliminary lecture on a general topic in computer science and artificial intelligence that will provide the main running example for the class and laboratory sessions (1 hour).
- **Module 2**: an introduction to basic concepts of natural language, logic, knowledge representation and programming with the pure fragment of Prolog (2 hours) followed by an introduction to inductive definitions and basic data structures such as lists (2 hours).
- **Module 3**: an introduction to existing tools supporting exercises related to the main application discussed in the first module (1-2 hour).
- **Module 4**: a practical session with Prolog aimed at developing an application related to the main topic of the course (5-6 hours).

In addition to a description of the teaching activities above, in this paper we address also the problem of evaluating the results of the workshop. In particular, we try to assess whether questionnaires compiled at the beginning and at the end of the workshop can help us in evaluating the impact of the event on the students.

2 Introduction to the Course Leitmotif

The first module of the course introduces the course leitmotif, i.e., ontology-driven sentiment analysis.

Sentiment analysis, also known as opinion mining, is a linguistic analysis technique where a body of text is examined to characterize the tonality of the document[8]. It was first introduced by Pang, Lee and Vaithyanathan in 2002

[4] `http://www.dobrev.com/`.
[5] `https://code.google.com/p/prettyprolog/`.
[6] `http://learnprolognow.org/lpnpage.php?pageid=top`.
[7] `https://snap.berkeley.edu/`.
[8] Definition from The Financial Times Lexicon, `http://lexicon.ft.com/Term?term=sentiment-analysis`.

[17]; a survey by two of these three authors dating back to 2008 [14] defines opinion mining and sentiment analysis as areas dealing with the computational treatment of opinion, sentiment, and subjectivity in text. Given the growth of user-generated contents, sentiment analysis is useful in social media monitoring to automatically characterize the overall feeling or mood of groups of people, e.g., how consumers feel with respect to a specific brand or company. To make an example, Thomson Reuters uses sentiment analysis in a number of its different products to provide traders with information about how companies are faring in news articles.

In the context of computer science, an ontology is a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents [5]. One of the first papers describing the idea of using an ontology to drive the classification of texts by providing lexical variations and synonyms of terms that could be met in the documents was [15]. The domain of interest was that of online product reviews. Among other examples of ontology-driven sentiment analysis we may mention [4,1,6,10].

Our initial lecture was entitled "Ontologies and text classification: two relevant elements to program thinking machines". It first introduced the notion of ontologies and their application, then mentioned the semantic web [2], and finally introduced the general problem of classifying texts and its application to sentiment analysis. As the lecture was only 1 hour long, the topics were just touched upon. A more technical presentation of the steps to be faced in order to detect the polarity of a textual document was given as part of the practical session with Prolog discussed in Section 5.

3 Introduction to Inductive Reasoning with Prolog

The goal of the second module of the course is to introduce the main concepts of the computational paradigm underlying Prolog. To simplify the task, it is convenient to consider the pure fragment of Prolog without use of complex predicates tied to the evaluation strategies of the interpreter, e.g., "cut". The lectures are divided in four parts, discussed in detail in the rest of the Section.

The first part introduces the key ingredients of the Prolog language, i.e., the distinction between a program, a query, and the interpreter. The program is viewed here as the representation of some knowledge, and thus we only considered Datalog-like programs containing predicates and constants. Free variables were introduced together with queries. We started with simple examples of assertions in natural language like: *Mia is a person*, *Jody is a person*, *Yolanda is a person*, *Jody plays guitar*. The above mentioned assertions were represented then via the facts:

```
person(mia).  person(jody).  person(yolanda).  plays(jody,guitar).
```

The example was mainly used to (*i*) explain the difference between a predicate and a constant, and (*ii*) show how unary predicates can be used to describe concepts, constants to describe instances, and binary predicates to describe relations

between instances of concepts. Before introducing variables and clauses, we gave an intuition on how the Prolog environment works by using simple queries like `?- person(mia).` We showed examples of successful queries, failures and errors to emphasize the differences between them. Noticeably, students could immediately see the parallel between the Prolog framework and classical examples of human-computer interaction thanks to the yes/no dialog of the interpreter. We explained here that the intelligence of the machine is programmed by writing an appropriate knowledge base, i.e., the programmer's task is to give intelligence to the machine.

In the second part we introduced increasingly complex queries to pave the way towards clauses and inductive definitions. We started from simple queries such as `?- person(X)`. showing also how to get multiple answers using the ';' command. We then moved to other examples based on binary predicates such as `?- plays(jody,X), ?- plays(X,Y)`. These examples were used to describe the interplay between constants and variables, and to introduce (informally) the notion of matching and unification. The next important concept to introduce was that of conjunctive queries. For this purpose, we used examples such as `?- person(jody) , plays(jody,guitar)`, `?- person(X) , plays(X,guitar)`. We explained the logical interpretation of comma as conjunction, and interpreted the shared variable X as a communication channel between two subqueries: the first one instantiates it, the second one validates the instantiation. We introduced then the notion of clause as a way to define implicit knowledge, i.e., to derive new facts without need of enumerating all of them. The first examples were used to illustrate the syntax of a clause:

```
happy(jim).
hasMp3(jim).
listens2Music(jim) :- happy(jim), hasMp3(jim).
```

We explained the semantics using the following assertions: *Jim is happy. Jim has an MP3 player. If Jim is happy and has got an MP3 player, he listens to music.* The use of ground rules, which allows only one specific inference, is quickly abandoned by introducing free variables to show how to infer several new facts with a single clause:

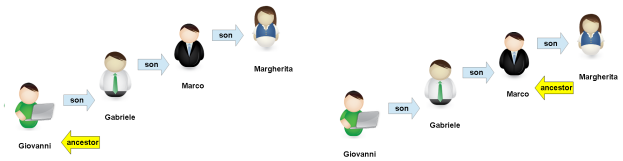
```
guitarist(X) :- plays(X,guitar).
```

The third part was dedicated to inductive definitions. We started from inductive definitions not involving data structures. The classical example was the program defining the transitive closure of a given relation. We used here the relation `son` defined as follows.

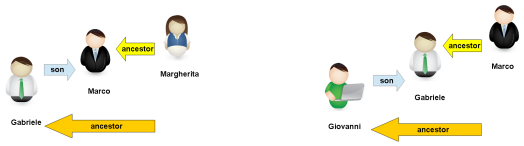
```
son(giovanni,gabriele).  son(gabriele,marco).  son(marco,margherita).
```

We first juxtaposed the ancestor relation over the different definitions of the son relation. We then showed how to combine the son and ancestor relations in order to deduce all other cases. In the explanation we avoided inductive definitions on ancestor predicates only. The following pictures give a diagrammatic

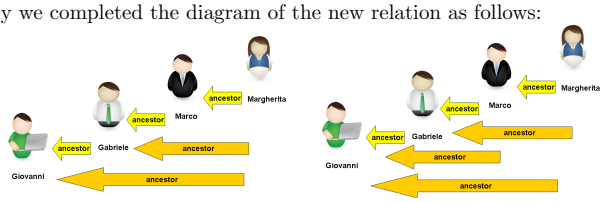
presentation of the inductive definition (introduced step by step in the lecture). We started from the redefining the facts of the `son` relation by using `ancestor`.



The ancestor relation was introduced step by step using the transitive closure:



Finally we completed the diagram of the new relation as follows:



The use of inductive steps defined by combining `son` and `ancestor` (instead of applying the transitive closure on `ancestor` only) worked well and did not cause ambiguity in the explanations. In this way, we immediately focused the attention on the standard way to avoid non terminating recursive definitions in Prolog. The intuition behind the base step of the inductive definition was first expressed using assertions in natural language like: *For every X,Y, if X is a son of Y, then Y is an ancestor of X.* The assertion was then formalized as the Prolog clause

```
ancestor(Y,X) :- son(X,Y).
```

Again we stressed the fact that clauses with free variables can be used to define new facts on top of existing ones without need of enumerating all of them, i.e., clauses define implicit knowledge. Similarly, the inductive step was first expressed using assertions in natural language like: *For every X,Y, if X is a son of Y and Z is an ancestor of Y, then Z is an ancestor of X.* The assertion was then formalized via the Prolog clause

```
ancestor(Z,X) :- son(X,Y) , ancestor(Z,Y).
```

We concluded the example by showing the possible result of a query.

The fourth and last part of the lectures was dedicated to simple data structures like lists. We first introduced the syntax and the intuition behind the data

structures. To explain how to manipulate lists in Prolog, we started by defining a predicate to check if a list contains names of persons only. To explain how the definition works, we consider a procedural interpretation of a recursive definition (consume elements until the list becomes empty). We used the parallel with the transitive closure example to split the definition in base and inductive steps, respectively. However we listed the inductive step before the base step to emphasize the idea that to process a list we first have to consume their elements, and then define what happens when the list becomes empty. We then moved to more complicated examples like `member`. To define the member predicate we first used the assertion in natural language for the base step: *The name X is in the list head. If the name X is in the list L, then X occurs in the list extended with Y different from X*. We presented then the clauses:

```
member(X,[X|L]).
member(X,[Y|L]):-X=/=Y,member(X,L).
```

We used a similar approach to introduce other operations like notin (X is not in a list L), add (add X to a list), and add* (add X if it is not already in L). We also introduced examples of nested lists. Finally, we used (nested) lists to represent and manipulate syntax trees of simple natural language sentences. Specifically, we considered the syntax tree of a text formed by a list of sentences. A sentence was represented as a list of words as specified by the grammar of the considered language, e.g., noun-verb-object We then defined examples of tokens like

```
noun(mia).  noun (jody).  verb(plays).  object(guitar).  object(drums).
```

The assertion *if S is a noun, V is a verb, O is an object, then S V O is a sentence* was modeled via the clause

```
sentence([S,V,O]):-noun(S),verb(V),object(O).
```

This allowed us to give an inductive definition of a text as follows.

```
text([]).
text([F|T]) :- sentence(F) , text(T).
```

Finally, we put all together and showed examples of derivations of queries like `?- text ([ [mia,plays,drums], [jody,plays,guitar] ])`. Again, we used the metaphor of traversal with consumption element-by-element to manipulate nested lists as in the case of simple lists.

4 Practical Session with an Ontology Editor

The third module is meant to introduce the tools useful for the specific course domain — in our case, ontology-driven sentiment analysis. Depending on the domain and related tools, this module may require different amount of time. In our instantiation of the course, we opted for a domain that allowed us to introduce intuitive and user-friendly tools, that high-school students could use

with as little training as possible. In particular, the tool session was organized as a practical laboratory session aimed at introducing the Protégé Ontology Editor. The main reasons for our choice is that the course teachers already had a background on this tool, and that the university students helping during the session had seen Protégé during the "Intelligent Systems and Machine Learning" Master's course. A second reason is that the Web Protégé version[9] can be used online without requiring any installation.

Because of hard time constraints we could not make an introductory lecture on Protégé, so the students just learned by doing during the practical sessions. Some students observed that it would have been useful to introduce the tool beforehand, so we plan to find at least half an hour for explaining the tool and the proposed exercises in the next course editions. The exercises we proposed are based on the Newspaper Example available from the Protégé Frames User's Guide[10]. The Newspaper ontology associated with that guide was made available to the students in their temporary home for easier use.

Exercise 1. The first exercise aimed at making the students acquainted with Protégé by exploring the already made `newspaper` ontology. The text was the following:

1. Open the Protégé ontology editor and select the Newspaper ontology. The ontology domain is that of managing the costs and organization of a newspaper. The ontology can answer the following questions:
   - Who is responsible for each section of the newspaper?
   - What is the content of each article in a section, and who is its author?
   - Who does each author report to?
   - What is the layout and cost of each section?
2. Explore the ontology and experiment with addition and removal of new classes, instances, properties.

Exercise 2. The second exercise was related to the course leitmotiv and asked to design and implement a simple ontology for the opinion mining in the hotel reviews domain, using names in English. The students could save their ontology and were informed that they could have used it in the next module. Students were suggested to identify the positive terms that they could expect in a positive review (for example charming, excellent, polite, clean, ...), the negative ones (dirty, bad, unsafe, ...) and the neutral ones. Then they were suggested to organize them in an ontology having three main branches, one for positive, one for negative and one for neutral words in this domain. To take inspiration for the words, we suggested to read some real reviews available for example on TripAdvisor.

[9] `http://webprotege.stanford.edu`.
[10] `http://protegewiki.stanford.edu/wiki/PrF_UG`.

Exercise 3. In case some students still had time, we proposed to create and run some queries on the `newspaper` ontology used in the first exercise, such as

- Find the journal that contains the article "Destination Mars" and save the query;
- Find the journals that either contain the article "Destination Mars" or have less than 100 pages and save the query.

5 Practical Session with Prolog

The last module of the course integrates all the competencies gained in the previous modules and proposes exercises with increasing complexity, aimed at developing a simple but working application for ontology-driven sentiment analysis. It uses pieces of Prolog code developed by the teachers, offering predicates that make use of external libraries.

A short lecture introduces the goal of the pratical Prolog session by means of an example. Given an ontology like the one depicted in Figure 1 and a review like

> *This hotel is beautiful! It is in a great location, easy to walk anywhere around the city. Very nice, comfortable room with lovely views. Staff can speak English. Fantastic breakfast with many different types of foods available. I would stay here again in a heartbeat.*

we asked the students how could we manage to obtain a classification like

```
6,[review]        6,[positive,review]        1,[nice]
1,[lovely]        1,[great]                  1,[comfortable]
1,[beautiful]     1,[available]
```

Fig. 1. A basic ontology for sentiment analysis in the hotel review domain (only the negative branch is shown for space constraints).

During this short lecture we emphasized that, in order to reach our goal, we had to fix the language of both the text and the ontology (we agreed on English) and we needed

- a tokenizer for transforming a text into the list of its elements (words, punctuation), in order to operate on lists and not directly on text;
- a list of English stopwords to be filtered out before processing the text, as they do not contribute to the text's semantics;
- a stemmer for removing most common morphological and inflectional endings from English words in order to normalize the terms in both the ontology and the text, to make their matching possible;
- a tool for reading an OWL ontology from file and transforming it into some format easy to manipulate.

Exercises 1, 2. The first practical exercise did not depend on the domain. In particular, it was taken from the SWISH web site `http://swish.swi-prolog.org/example/movies.pl` and it is based on querying and extending a movie database. The second exercise asked the students to implement the predicate for removing a ground item from a ground list.

Exercise 3. The third exercise, whilst still being a classical one for Prolog beginners, started to move towards the actual problem to be solved. We asked to implement a `subtractList` predicate for subtracting a list from another one, but we contextualized the problem supposing to have a list of words that represents all the words that are found in a review, and a list of stopwords and punctuation elements. The goal is to remove the stopwords from the list of words retrieved from a text.

We made available to the students the following material:

- The file `stopwords.txt` containing 430 English stopwords and punctuation marks.
- The `emotions.owl` ontology sketched in Figure 1. Since the students had already completed the laboratory with Protégé, we told them that they could use their own ontology instead of the provided one.
- A `textclassifier.pl` Prolog piece of code offering all the solutions to the exercises, but implemented as predicates with different names. We asked the students to refrain from reading this file thoroughly — as they would have found the solved exercises —, but just to consult it to find implementation of auxiliary predicates. The text classifier used the following SWI Prolog libraries:
  - The RDF database (`library(semweb/rdf_db)`)[11] for reading an ontology and transforming it into a set of Prolog facts.
  - The Porter Stem (`library(porter_stem)`)[12] implementing the Porter stemmer [16].

[11] `http://www.swi-prolog.org/pldoc/man?section=semweb-rdf-db`.
[12] `http://www.swi-prolog.org/pldoc/man?section=porter-stem`.

– A set of reviews of hotels in Genova, downloaded from the Booking.com site[13].

We provided the following suggestions to verify that the code was correct:

1. Consult the `textclassifier.pl` file which contains the Prolog code to implement a basic ontology-driven text classifier. The code offers many useful predicates to make your work easier.
2. Use the predicate `fromTextToList(FileName, List)` that takes as its first argument the name of a file and unifies the second argument with the list of the words found in the file: this predicate will allow you to read the contents of a file, be it a review or the `stopwords.txt` file, and turn it into a list that Prolog can manage.
3. After having called the predicate you implemented for subtracting a list from another one, use the predicate `printList(List)` to print the result.

We also provided an example of goals to call and the expected output:

```
?- [textclassifier].
?- fromTextToList('./review5.txt', Review5List),
   fromTextToList('./stopwords.txt', StopWordsList),
   subtractList(ListReview5, StopWordsList, Review5WithoutSWList),
   printList(Review5WithoutSWList).

extremely comfortable welcoming excellent service conveniently situated
railway station main sights palazzo reale best breakfast buffet
experienced week visit italy adjacent restaurant tralalero good
```

The file `review5.txt` contained

> *Extremely comfortable, welcoming, with excellent service and conveniently situated for the railway station and most of the main sights, such as Palazzo Reale. By far the best breakfast buffet I experienced during a two-week visit to Italy! Its adjacent restaurant, Tralalero, is also very good.*

Exercise 4. The fourth exercise asked to implement the `listStem(LWords, LStem)` predicate for obtaining the list of word stems, from the list of original words. The students could use the auxiliary predicate `extendedPorterStem(X, Y)` offered by `textclassifier.pl` to obtain the stemmed word of `X` and unify it with `Y`.

Exercise 5. With the fifth exercise, we started to practice with an ad-hoc Prolog representation of ontologies. `textclassifier.pl` implements a `loadOntology` predicate which, given the OWL ontology file name, asserts information on the ontology classes and subclass relationships. The asserted fact also provides information on the concept name and its stem. For example, by calling `loadOntology('./emotions.owl')`, the following facts are asserted into the Prolog Knowledge Base:

[13] `http://www.booking.com/reviews/it/hotel/bristol-palace.en-gb.html`.

```
class(http://www.owl-ontologies.com/o.owl#amazing,[amazing],[amaz])
class(http://www.owl-ontologies.com/o.owl#available,[available],[avail])
class(http://www.owl-ontologies.com/o.owl#bad,[bad],[bad])
class(http://www.owl-ontologies.com/o.owl#beautiful,[beautiful],[beauti])
class(http://www.owl-ontologies.com/o.owl#best,[best],[best])
class(http://www.owl-ontologies.com/o.owl#charming,[charming],[charm])
class(http://www.owl-ontologies.com/o.owl#positive_review,
                                        [positive,review], [posit,review])
........
subClass(http://www.owl-ontologies.com/o.owl#amazing,
           http://www.owl-ontologies.com/o.owl#positive_review)
subClass(http://www.owl-ontologies.com/o.owl#available,
           http://www.owl-ontologies.com/o.owl#positive_review)
subClass(http://www.owl-ontologies.com/o.owl#bad,
           http://www.owl-ontologies.com/o.owl#negative_review)
subClass(http://www.owl-ontologies.com/o.owl#beautiful,
           http://www.owl-ontologies.com/o.owl#positive_review)
```

The exercise asked the students to load `emotions.owl` and query the knowledge base in order to answer the following question:

1. Which are the words and stems associated with the ontology class `http://www.owl-ontologies.com/o.owl#amazing`?
2. Which ontology concept has the word list `[tasty]` associated with?
3. Which ontology concept has the stem list `[unavail]` associated with?
4. Which ontology concept has the stem list that contains `posit`?
5. Which is the direct superclass of the class whose word list is `[lovely]`?
6. Which are *all* the superclasses of the class whose stem list is `[excel]`?

We also asked the students to make experiments with the ontology they implemented in the Tools Session.

Exercise 6. The last exercise was the most challenging one, as it asked to put all the bricks implemented so far together, in order to implement an ontology-driven text classifier. We gave very limited written hints and we proposed a few variants of the text classification program for listing not only the words occurring in both the text and the ontology (after being stemmed), but also in counting their occurrences: if "beautiful" occurs twice, it should be counted twice and contribute to a more positive evaluation. Another variation we proposed was to use the subClassOf semantic relations present in the ontology to classify the text not only based on the words that coincide with classes in the ontology, but also with their superclasses.

Exercise 7. Since one implementation of the text classifier was available, we concluded this Prolog practical session by asking the students that could not complete Exercise 6, to experiment with our own implementation in order to see a program at work.

Discussion. Although we did not carefully trace the results achieved by the students during their practical experience, we can say that about 10-12 students out of 39 were able to complete the 6th exercise. This result was extremely surprising for all the instructors. In fact, having taught Prolog for many years to students with a solid background on imperative and object-oriented programming, we were aware of the difficulties that students meet when moving from the logic programming theory to the practice and we did not expect that some students could face and complete Exercise 6.

Without claiming to make a scientifically founded assertion, our feeling was that the lack of skills in imperative programming of the high school students, made them open to "think directly in Prolog", without trying to design imperative algorithms and then fit them into logical rules. We plan to take more precise statistics on the completion of the practical exercises in the next editions of the workshop, and maybe propose this course also to students with a computer science background, in order to confirm these feelings.

6 Evaluation

Our evaluation is targeted to understand — in a quantitative way — whether the students were able to improve their computational thinking capabilities after the workshop.

To this purpose, data were collected the first and the last day of the workshop via a questionnaire, with some overlap w.r.t. the general one. The aim was to collect information about the profile of the students and to check their ability to solve easy logic problems. The questionnaire was proposed before and after the experience in order to test whether the experience increased student's knowledge of the subjects.

A First part (profile part) of the specific questionnaire asked students to provide information about their gender, grade, their favorite subject at school (Humanities, Science, Technical subject, Language or Arts), their daily use of computers (from 1 = less than half an hour, to 4 = more than 2 hours), their perceived level of academic achievement (from 1 = excellent to 4 = barely passing), whether they were planning to enroll in a course at the University (yes, maybe, no) and whether they were planning to enroll in a program in Computer Science (yes, maybe, no). In the Second part (questions part) students were asked to answer questions based on logic skills; for this part no specific knowledge is provided or expected. They were recommended to answer only the questions they were able to solve, without trying to answer randomly. Accordingly, the score for every answer was: +1 if right, 0 for blank, and -0.5 if wrong. With this choice of weights, since every question has three alternatives and only one is correct, choosing randomly between the answers yields an expected score of

$$\frac{1}{3} \cdot 1 - \frac{1}{3} \cdot 0.5 - \frac{1}{3} \cdot 0.5 = 0$$

We first tested whether the experience increased student's knowledge of the subjects. The statistical test is performed by using a paired t-test to check the

difference in the results of the Second part before (Time 1) and after (Time 2) the experience. Our results indicate that at Time 2 students are slightly more likely to correctly solve questions, but not sufficiently to have a statistical significance; this emerges even by splitting the population on a subject-base or on a perceived level-base. As a possible explanation, we could consider the relatively small amount of questions provided in the questionnaire, and also to the very condensed format of the training.

C.S.program	Post Cert.	Post Uncert.
Pre Cert.	23	1
Pre Uncert.	7	4

Fig. 2. Interest of enrollment at the Computer Science program: the rows and the columns depict respectively the occurrence or the answers during the test before and after the experience.

Since the experience had the main goal of adequately introducing a Computer Science curriculum, it was also interesting to test whether the experience modified students' intentions to enroll either in the University or in the Computer Science program. The difference between certainty and uncertainty in the intention was tested at Time 1 and Time 2 by performing an exact McNemar test [12]. Regarding the interest to enroll at the University, there was no significant result: only 2 students passed from uncertainty to certainty ($p = 0.47$). The same statistical test was performed on the interest to enroll at the Computer Science program and the result showed that a small proportion of students switched from uncertainty to certainty in their intention to enroll (see Fig. 2) with a $p - value$ of 0.07. Even if this result is higher than our statistical significance threshold (0.05), the fact that the p-value is close to it suggests that the experience might have helped them make up their mind. Despite the limitations of this study (i.e., relatively low sample size, small number of questions, lack of observations by external observers) we can conjecture that with an improved and enlarged set of questions, significant results can be achieved to better guide teachers in their choice of course contents.

7 Conclusions

In this paper we have discussed a format for a tutorial about the basic concepts underlying the computational thinking paradigm using a declarative language like Prolog. The tutorial is centered around a specific application domain. In this paper we discussed the domain of Natural Language Processing, Semantic Web and Ontologies, three hot topics in Computer Science and Artificial Intelligence with several important real-life applications. The domain was adopted as main example in a workshop for high school students taught at the University of Genoa.

The tutorial is based on a crash-course that introduces the application domain, followed by lectures on declarative programming, on the use of tools, and practical sessions to learn the basics of programming, and a final project in which the students apply their new programming skills to a concrete problem. Declarative languages are used here to introduce complex concepts like recursive and inductive definitions with the help of natural language assertions. Ontologies and natural language processing are particularly useful to give an application-oriented flavor to the workshop.

References

1. Matteo Baldoni, Cristina Baroglio, Viviana Patti, and Paolo Rena. From tags to emotions: Ontology-driven sentiment analysis in the social semantic web. *Intelligenza Artificiale*, 6(1):41–54, 2012.
2. Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, pages 29–37, May 2001.
3. Patrick Blackburn, Johan Bos, and Kristina Striegnitz. *Learn Prolog Now!*, volume 7 of *Texts in Computing*. College Publications, 2006.
4. Marcirio Chaves and Cássia Trojahn. Towards a multilingual ontology for ontology-driven content mining in social web sites. In *Proc. of ISWC 2010, Volume I*, 2010.
5. Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.
6. Efstratios Kontopoulos, Christos Berberidis, Theologos Dergiades, and Nick Bassiliades. Ontology-based sentiment analysis of twitter posts. *Expert Syst. Appl.*, 40(10):4065–4074, 2013.
7. Robert A. Kowalski. Logic as a computer language for children. In *ECAI*, pages 2–10, 1982.
8. Robert A. Kowalski. Logic as a computer language for children. In *New Horizons in Educational Computing, (ed. M. Yazdani), Ellis Horwood Ltd., Chichester*, pages 121–144, 1984.
9. Robert A. Kowalski. A proposal for an undergraduate degree in the uses of logic. In *Artificial Intelligence in Higher Education, CEPES-UNESCO International Symposium, Prague, CSFR, October 23-25, 1989, Proceedings*, pages 94–97, 1989.
10. Maurizio Leotta, Silvio Beux, Viviana Mascardi, and Daniela Briola. My MOoD, a multimedia and multilingual ontology driven MAS: Design and first experiments in the sentiment analysis domain. In Cristina Bosco, Erik Cambria, Rossana Damiano, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 2nd Workshop on Emotion and Sentiment in Social and Expressive Media (ESSEM) 2015, a satellite workshop of AAMAS 2015*, 2015.
11. Hector J. Levesque. *Thinking as Computation*. The MIT Press, 2012.
12. Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, June 1947.
13. Patrick Mendelsohn, T.R.G. Green, and Paul Brna. Programming languages in education: The search for an easy start. In J.-M. Hoc, T. R. G. Green, R. Samurçay, and D. J. Gilmore, editors, *Psychology of Programming*, pages 175–200. London, Academic Press, 1990.
14. Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

15. Jantima Polpinij and Aditya K. Ghose. An ontology-based sentiment classification methodology for online consumer reviews. In *Proc. of IEEE/WIC/ACM WI-IAT'08*, pages 518–524, 2008.
16. Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
17. Peter D. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proc. of ACL 2002*, pages 417–424, 2002.

A case study on graph-based planning for emergency evacuation

Santa Agreste[1], Pasquale De Meo[2], Massimo Marchi[3], Maria Francesca Milazzo[4], Salvatore Nunnari, Alessandro Provetti[1]

[1] DMI, University of Messina, Italy
[2] DICAM, University of Messina, Italy
[3] Network services, University of Milan, Italy
[4] DIECII, University of Messina, Italy

Abstract. We present a pilot study on the implementation of a software, based on declarative knowledge representation and logic-based automated planning, which assists the management of severe Chemical hazard events, e.g. fire or emissions that may require the evacuation of the area surrounding the affected Chemical plant. We model the geography, the road network and the population of the chemical plant and the surroundings by weighted, labeled graphs, which are updated as the hazardous situation develops. Intervening factors, e.g. the spread of toxic in the air, are represented in the graph in terms of their effects. Risk for the resident population and possible evacuation plans are evaluated and re-evaluated as the accident develops. Also evacuation plans are contingent and as conditions change may be re-evaluated from scratch; moreover, they may involve complex coordinated actions among the rescue units. Both the evaluation of the emergency scenario and the evacuation planning phases have been prototyped by means of an Answer Set Programming planner.

Keywords: Automated Planning, Applied Computational Logic, Chemical Plants Safety

1 Introduction

We present a pilot study on the implementation of the central component of a safety tool for risk evaluation and evacuation planning to be deployed in response to fire or emissions due to accidents at a large Chemical plant[5].

We have created an abstract model of the geography of the chemical plant and of its surroundings by a weighted, labeled graphs, which is updated as the hazardous situation unfolds. Evacuation from the surroundings of the plant is evaluated and planned wrt. to the graph representation. Intervening factors, e.g. the spread of toxic in the air, are represented in the graph in terms of their effects. Evacuation plans are contingent and as conditions change may be re-evaluated

[5] Due to legal reasons, at this stage we must omit the name of the plant and of the residential area for which the tool has been developed.

from scratch. Both the computation of the likely effects and the planning phase have been prototyped by an Answer Set Programming planner.

Answer Set Programming (ASP) [5] [9] (also called Stable Logic Programming (SLP) [10]), is a relatively recent but well-established style of logic programming: each solution to a problem is represented by an answer set (also called stable model), and not by answer substitutions produced in response to a query. A rich literature exists on applications of ASP in many areas, including problem solving, configuration, information integration, security analysis, agent systems, semantic web, and planning (see, among many, [2, 1, 6, 13, 4] and the references therein).

ASP is the language by which we represent all types of knowledge required to address this scenario: declarative knowledge about the surroundings, procedural knowledge about actions (escape actions, take cover actions and so on), contingencies, and planning as a domain-independent strategy to solve a motion problem. In this sense, our approach is in the same vein as the pioneer work of Zepeda and Sol [14] on evacuation as an instance of logic-based automated planning; the first complete (and delivered) instance of this approach in their *Plan Popocatpetl* project. We believe that our solution represents a marked improvement and generalization of their approach for the following main reason: we have adopted an intermediate formal representations with labeled graph for the representation of the scenario and the a priori evaluation of risk, for the declarative specification of the planning and observing part. These two intermediate representations make the modularization of the underlying ASP code possible and manageable thus enabling the higher degree of adaptivity that is required by the problem.

2 Evacuation plans

The drafting of evacuation plans in emergency requires finding a sequence of actions that lead from an initial state, representing a risk scenario, to a goal objective, representing a situation where the entire population is rescued (or generally safe). Unlike the planning scenarios that are traditionally considered in the Artificial Intelligence literature, evacuation plans specify administrative/security policies which can be hard to formulate[6]—even informally and often hard to execute even in small scenarios, i.e., those where the number of subjects, the spacial dimension and the time-scale are reduced.

Another important difference is the value to give to the 'do nothing' action. While in AI planning the so-called *nop* action is there mostly for padding fixed-length plans, in our applicative scenario they have a precise meaning which must be re-evaluated constantly: in case of chemical hazard, staying inside the building and limiting air circulation could be safest option available. Therefore we can say that automated planning with AI techniques, which is the subject of this

[6] See the norms regulating Save & rescue in Italy from `http://www.protezionecivile.gov.it/`

paper, is only one dimension of the inherent complexity of emergency evacuation management.

The other key element to the formalization of evacuation plans is the representation of the area, with a dynamic description of elements such as i) source and type of hazard ii) risk diffusion maps, which are specific to the type of risk, i.e., iii) number and localization of the population that needs to be evacuated iv) transport means and their level of mobilization v) Accident & Emergency (A & E) services with trained personnel and specialized equipment. For know risks, normally associated to Chemical/energy plants, the complexity of the task is essentially decreased by the availability of pre-compiled maps, which can specify the following two types.

First, during the emergency the danger areas extend (or contract) following the evolution of the accident, moreover such expansion/contraction is not easily characterized by simple circumferences around the site of the accident (consider, e.g., liquid chemicals in rivers, or fire under constant-direction winds). Normally, risk diffusion maps create a three-level partition of the areas in *i) impact,* i.e., areas close to the epicenter of the disaster, with high likelihood of lethality, *ii) damage,* normally external to the former, where lack of protection would cause irreversible damage to those who are contaminated, especially children and old people, and *iii) attention,* where damage is possible but not irreversible, in any case requiring medical treatment and possibly causing unrest in the population.

Second, so-called *safe areas* and their features. These areas are further detailed in i) waiting areas, ii) concentration areas for the rescuers and iii) recovery areas, which are safe places where the refugees will end up as a result of the evacuation.

3 Representation of the geography and of the escape scenarios

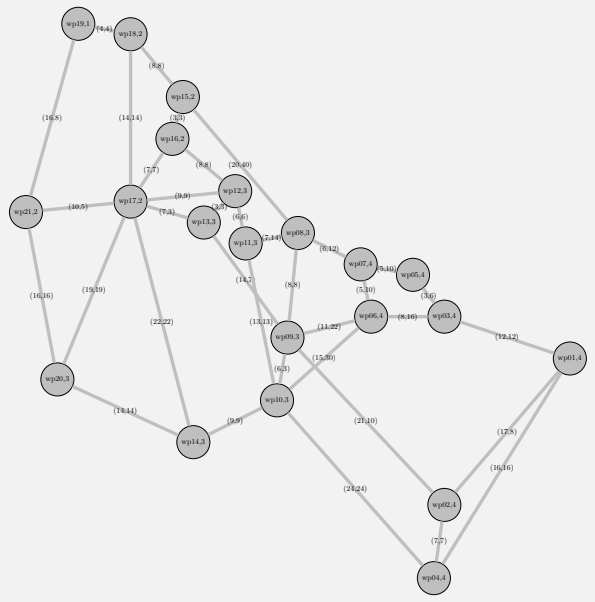
Two key aspects of the knowledge representation needed for this planning instance are the representation of the geography, namely roads and rivers, and of the level of risk assigned to areas by the domain experts. These information are synthesized by the risk graph, which is reported in Figure 1 for the first instance of problem we considered.

For comparison, we report in Figure 1 one of the annotated maps, in the standard format for Geographical Information Systems (GIS) that have been used to compile the graph in Figure 1. The twenty-one relevant area (called *waypoints)* identified by the domain experts (in this case, Fire patrol senior officers) are connected by 36 relevant routes.

As it can be noticed in Figure 1, domain experts have assigned each waypoints to one of 5 levels of risk for the population, according to the following standard risk scale.

- $RiskLevel = 1$: recovery area, destination for evacuation plans;
- $RiskLevel = 2$: low-risk area, close to recovery areas and far from the risk areas;

Fig. 1. The graph representing the geography and the risk levels of designed areas



- $RiskLevel = 3$: average-risk area, far from recovery areas but sufficiently close to the risk areas;
- $RiskLevel = 4$: high-risk area, close to high-risk areas, thus very far from recovery areas;
- $RiskLevel = 5$: high-risk area, the starting point for evacuation plans.

3.1 Description in ASP

This subsection describes the ASP predicate definitions that have been developed to capture the specific aspects of the evacuation planner. The following description does not cover the general part of the evacuation planner, which has been adopted as is from the ASP translation of the action description languages $\mathcal{L}_\prime$ and $\mathcal{L}_\infty$ developed in [3] and extensively described thereof. The only changes were made to embed the general rules into an answer-set program ready for interpretation by the ASP solver DLV [8, 7]; the syntax of the ASP program is thus specific to that accepted by the DLV grounder.

It should be added that some simplifying assumptions have been embodied directly in the ASP representation of the planning instance. These assumptions

Fig. 2. Annotated GIS information on the area subject to evacuation planning

are taken from the pre-compiled evacuation plan now in use, e.g., that for all evacuation actions there are some vehicles, typically buses, available to transport people to recovery areas. Another simplification is in the consideration of groups of evacuees, as opposed to single individuals. The structure of the graph in Figure 1 is embedded into the ASP program by means of the waypoint relation:

$$waypoint(Name, RiskLevel). \tag{1}$$

where variables *Name* and *RiskLevel* indicate the vertex of the graph and its assigned risk level. Communication routes, i.e., the edges of the graph are described by facts of this type:

$$route(Place1, Place2, Length, Criticality). \tag{2}$$

where variables *Place1* and *Place2* indicate the two areas that are connected, *Length* captures geographical distance and *Criticality* is a parameter representing the danger assigned to the usage of the given connection. Another important information is the representation of the evacuee groups:

$$group(Name). \tag{3}$$

where *Name* is assigned to thus-formed groups. The cardinality is not specified but as noted above we assume than one vehicle can evacuate a group. The last type of extensional predicate is for representing the position of the groups on the map, by this type of facts:

$$holds(position(Group, Place), 0). \tag{4}$$

Notice how relation $position(Group, Place)$ is reified into a fluent; variables *Group* and *Place* have their obvious meaning, whereas time-stamp 0 relates these facts to the initial state of the planning activity. Of course, we can have more than one group sitting on the same waiting area, as well as empty waiting areas. Finally, to describe actions where a certain areas become unreachable, i.e., a communication route has become nonviable (e.g., busy or disrupted or dangerous), we use these types of fact:

$$\begin{array}{l} waypoint\_blocked(WP1). \\ route\_blocked(WP1, WP2). \end{array} \tag{5}$$

The predicate described above are to be added to the domain description and changed often, to adapt to the changing scenario, especially the (possible) disruption of roads, to be acquired, in the full version of this planner, from real-time GIS information.

4 Results and open issues

One of the most important problems to be solved in case of disasters is the draw and quick deployment of evacuation plans for the population. We describe a methodology based on knowledge representation and reasoning to formulate Evacuation Plans, using the intermediate graph representation and the DLV inferential engine.

Our evacuation planner considers the present situation, the type of danger, weather conditions, traffic or other modification of the zone to be evacuated, and formulates alternative evacuation plans to be face-validated on a case-by-case basis.

Studying the real case of the External Emergency Plan for a Refinery, we have implemented a planner able to generate appropriate evacuation plans, on the basis also of incomplete information derived from a possible Geographic Information System, supplying a representation of the scenario on the ground.

The results against a benchmark of 5 realistic emergency scenarios are encouraging: computation times remain within few minutes and the generated solutions were rated "excellent" by domain experts. From the point of view of computational logic, these results are entirely satisfactory, and, in our opinion, should become even more significant and widely applicable by the introduction of two further formal devices. The first device is the formal apparatus of ASP programs with weak constraints developed by Leone et al. and implemented in DLV. Even though there have been successful applications in literature, e.g. [12], at the moment, our tests indicate that weak constraints are too heavy computationally to be deployed in our platform, so we have decided to leave them out of the current implementation.

The second improvement would be a full model of context to be applied to data, i.e., to redesign the data as to capture their contextual aspects, and have the devices, e.g., local-cell emergency broadcasting, to selectively handle them, along the lines of the methodology defined Rauseo et al. [11].

References

1. Anger, C., Schaub, T., Truszczyński, M.: ASPARAGUS – the Dagstuhl Initiative. ALP Newsletter 17(3) (2004), see http://asparagus.cs.uni-potsdam.de
2. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press (2003)
3. Baral, C., Gelfond, M., Provetti, A.: Representing actions: Laws, observations and hypotheses. Journal of Logic Programming 31(1-3), 201–243 (1997), http://dx.doi.org/10.1016/S0743-1066(96)00141-0
4. Gelfond, M.: Answer sets. In: Handbook of Knowledge Representation, chapter 7. Elsevier (2007)
5. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP. pp. 1070–1080 (1988)
6. Leone, N.: Logic programming and nonmonotonic reasoning: From theory to systems and applications. In: Baral, C., Brewka, G., Schlipf, J.S. (eds.) Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007. p. 1 (2007)
7. Leone, N., Faber, W.: The dlv project: A tour from theory and research to applications and market. In: de la Banda, M.G., Pontelli, E. (eds.) ICLP. Lecture Notes in Computer Science, vol. 5366, pp. 53–68. Springer (2008)
8. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Logic 7(3), 499–562 (Jul 2006), http://doi.acm.org/10.1145/1149114.1149117
9. Lifschitz, V.: Answer set planning. In: Schreye, D.D. (ed.) Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999. pp. 23–37. MIT Press (1999)
10. Marek, V.W., Truszczyński, M.: Stable logic programming - an alternative logic programming paradigm, pp. 375–398. Springer (1999)
11. Rauseo, A., Martinenghi, D., Tanca, L.: Context through answer set programming. In: Fletcher, G.H.L., Staworko, S. (eds.) Proceedings of the 4th International Workshop on Logic in Databases, Uppsala, Sweden, (EDBT/ICDT '10 joint conference), March 25, 2011, Proceedings. p. 58. ACM (2011), http://doi.acm.org/10.1145/1966357.1966369
12. Rauseo, A., Martinenghi, D., Tanca, L.: Contextual data tailoring using ASP. In: Schewe, K., Thalheim, B. (eds.) Semantics in Data and Knowledge Bases, 5th International Workshop, SDKB 2011, Zürich, Switzerland, July 3, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7693, pp. 99–117. Springer (2011), http://dx.doi.org/10.1007/978-3-642-36008-4_5
13. Truszczyński, M.: Logic programming for knowledge representation. In: Dahl, V., Niemelä, I. (eds.) Logic Programming, 23rd International Conference, ICLP 2007. pp. 76–88 (2007)
14. Zepeda, C., Sol, D.: Evacuation planning using answer set programming: An initial approach. Engineering Letters 15(2), 240–249 (2007)

How Answer Set Programming Can Help In Digital Forensic Investigation

Stefania Costantini[1] stefania.costantini@univaq.it,
Giovanni De Gasperis[1] giovanni.degasperis@univaq.it, and
Raffaele Olivieri[1,2] raffaele.olivieri@gmail.com

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica,
Universitá degli Studi dell'Aquila,
Via Vetoio 1, 67100 L'Aquila, Italy
[2] Raggruppamento Carabinieri Investigazioni Scientifiche (Ra.C.I.S.),
(The Italian Department of Scientific Investigations of Carabinieri),
viale di Tor di Quinto 119, 00191 Rome, Italy.

Abstract. The results of the evidence analysis phase in Digital Forensics (DF) provide objective data which however require further elaboration by the investigators, that have to contextualize analysis results within an investigative environment so as to provide possible hypotheses that can be proposed as proofs in court, to be evaluated by lawyers and judges. Aim of our research has been that of exploring the applicability of Answer Set Programming (ASP) to the automatization of evidence analysis. This offers many advantages, among which that of making different possible investigative hypotheses explicit, while otherwise different human experts often devise and select different solutions in an implicit way. Moreover, ASP provides a potential for verifiability which is crucial in such an application field. Very complex investigations for which human experts can hardly find solutions turn out in fact to be reducible to optimization problems in classes P or NP or not far beyond, that can be thus expressed in ASP. As a proof of concept, in this paper we present the formulation of some real investigative cases via simple ASP programs, and discuss how this leads to the formulation of concrete investigative hypotheses.

1 Introduzione

Digital Forensics (DF) is a branch of criminalistics which deals with the identification, acquisition, preservation, analysis and presentation of the information content of computer systems, or in general of digital devices [1, 2]. The aim is to identify digital sources of proofs, and to organize such proofs in order to make them robust in view of their discussion in court, either in civil or penal trials. Digital forensics is concerned with the analysis of potential elements of proof after a crime has been committed ("post-mortem"). Clearly, the development of digital forensics is highly related to the development of Information and communication technologies in the last decades, and to the widespread diffusion of electronic devices and infrastructures. It involves various disciplines such as computer science, electronic engineering, various branches of law, investigation techniques and criminological sciences. Rough evidence must be however used

to elicit hypotheses concerning events, actions and facts (or sequences of them) with the goal to present them in court. Evidence analysis involves examining fragmented incomplete knowledge, and defining complex scenarios by aggregation, likely involving time, uncertainty, causality, and alternative possibilities. No single methodology exists today for digital evidence analysis. The scientific investigation experts usually work by means of their experience and intuition (expertise).

In fact, evidence acquisition is supported by a number of hardware and software tools, both closed and open source. These tools are continuously evolving to follow the evolution of the involved technologies and devices. Instead, evidence analysis, is much less supported. In evidence analysis, the technicians and experts perform the following tasks. (i) collect, categorize and revise the evidence items retrieved from electronic devices. (ii) examine them so as to hypothesize the possible existence of a crime and potential crime perpetrators. (iii) elicit from the evidence possible proofs that support the hypotheses. (iv) organize and present the proofs in a form which is acceptable by the involved parties, namely lawyers and judges, which may include to exhibit explicit supporting arguments. Figure 1[3] shows some real sequences of the technical activities involved. Few software tools exist that only cover some partial aspects, and all of them

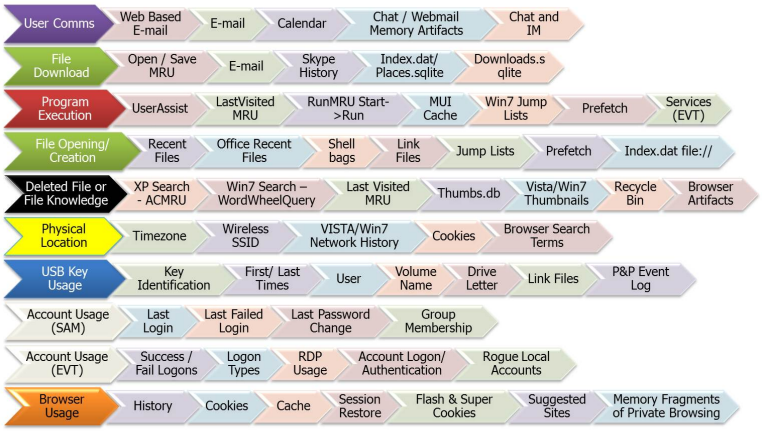


Fig. 1. Example of sequences of technical activities

are "black box" tools, i.e., they provide results without motivation or explanation, and without any possibility of verification. Thus, such results can hardly be presented as reliable proofs to the involved parties. Moreover, the absence of decision support sys-

[3] Image by courtesy from SANS web site url https://blogs.sans.org

tems leads to undesirable uncertainty about the outcome of evidence analysis. Often, different technicians analyzing the same case reach different conclusions, and this may determine different judge's decisions in court.

Formal and verifiable artificial intelligence and automated reasoning methods and techniques for evidence analysis would be very useful for the elicitation of sources of proof. Several aspects should to be taken into account such as timing of events and actions, possible (causal) correlations, context in which suspicious actions occurred, skills of the involved suspects, validity of alibis, etc. Moreover, given available evidence, different possible underlying scenarios may exist, that should be identified, examined and evaluated. All the above should be performed via "white box" techniques, meaning that such techniques should be verifiable with respect to the results they provide, how such results are generated, and how the results can be explained. The new wished-for software tools should be reliable and provide a high level of assurance, in the sense of confidence in the system's correct behavior. computational logic is a suitable candidate to definition and implementation of such tools, and non-monotonic reasoning is clearly extensively required.

The long-term objective of this research is to provide law enforcement, investigators, intelligence agencies, criminologists, public prosecutors, lawyers and judges with decision-support-systems that can effectively aid them in their activities. The adoption of such systems can contribute to making how to proceed clearer and faster, and also under some respects more reliable. In fact, the choice of computational logic as a basis guarantees transparency and verifiability of tools and results. The aim of the present paper is to provide a proof-of-concept of the applicability of computational logic and non-monotonic reasoning to such tasks. In order to convince the several parties involved, whatever limited their computer science expertise might be, we have considered a series of fragments of cases and have transposed them into simple self-explanatory answer set programs which provide results which are easy to understand. However, we have considered fragments of real cases which are presently being investigated by the Italian Department of Scientific Investigations of Carabinieri [4].

We have adopted Answer Set Programming (ASP, cf., among others, [3–9]) because ASP programs are declarative and readable and because, as shown in the following sections, as a matter of fact several analysis problems can be nicely reduced to optimization problems for which ASP is particularly well-suited. In fact, the above picture shows how the analysis phase consists in an ordered sequence of detailed technical activity, performed on the seized memories (or on their forensic copy) following hard rules and formal procedures, through tools and/or forensic equipment, to research specific elements, perform the verification of a state or condition, etc. The outcome of the analysis is a detailed and well-motivated technical report. However these reports , although providing a comprehensive response in technical terms, may be insufficient or even not directly usable from the investigation point of view. Further elaboration is in general needed in order to contextualize technical data in the investigation context. Due to the experience in DF gained by one of the authors of this paper as a member officer of a DF laboratory of an italian police force, with national jurisdiction, it has been possible to identify and experimentally treat (fragments of) complex investigations by reduction to

[4] the Police branch of the Italian Army `http://www.carabinieri.it`

answer set programming. This paper presents the results of these experiments. Results are indeed very promising, as the reduction of investigation cases to answer set programming has allowed the experts to identify new investigative hypotheses, that have been practically exploited.

The paper is organized as follows: in Section 2 we provide a short introduction to ASP; in Sections 3- 5 we present three simple though representative examples; finally, in Section 6 we conclude.

2 Answer Set Programming (ASP) in a Nutshell

"Answer Set Programming" (ASP) is a well-established logic programming paradigm adopting logic programs with default negation under the *answer set semantics*, which [3, 4] is a view of logic programs as sets of inference rules (more precisely, default inference rules). In fact, one can see an answer set program as a set of constraints on the solution of a problem, where each answer set represents a solution compatible with the constraints expressed by the program. The reader may refer to [3–9], among others, for a presentation of ASP as a tool for declarative problem-solving.

Syntactically, a program Π is a collection of *rules* of the form:

$$H \leftarrow \ L_1, \ldots, L_m, \ not \ L_{m+1}, \ldots, not \ L_{m+n}$$

where H is an atom, $m \geq 0$ and $n \geq 0$, and each L_i is an atom. Symbol $\leftarrow$ is usually indicated with :- in practical systems. An atom L_i and its negative counterpart $not\, L_i$ are called *literals*. The left-hand side and the right-hand side of the rule are called *head* and *body*, respectively. A rule with empty body is called a *fact*. A rule with empty head is a *constraint*, where a constraint of the form $\leftarrow L_1, ..., L_n.$ states that literals $L_1, \ldots, L_n$ cannot be simultaneously true in any answer set.

A program may have several answer sets, each of which represent a solution to given problem which is consistent w.r.t. the problem description and constraints. If a program has no answer set, this means that no such solution can be found and the program is said to be *inconsistent* (as opposed to *consistent*).

In practical terms a problem encoding, in the form of ASP program, is processed by an ASP solver which computes the answer set(s) of the program, from which the solutions can be easily extracted (by abstracting away from irrelevant details). Several well-developed answer set solvers [10] that compute the answer sets of a given program can be freely downloaded by potential users. All solvers provide a number of additional features useful for practical programming, that we will introduce only whenever needed. Solvers are periodically checked and compared over well-established benchmarks, and over challenging sample applications proposed at the yearly ASP competition (cf. [11] for a recent report).

The expressive power of ASP, as well as, its computational complexity have been deeply investigated [12]. Precisely, in the propositional case the problem of deciding whether a given program admits answer sets is NP-complete, and so is the problem of deciding whether there is an answer set containing a specific atom (while deciding whether a specific atom is in all the answer sets is Co-NP-complete). ASP is clearly able to express NP-complete problems.

3 Case 1: Data Recovery and File Sharing Hypotheses

3.1 The Investigative Case

The judicial authority requested the digital forensics laboratory to analyze the contents of an hard disk, in order to check for the presence of illegal contents files. If so, they requested to check for potential activities of sharing on Internet of illegal materials. The hard disk under analysis was physically damaged (as often done by criminals if they suspect capture). Therefore, after a head replacement, the evidence acquistion phase recovered a large amount of files (of various types: images, videos, documents, etc.), however without their original name. This because the damage present on the disk plates disallowed the recovery the information of the MFT[5]. For this reason, an arbitrary name has been assigned to all the files recovered. Information about the original name of files and their original location in the file system is thus missing.

3.2 Elements

By analyzing the recovered files, technicians detected the occurrence of:

- files with illegal contents;
- various "INDX files", corresponding in the NTFS file system to directory files, which contains the follow METADATA:
  - filename;
  - physical and logical size of the file;
  - created, accessed, modified and changed timestamps;
- index related to the eMule (which is a widely-used file-exchange application), including a file containing sharing statistics, whose original name is "known.met".

Starting from the elements described above, we have been able to reply to the judicial authority's question with: a reasonably reliable hypothesis of association of the recovered file to the respective original name; a reasonable certainty that illegal files were actually exchanged on the Internet. This has been obtained by modeling the given problem by means of a very simple well-known ASP example, reported below.

3.3 The Marriage Problem

The Marriage Problem (or SMP - Stable Marriage Problem) is a well-known NP-hard optimization problem which finds a stable matching between two sets of elements S_1 and S_2 (say **men** and **women**) given a set of preferences for each element. A matching is a mapping from the elements of one set to the elements of the other set which thus creates a set of couples (A, B) where $A \in S_1$ and $B \in S_2$. A matching is stable whenever it is not the case that some element $\hat{A}$ of the first matched set prefers some given element $\hat{B}$ of the second matched set over the element to which $\hat{A}$ is already matched, and the same holds for $\hat{B}$.

[5] Master File Table: structured block table containing the attributes of all files in the volume of an NTFS file system.

3.4 Reduction

The given problem is in fact reducible to SMP as follows. In the real case, the lists have been created as follow:

- **men** list: defined as the list of names extracted from directory files “INDX files”;
- **women**: defined as the list of recovered files with have been provisionally assigned arbitrary names.

The **preferences** list (or relation order) between the **men** and **women** lists is derived from the comparison of the properties of the individual recovered files (file type, size, etc.) with those identified in file ‘INDX files”.

3.5 Answer Set Programming Solution

Once compiled the lists **men**, **women** and **preferences**, you can search for answer sets by means of the following ASP program (in the syntax of the *smodels* solver). Facts in the program correspond to a real (though very small) example.

```
preference(f001, flower_jpg).
preference(f001, woman_jpg).
preference(f002, flower_jpg).
preference(f002, child_jpg).
preference(f003, child_jpg).
preference(f003, woman_jpg).

bigamy(X,Y) :- preference(X,Y), preference(X,Y1), couple(X,Y), couple(X,Y1), Y!=Y1.
bigamy(X,Y) :- preference(X,Y), preference(X1,Y), couple(X1,Y), X!=X1.
couple(X,Y) :-  preference(X,Y), not bigamy(X,Y).

#hide.
#show couple(X,Y).
```

3.6 Results

The results obtained with the *smodels* solver on the real example are the follow:

```
smodels version 2.26.
Answer: 1
Stable Model: couple(f002,child_jpg) couple(f001,woman_jpg) couple(f001,flower_jpg)
Answer: 2
Stable Model: couple(f003,child_jpg) couple(f001,woman_jpg) couple(f001,flower_jpg)
Answer: 3
Stable Model: couple(f003,child_jpg) couple(f002,flower_jpg) couple(f001,woman_jpg)
Answer: 4
Stable Model: couple(f002,child_jpg) couple(f002,flower_jpg) couple(f001,woman_jpg)
Answer: 5
Stable Model: couple(f003,woman_jpg) couple(f002,child_jpg) couple(f002,flower_jpg)
Answer: 6
Stable Model: couple(f003,woman_jpg) couple(f002,child_jpg) couple(f001,flower_jpg)
Answer: 7
Stable Model: couple(f003,woman_jpg) couple(f003,child_jpg) couple(f001,flower_jpg)
Answer: 8
Stable Model: couple(f003,woman_jpg) couple(f003,child_jpg) couple(f002,flower_jpg)
```

From the answer sets, it is possible (as the reader can see) to formulate hypotheses about the original names of the recovered files. Furthermore, by comparing the file

names indexed in the file *Known.met*[6], it has been possible to make reasonable assumptions about the effective sharing of files with illegal content.

4 Case 2: Path Verification

4.1 The Investigative Case

After a heinous crime, an allegedly suspect has been arrested. The police sequestered all his mobile devices (smartphone, route navigator, tablet, etc...). The judicial authority requested the DF laboratory to analyze the digital contents of the mobile devices in order to determine their position with respect to the crime site during an interval of time which includes the estimated time when the crime was perpetrated.

4.2 Elements

From the analysis of the mobile devices, a set of geographical GPS coordinates have been extracted, some of them related to the the time interval under investigation. There are however some gaps, one of them certainly due to a proven switch off of few minutes around the crime time. To start with, a list called GPS-LIST is generated, collecting all the positions extracted from the various devices, grouped and ordered by *time unit* of interest (seconds, multiple of seconds, minutes, etc..). The objective is that of establishing whether the known GPS coordinates are compatible with some path which locates the given mobile devices at the crime site during the given time interval. If no such path exists, then the suspect must be discharged. If some compatible path is found, then the investigation about the potential perpetrator can proceed. The objective has been reached via reduction to the following simple game.

4.3 Hidato Puzzle (Hidoku)

Hidato is a logical puzzle (also known as "Hidoku") invented by the Israeli mathematician Dr. Gyora Benedek. The aim of Hidato is to fill a matrix of numbers, partially filled a priori, using consecutive numbers connected over a horizontal, vertical or diagonal ideal line. Below we show, as a simple example, a 6x6 initial matrix.

18	0	0	0	26	0
19	0	0	27	0	0
0	14	0	0	23	31
1	0	0	8	33	0
0	0	5	0	0	0
0	0	10	0	36	35

[6] As mentioned, known.met is a file of the widely-used *eMule* file-exchange application that stores the statistics of all files that the software shared, all files present in the download list and downloaded in the past.

4.4 Reduction

It has been possible to perform the reduction of the given investigation problem to the "Hidato Puzzle" problem, by creating a matrix representing the geographical area of interest, where each element of the matrix represents a physical zone crossable in a unit of time. The physical size of the individual cell of the matrix (grid) on the map will be proportionate to the time unit that will be considered, both the hypothetical transfer speed. The matrix has been populated with the elements of the previously-created LIST-GPS, i.e., with known positions of the suspect.

Considering the above matrix, assume that the crime has been committed at location 34, at a time included in the interval with lower bound corresponding to when the suspect was at location 1 and upper bound corresponding to when the suspect was at location 36. All devices have been provably switched off between locations 5 and 10.

4.5 Answer Set Programming Solution

Once built the matrix, we can determine whether a suspect route exists by finding the answer sets of the following ASP program [13] (here we have used the *clingo* solver). Notice that the omitted cells are assumed to have value 0.

```
#const n = 6.
matrix(1, 1, 18).  matrix(1, 5, 26).  matrix(2, 1, 19).  matrix(2, 4, 27).
matrix(3, 2, 14).  matrix(3, 5, 23).  matrix(3, 6, 31).  matrix(4, 1, 1).
matrix(4, 4, 8).   matrix(4, 5, 33).  matrix(5, 3, 5).   matrix(6, 3, 10).
matrix(6, 5, 36).  matrix(6, 6, 35).

size(1..n).
values(1..n*n).
values2(1..n*n-1).
diffs(-1;0;1).

1 { x(Row, Col, Value) : values(Value) } 1 :- size(Row), size(Col).
1 { x(Row, Col, Value) : size(Row) : size(Col) } 1 :- values(Value).
x(Row, Col, Value) :- matrix(Row, Col, Value).

valid(Row, Col, Row2, Col2) :- diffs(A), diffs(B), Row2 = Row+A, Col2 = Col+B,
                               Row2 >= 1, Col2 >= 1, Row2 <= size, Col2 <= size,
                               size(Row), size(Col).

:- x(Row, Col, Value+1), x(Row2, Col2, Value),
   not valid(Row, Col, Row2, Col2), values2(Value).
#hide.
#show x(Row, Col, Value).
```

4.6 Results

The results obtained via the *clingo* solver are the following:

```
Answer: 1
x(1,1,18) x(1,5,26) x(2,1,19) x(2,4,27) x(3,2,14) x(3,5,23) x(3,6,31) x(4,1,1)
x(4,4,8) x(4,5,33) x(5,3,5) x(6,3,10) x(6,5,36) x(6,6,35) x(5,1,2) x(6,1,3)
x(6,2,4) x(6,4,6) x(5,5,7) x(5,4,9) x(5,2,11) x(4,2,12) x(3,1,13) x(4,3,15)
x(3,3,16) x(2,3,21) x(3,4,22) x(2,6,24) x(1,6,25) x(1,3,28) x(1,4,29) x(2,5,30)
x(4,6,32) x(5,6,34) x(1,2,20) x(2,2,17)

Answer: 2
x(1,1,18) x(1,5,26) x(2,1,19) x(2,4,27) x(3,2,14) x(3,5,23) x(3,6,31) x(4,1,1)
x(4,4,8) x(4,5,33) x(5,3,5) x(6,3,10) x(6,5,36) x(6,6,35) x(5,1,2) x(6,1,3)
x(6,2,4) x(6,4,6) x(5,5,7) x(5,4,9) x(5,2,11) x(4,3,12) x(3,3,13) x(4,2,15)
x(3,1,16) x(2,3,21) x(3,4,22) x(2,6,24) x(1,6,25) x(1,3,28) x(1,4,29) x(2,5,30)
x(4,6,32) x(5,6,34) x(1,2,20) x(2,2,17)
```

These results are particularly interesting for the investigation, as they both correspond to paths which are compatible with the hypothesis of the suspect committing the crime.

18	20	28	29	26	25
19	17	21	27	30	24
13	14	16	22	23	31
1	12	15	8	33	32
2	11	5	9	7	34
3	4	10	6	**36**	35

18	20	28	29	26	25
19	17	21	27	30	24
16	14	13	22	23	31
1	15	12	8	33	32
2	11	5	9	7	34
3	4	10	6	**36**	35

It should be noted that a variant of the Hidato algorithm exists, that considers maps whose structure is more complex than a rectangular matrix.

5 Case 3: Alibi Verification

5.1 The Investigative Case

During an investigation concerning a bloody murder, it is necessary check the alibi provided by a suspect. In the questioning, the suspect has been rather vague about the timing of his movements. However, he declared what follows.

- to have left home (place X) at a certain time;
- to have reached the office at place Y where he worked on the computer for a certain time;
- to have subsequently reached place Z where, soon after opening the entrance door, he discovered the body and raised the alarm.

In order to verify the suspect's alibi, the judicial authority requested the DF laboratory to analyze:

- the contents of the smartphone owned by the suspect;
- the computer confiscated in place Y, where the suspect says to have worked;
- a video-surveillance equipment installed at a post office situated near place Z, as its video-camera surveys the street that provides access to Z.

5.2 Elements

The coroner's analysis on the body has established the temporal interval including the time of death. From the forensic analysis of the smarphone it has been possible to compile a list of GPS positions related to a time interval including the time of death, denoted by GPS-LIST. The analysis of the computer allowed the experts to extract the list of accesses on the day of the crime, denoted by LOGON-LIST. The analysis of the video-surveillance equipment allowed the experts to isolate some sequences, denoted by VIDEO-LIST, that show a male subject whose somatic features are compatible with the suspect. All the above lists have been ordered according to the temporal sequence of their elements. The investigation case at hand can be modeled as a planning problem where time is a fundamental element in order to establish whether a sequence of

actions exist that may allow to reach a certain objective within a certain time. Several approaches to causal and temporal reasoning in ASP exist, that could be usefully exploited for this kind of problem[7]. Here, for lack of space and for the sake of simplicity we model the problem by means of the very famous "Monkey & Banana" problem, which is the archetype of such kind of problems in artificial intelligence.

5.3 Monkey & Banana

The specification of "Monkey & Banana" is the following: A monkey is in a room. Suspended from the ceiling is a banana, beyond the monkey's reach. In the room there is also a chair (in some versions there is a stick, that we do not consider). The ceiling is just the right height so that a monkey standing on a chair could knock the banana down (in the more general version by using the stick, in our version just by hand). The monkey knows how to move around, carry other things around, reach for the banana. What is the best sequence of actions for the monkey? The initial conditions are that: the chair is not just below the bananas, rather it is in a different location in the room; the monkey is in a different location with respect to the chair and the bananas.

5.4 Reduction

The reduction of the case at hand to the 'Monkey & Banana" problem is the following. Notice the reduction of the "idle" state of the monkey to unknown actions that the suspect may have performed at that time.

Monkey	$\rightarrow$	Suspect
Banana	$\rightarrow$	Body
Eats Banana	$\rightarrow$	Raise Alarm
Initial Position Monkey	$\rightarrow$	X
Initial Position Chair	$\rightarrow$	Y
Below Banana	$\rightarrow$	Z
Walks	$\rightarrow$	Walks
Move Chair	$\rightarrow$	Motion to Z
Ascend	$\rightarrow$	Open the Door
Idle	$\rightarrow$	Unknown Action

Problem's constraint are that, at any time, the monkey:

- may perform only one action at each time instant among walk, move chair, stand on chair, or stay idle;
- if the monkey stands on the chair, it cannot walk, and it cannot climb further;
- if the chair is not moved then it stays where it is, and vice versa if it is moved it changes its position;
- the monkey is somewhere in the room, where it remains unless it walks, which implies changing position;

[7] For lack of space we cannot provide the pertinent bibliography: please refer to [14] and to the references therein.

– the monkey may climb or move the chair only if it is in the chair's location;
– the monkey can reach the banana only if it has climbed the chair, and the chair is under the banana.

5.5 Answer Set Programming Solution

The following ASP program, obtained by modifying a version that can be found at http://www.dbai.tuwien.ac.at/proj/dlv/tutorial/, is formulated for the DLV solver, and provides in the answer sets the timed sequences of actions (if any exists) by which the monkey can reach and eat the banana.

```
walk(Time) v move_chair(Time) v ascend(Time) v idle(Time) v eats_banana(Time) :-
                                                                    #int(Time).
monkey_motion(T) :- walk(T).
monkey_motion(T) :- move_chair(T).

stands_on_chair(T2) :- ascend(T), T2 = T + 1.
:- stands_on_chair(T), ascend(T).
:- stands_on_chair(T), monkey_motion(T).
stands_on_chair(T2) :- stands_on_chair(T), T2 = T + 1.

chair_at_place(X, T2) :-
    chair_at_place(X, T1), T2 = T1 + 1,not move_chair(T1).
chair_at_place(Pos, T2) :-
    move_chair(T1),T2 = T1 + 1,monkey_at_place(Pos, T2).
:- move_chair(T1),chair_at_place(Pos,T2),chair_at_place(Pos1,T1),T2 = T1+1, Pos=Pos1.

monkey_at_place(monkey_starting_point, T) v
monkey_at_place(chair_starting_point, T) v
monkey_at_place(below_banana, T) :- #int(T).

:- monkey_at_place(chair_starting_point, 0).
:- monkey_at_place(below_banana, 0).
:- not monkey_at_place(monkey_starting_point, 0).

:- monkey_at_place(Pos1, T2),
   monkey_at_place(Pos2, T1), T2 = T1 + 1,
   Pos1 != Pos2, not monkey_motion(T1).
:- monkey_at_place(Pos, T2), monkey_at_place(Pos, T1),
   T2 = T1 + 1, monkey_motion(T1).
:- ascend(T),monkey_at_place(Pos1, T),
   chair_at_place(Pos2, T),Pos1 != Pos2.
:- move_chair(T),monkey_at_place(Pos1, T),
   chair_at_place(Pos2, T),Pos1 != Pos2.

monkey_at_place(monkey_starting_point, 0) :- true.
chair_at_place(chair_starting_point, 0) :- true.

reach_banana(T) :- can_reach_banana(T).
can_reach_banana(T) :- stands_on_chair(T),
        chair_at_place(below_banana, T).
:-eats_banana(T), not can_reach_banana(T).
:- eats_banana(T1),eats_banana(T2), T1!=T2.
happy :- eats_banana(T).
:- not happy.

step(N, walk, Destination) :- walk(N),
         monkey_at_place(Destination, N2),N2 = N + 1.
step(N, move_chair, Destination) :-
         move_chair(N),monkey_at_place(Destination, N2),
         N2 = N + 1.
step(N, ascend, " ") :- ascend(N).
step(N, idle, " ") :- idle(N).
step(N, eats_banana, " ") :- eats_banana(N).
```

5.6 Results

The proposed reduction in the first place allows investigators to verify the alibi provided by the suspect. In fact, the possible timed lists of actions performed by the suspect are determined as answer sets of the above program. Such lists are constructed so as to be compatible with the detected GPS positions of the suspect, the detected computer activity and the actions that the suspect has declared to have performed. By running the solver on the real case with a maximum number of steps N = 3, corresponding to the case where the suspect is provably at the office at time 0, we get exactly the action sequences needed to reach the goal.

```
{step(0,walk,chair_starting_point), step(1,move_chair,below_banana),
 step(2,ascend," "), step(3,eats_banana," ")}
```

Therefore, if the suspect raised the alarm at time 3 he actually had no time for committing the crime and therefore he should presumably be discharged.

In case instead the alibi is not fully verified, then further investigation is needed. By increasing the time, for example to N = 5, we in fact get many sets of possible alternative actions, where *idle* is an unknown action for which it might interesting to investigate further so as to prove or reject the investigation thesis.

```
{step(0,idle," "),step(1,walk,chair_starting_point),
step(2,move_chair,below_banana), step(3,ascend," "),
step(4,idle," "),step(5,eats_banana," ")}

{step(0,walk,below_banana), step(1,walk,chair_starting_point),
step(2,move_chair,below_banana), step(3,ascend," ")
step(4,idle," "), step(5,eats_banana," ") }
```

Among the answer sets there are many which suggest suspicious behavior. The first one above outlines a scenario where the initial suspect's actions are unknown. Then he moves to the crime site where however he has the time and opportunity to commit the crime at step 4. Even worse is the second answer set, where the suspect moves to the crime site, than moves back to the office, moves a second time to the crime site where again he has the time and opportunity to commit the crime at step 4. As the suspect's presence at the crime site is confirmed by the video-surveillance equipment records, this behavior is suggestive of, e.g., going to meet the victim and having a discussion, going back to the office (maybe to get a weapon) and then actually committing the crime.

6 Conclusions

In this paper we have demonstrated the applicability of non-monotonic reasoning techniques to evidence analysis in digital forensics by mapping some fragments of real cases to existing simple answer set programs. The application of artificial intelligence and in particular of non-monotonic reasoning techniques to evidence analysis is a novelty: in fact, even very influential publications in digital forensics such as [1, 2] are basically a guide for human experts about how to better understand and exploit digital data. Therefore the present work, though preliminary, opens significant new perspectives. Future

developments include building a toolkit exploiting not only ASP but also other non-monotonic-reasoning techniques such as abduction, temporal reasoning, causal reasoning and others, as elements of decision-support-systems that can effectively aid investigation activities and support of the production of evidence to be examined in trial. The multidisciplinary future challenge is that of making such tools formally accepted in court proceedings: this involves in general terms complex societal and psychological issues. From the technical point of view, for making such tools acceptable and perceived as reliable, it is crucial to develop verification, certification, assurance and explanation techniques.

References

1. Casey, E.: Handbook of Digital Forensics and Investigation. Elsevier (2009)
2. Casey, E.: Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet. books.google.com (2011)
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proc. of the 5th Intl. Conf. and Symposium on Logic Programming, MIT Press (1988) 1070–1080
4. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
5. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press (2003)
6. Leone, N.: Logic programming and nonmonotonic reasoning: From theory to systems and applications. In: Logic Programming and Nonmonotonic Reasoning, 9th Intl. Conference, LPNMR 2007. (2007)
7. Truszczyński, M.: Logic programming for knowledge representation. In Dahl, V., Niemelä, I., eds.: Logic Programming, 23rd Intl. Conference, ICLP 2007. (2007) 76–88
8. Dovier, A., Formisano, A., Pontelli, E.: An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. Journal of Experimental and Theoretical Artificial Intelligence **21**(2) (2009) 79–121
9. Dovier, A., Formisano, A.: Programmazione Dichiarativa in Prolog, CLP, ASP, e CCP. (2008) Available (in Italian) at `https://users.dimi.uniud.it/~agostino.dovier/DID/corsi.html`.
10. Web references of ASP solvers: Clasp: `potassco.sourceforge.net`; Cmodels: `www.cs.utexas.edu/users/tag/cmodels`; DLV: `www.dbai.tuwien.ac.at/proj/dlv`; Smodels: `www.tcs.hut.fi/Software/smodels`.
11. Calimeri, F., Ianni, G., Krennwallner, T., Ricca, F.: The answer set programming competition. AI Magazine **33**(4) (2012) 114–118
12. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys **33**(3) (2001) 374–425
13. Kjellerstrand, H. Available at `http://www.hakank.org/answer_set_programming` (2015)
14. Cabalar, P.: Causal logic programming. In: Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz. Volume 7265 of Lecture Notes in Computer Science., Springer (2012) 102–116

Leveraging Semantic Web Technologies for Analysis of Crime in Social Science

Luca Pulina[1], Antonietta Mazzette[1], Laura Pandolfo[2], Elena Piga[1], Maria Laura Ruiu[1], and Camillo Tidore[1]

[1] POLCOMING, Università di Sassari, Viale Mancini n. 5 – 07100 Sassari – Italy
{lpulina,mazzette,mlruiu,tidore}@uniss.it
[2] DIBRIS, Università di Genova, Via Opera Pia, 13 – 16145 Genova – Italy
laura.pandolfo@edu.unige.it

Abstract. In this paper we present the conceptual level of an ontology-based application aimed to support social scientists in their sociological analysis related on crime. Starting from several concrete issues posed by the research team of the *Social Observatory on Crime* of the University of Sassari, our goal is to build a Semantic Web based tool to collect, organize, and analyze data on crime, as well as for the exploitation of research results by institutions and civil society.

1 Introduction

Generally speaking, crime analysis is the activity aimed at finding trends in crimes in order to devise both policies and solutions to crime-related issues. From a sociological point of view, the analysis of crime is addressed to understand how criminal phenomena might influence social assets in the context of urban and rural areas. Moreover, it aims to identify prevention measures and their potential effects on social configurations. In order to do that, data analysis plays a role of paramount importance. Nowadays, information on urban crime is profitably used by a growing number of local governments to identify major risks and make decisions about the safety of their community.

Computer-assisted qualitative data analysis software (CAQDAS) were successfully adopted by the sociological research community in order to both organize and analyze such kind of data – see, e.g., [1]. A number of benefits of CAQDAS have been recognized by sociological literature. On the other hand, several issues are still open in relation to theoretical, methodological issues – see, e.g., [2] – and practical aspects, such as heterogeneity of data sources, data integration, and the exploitation of implicit knowledge related to the collected data. It is well-established that the usage of Semantic Web (SW) technologies can provide a valuable support in order to overcome the practical limits listed above. Moreover, the usage of such technologies in crime and public safety fields is not new – see, e.g., [3].

In this paper we present the conceptual level of an ontology-based application aimed at supporting social scientists in their sociological analysis related to

crime. It represents the first step towards the development of a tool aimed at organize and manage both quantitative and qualitative data related to this application domain. In particular, the need of such a tool has emerged from several concrete issues posed by the research team of the *Social Observatory on Crime* (OSC)[3] of the University of Sassari. On the one hand, the creation of a SW-based crime information platform might allow stakeholders – institutions and civil society – to easily access to data on crime, for instance by mapping crimes and crime-related issues, and identifying where and how they are occurring, where they are concentrated and why. On the other hand, it can facilitate researchers' work in organizing and managing data collected from different sources, such as statistical data and qualitative information from newspapers.

The remainder of the paper is organized as follows. In Section 2 we describe the OSC, while in Section 3 we report the whole process of design and implementation of the presented ontology. We conclude the paper in Section 4 with some final remarks and discussing future works.

2 The Social Observatory on Crime

The OSC originated in 2012 thanks to an interdisciplinary team (in particular social scientists such as sociologists, psychologists, economists, jurists) which involves researchers from the University of Sassari. It originated from the Urban Study Center (CSU)[4], that focuses on urban and territory evolution and transformations; coordinates empirical study and promotes the culture of legality by involving students in actively doing research. The CSU contributes to promoting the adoption of governance approaches by involving private and public bodies as both producers and beneficiaries of research outcomes. It also aims to disseminate activities' results through seminars, conferences, educative courses and scientific publications. Since 2004, the CSU has started to focus on crime and insecurity in order to observe their impacts on the Sardinian social context and territory. The analysis and monitoring activities are based on data collected from documents provided by justice officers, newspapers and national statistical reports (e.g., reports of the Italian National Institute of Statistics, ISTAT).

The OSC was built aimed at promoting and generating governance approaches by trying to involve different kinds of stakeholder, such as private and public bodies. In fact, the inclusion of these actors was supposed to be relevant both to collect basic information and data, and to define concerted strategies for reducing criminal behaviors and attitudes. For a long time, both literature and policies have focused on situational crime prevention strategies by creating "defensible spaces", and less focused on the contrast of the motivations that encourage deviance. During the last twenty years governance approaches to crime have been promoted, through collaboration among a multiplicity of actors (also "external" to the control/protection functions). Following this approach, the CSU concentrated its efforts on identifying processes for increasing

[3] *Osservatorio Sociale sulla Criminalità*, `http://polcoming.uniss.it/node/1133`
[4] *Centro Studi Urbani*, `http://www.centrostudiurbani.it`

degree of key-stakeholders' participation and networking rather than defining specific "interventions" and "architectonic fences". Recently [4], OSC identified social indicators in order to create an "Informative System for data collection and analysis". This tool was thought to support policy planning and decision-making oriented at fighting and reducing criminal and illegal activities.

The main goal of OSC is to develop an exhaustive database which includes quantitative data (primary and secondary data from different sources, e.g., ISTAT and regional prosecutors) and qualitative information obtained, e.g., by analyzing local newspapers (in particular *La Nuova Sardegna*[5] and *L'Unione Sarda*[6]). The newspaper consultation has allowed the collection a number of detailed information (e.g., description of places where murders happened, description of authors way of life and their past experiences, connections with other types of crime, etc.) that otherwise would have been difficult to gain and record. However, the consultation refers to those relevant crimes which get newspapers attention such as murders, robberies, attacks, threats, and cannabis cultivation.

A further objective of OSC is to analyze connections between widespread insecurity and crime. In fact, individual and collective behaviors, decision-making and economic activities are often strongly related to the types and the intensity of these connections. Moreover, criminal phenomena should be analyzed in relation to the process of modernization (and its consequences) that has involved the targeted territories by shaping social configuration of urban and rural areas.

3 The OCRA Ontology

OCRA aims at being the conceptual layer of a Semantic Web based tool focused on the improvement of the processes related to the collection, organization, management, and analysis of data on criminal phenomena in Sardinia by OSC. In the following, we describe design and implementation of OCRA (Ontology for CRime Analysis). We can summarize as follows the main steps of this process:

1. Definition of the domain.
2. Identification of the key concepts of the domain to be described.
3. Identification of the proper language and Tbox implementation.
4. Ontology population, i.e., filling the Abox with known facts.

Firstly, we reviewed the semi-structured dataset collected by OSC during a 11 years-long research on criminality in Sardinia. Data on criminal phenomena were collected through specific forms to be filled with information obtained, e.g., by local newspapers such as *La Nuova Sardegna* and *L'Unione Sarda*. The criminal phenomena recorded are murder, attacks, robbery and cultivation of cannabis. Data collection forms were mainly composed of the following information:

– Data concerning the newspaper, e.g., name of the newspaper, date, and title of the related article.

[5] `http://lanuovasardegna.it`
[6] `http://unionesarda.it`

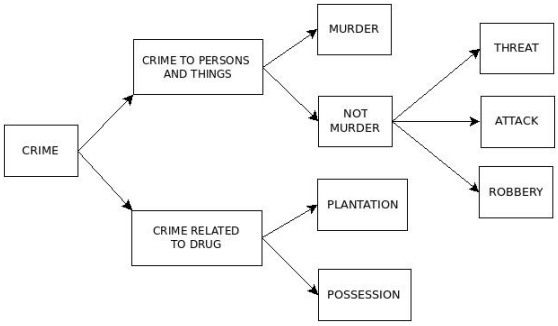


Fig. 1. Raw classification of the criminal phenomena.

- Data concerning the crime, e.g., type, place, date, and motive.
- Data concerning the authors of the crime and victims, e.g., name, job, age, and records of criminal offenses.

Regarding the second point, we analyzed collected data in order to highlight common terminology, redundancies, and relationships between different elements, as suggested by the domain experts of the OSC. The results of this process enabled us to compute a taxonomy – depicted in Figure 1 – related to different crimes.

Considering the third point, we proceeded with the choice of the modeling language analyzing the different alternatives offered by OWL 2. To retain most of the practical advantages of OWL 2, but to improve on its applicability, in [5] has been introduced OWL 2 profiles, i.e., a sub-language of OWL 2 featuring limitations on the available language constructs and their usage.

Considering the available profiles, we excluded OWL 2 EL because it does not support inverse object properties, while we discarded both OWL 2 QL and OWL 2 RL because they do not support, e.g., existential quantification to individuals. Thus, OCRA has been developed in OWL 2 DL, and its Tbox is composed of 81 classes, 36 object properties, and 53 data properties. In the following, we describe main classes of the OCRA ontology[7]:

`ArticoloGiornale` *(NewspaperArticle)* represents the class containing the newspapers information about the specific crime. Every instance of this class has data properties such as `Titolo` *(Title)* and `DataArticolo` *(ArticleDate)*.

`Luogo` *(Place)* includes the place where the crime has occurred. Individuals of `Luogo` are also the places in which the victims and offenders were born or live.

[7] The full documentation is available at `http://visionlab.uniss.it/OCRA`.

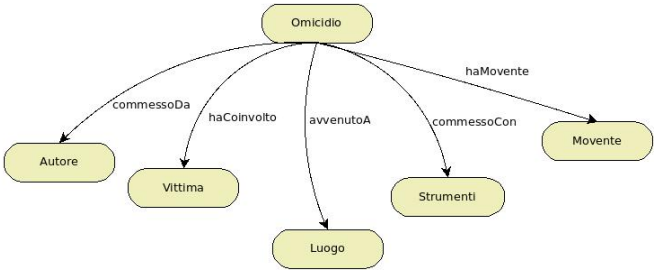


Fig. 2. Class `Omicidio` and related classes in the OCRA ontology.

`Movente` *(Motive)* aims to model the motive of the crime. It has different sub-classes, each of which is a specific motive, such as economical, political, revenge, etc.

`PersonaFisica` *(Person)* models people related to a specific crime. It has two sub-classes, namely `Vittima` *(Victim)* and `Autore` *(Offender)*. Every individual belonging to those classes has data properties such as `Età` *(Age)*, `Sesso` *(Gender)*, `StatoCivile` *(MaritalStatus)*, `Precedenti` *(RecordsOfCriminalOffenses)*.

`Reato` *(Crime)* is one of the central classes of OCRA. It has two sub-classes related to the main types of offenses taken into account: crimes involving people or things and crimes related to drug – see below. `Reato` has different data properties such as `DataReato`*(CrimeDate)*, `NumeroVittime`*(NumberOfVictims)*, `NumeroAutori` *(NumberOfOffenders)*.

`ReatoAPersoneECose` *(CrimeToPersonsAndThings)* In this class are included individuals related to crimes that caused material damage or injure people. `ReatoAPersoneECose` has two sub-classes: `Omicidio`*(Murder)*, which includes crimes with homicide, and `NonOmicidio`*(NotMurder)*. The latter covers a large series of crimes which have not led to murder. `NonOmicidio` has three sub-classes, each one modeling different category offenses, namely `Attentato`*(Threat)*, `Minaccia`*(Attack)*, `Rapina`*(Robbery)*.

`ReatoCollegatoAllaDroga` *(CrimeRelatedToDrug)* is the other principal sub-class of `Reato` and is related to all the drug offences. In particular, we modeled the following two sub-classes of drug offences: `Coltivazione`*(Plantation)* and `Detenzione`*(Possession)*. Some of the most relevant data properties of `ReatoCollegatoAllaDroga` are connected to the type and the number of drugs confiscated by the authorities, such as `Semi` *(Seeds)* and `Piante`*(Plants)*.

`Strumenti` *(Weapons)* represents the class containing the instruments used by an offender to commit the crime. It has various sub-classes, such as `ArmiDaFuoco` *(FireArms)*, `Esplosivi` *(Explosives)* and `Veicoli` *(Vehicles)*.

Concerning object properties, we briefly describe the ones related to `Omicidio`, because they enable domain experts to involve in their analysis important

data regarding places in which the crime has occurred, offenders, and victims. Noticeable object properties are:

- `commessoDa` *(CommittedBy)*: connects `Omicidio` to `Autore`.
- `haCoinvolto` *(hasInvolvedIn)*: relationship between `Omicidio` and `Vittima`.
- `avvenutoA` *(takesPlaceIn)*: allows the identification of murder's place (`Luogo`).
- `commessoCon` *(hasWeapon)*: relationship with the murder weapon.
- `haMovente` *(hasMotive)*: it connects the offense with the motive (`Movente`).

In Figure 2 we show a graphical example of these relationships.

Finally, the OCRA Abox has been populated using data provided by the OSC. Actually, the Abox contains more than 15000 individuals, with their related properties, while the whole ontology is composed of about 365000 triples.

4 Conclusions

In this paper we described design and development of the OCRA ontology, the conceptual level of the ontology-based application aimed to support people of OSC in their sociological analysis related to crime.

Currently, we are developing a data integration layer in order to exploit information coming from relevant external sources, e.g., open data provided by ISTAT and DBpedia. We are also designing a Graphical User Interface to support the ontology population stage, in order to make this process of knowledge acquisition more interactive and dynamic. More, concerning the ontology population, we are studying automated solutions for data collection and insertion.

Finally, we are planning to perform more detailed experimental analysis on the OCRA ontologies. Some preliminary experiments have shown us that OCRA could be a challenging benchmark for OWL 2 DL reasoners.

Acknowledgments The authors wish to thank the anonymous reviewers for their valuable suggestions, which were helpful in improving the final version of the paper.

References

1. Mangabeira, W.C.: Caqdas and its diffusion across four countries: National specificities and common themes. Current Sociology **44**(3) (1996) 191–205
2. John, W.S., Johnson, P.: The pros and cons of data analysis software for qualitative research. Journal of Nursing Scholarship **32**(4) (2000) 393–397
3. Asaro, C., Biasiotti, M.A., Guidotti, P., Papini, M., Sagri, M.T., Tiscornia, D.: A domain ontology: Italian crime ontology. In: Proceedings of the ICAIL 2003 Workshop on Legal Ontologies & Web based legal information management. (2003)
4. Mazzette, A., ed.: La criminalità in Sardegna, Quarto rapporto di ricerca. EDES, Sassari (2014)
5. Motik, B., Patel-Schneider, P., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation **27** (2009)

Parametric Protocol-Driven Agents and their Integration in JADE*

Angelo Ferrando

DIBRIS, Genoa University, Italy
s3479302@studenti.unige.it

Abstract. In this paper we introduce "Template Global Types" which extend Constrained Global Types to support a more generic and modular approach to define protocols, meant as patterns of events of a given type. Protocols can be used both for monitoring the behavior of distributed computational entities and for driving it. In this paper we show the potential of Template Global Types in the domain of protocol-driven intelligent software agents. The interpreter for "executing" Template Global Types has a very natural implementation in Prolog which can easily implement the transition rules for moving from one state to another one, given that an event has been perceived (in case of monitoring) or generated for execution (in case of protocol-driven behavior). This interpreter has been integrated into the Jason logic-based agent framework with limited effort, thanks to the native support that Jason offers to Prolog. In order to demonstrate the flexibility and portability of our approach, which goes beyond the boundaries of logic-based frameworks, in this paper we discuss the integration of the protocol-driven interpreter into the JADE agent framework, entirely implemented in Java.

1 Introduction

Nowadays, having guarantees on the correct behavior of developed systems is gaining more and more importance. Especially in the case of distributed systems, increasing their robustness is a mandatory target. To achieve this goal, we can use two different techniques:

- Testing, which reduces the percentage of bugs in software by trying out the largest number of features offered by the system in order to find errors or inconsistencies.
- Verification, which allows the developers to perform an exhaustive search in order to check if all chosen properties are maintained, such as the absence of deadlock within a concurrent system.

Checking correctness is not an easy problem, especially when considering distributed systems.

* The paper contains original material. The author is a student of the Computer Science Master's Degree at Genova University.

A typical example of complex, heterogeneous, open and dynamic distributed system is the multi-agent system (MAS), where each component is autonomous and communication is vital.

In order to verify the correct behavior of the agents we can monitor them with the help of a monitor agent, as the one by Briola et al. [6]. An evolution of that approach is described by Ancona et al. [11] in which, instead of having one monitor agent that controls the behavior of the system by checking that everything is running correctly, the behavior of each individual agent is driven by the protocol. Since the agent's behavior is driven by a protocol, it is correct without needing to be controlled, or better, the agent in a certain way becomes controller of itself, being able to do only what the protocol allows it to do.

Protocols can be defined using many different formalisms. In the previous work carried out at the University of Genova, an original formalism named "Global Types" was proposed along with its successive extensions including "Constrained Global Types" [1, 2].

Protocols expressed as Constrained Global Types include both Prolog equations and transition rules moving from one allowed protocol state to another one, given a perceived (or a generated, in case of protocol-driven agents) event fully implemented in Prolog. For this reason, the first attempt to implement an interpreter for protocol-driven agents was with Jason[1], because it is able to run Prolog code directly within the agent [11].

This paper advances a previous work [11] in the following issues:

- first extends Constrained Global Types with a mechanism for making them more modular and flexible ("Template Global Types");
- second implements agents driven by protocols expressed using Template Global Types in Jason;
- third implements agents driven by protocols expressed using Template Global Types in a framework not based on Prolog. The choice fell on JADE[2], a Java platform for the creation of multi-agent systems widely used in industrial applications (see Figure 1).

The integration of an interpreter for Template Global Types protocol-driven allows us to demonstrate two important points:

- the choice of the framework for the implementation of the MASs is not a constraint. For instance, agents can follow the BDI (Beliefs, Desires, Intentions [23]) approach, as happens in Jason, or not, as happens in JADE;
- the only requirement is the ability to implement an interface between a Prolog engine supporting cyclic terms and the underlying MAS framework.

[1] `http://jason.sourceforge.net/`
[2] `http://jade.tilab.com/`

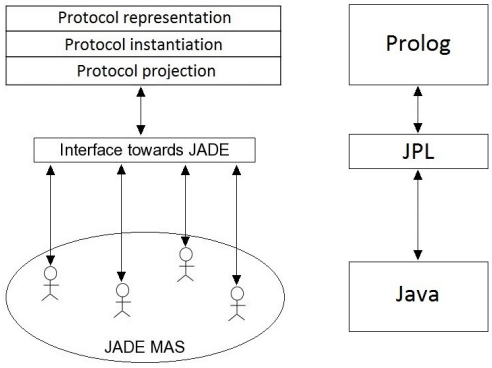


Fig. 1. Our framework for protocol-driven JADE agents.

2 Background

In [11] we used Constrained Global Types without variables (that is instead typical of Attribute Global Type [18]). In order to better understand the content of this paper, we briefly sketch their syntax and semantics.

Omitting the operators that are not used in our examples, a constrained global type may look like the following:

- *EvType:PrSpec* (*sequence*), where *EvType* is an event type and *PrSpec* is a constrained global type. From a semantic point of view, *EvType:PrSpec* represents the set of all event traces whose first event *Ev* matches the event type *EvType* ($Ev \in EvType$), and the remaining part is a trace in the set represented by *PrSpec*.
- $PrSpec_1$ | $PrSpec_2$ (*fork*), where both $PrSpec_1$ and $PrSpec_2$ are constrained global types, representing the set obtained by shuffling the traces in $PrSpec_1$ with the traces in $PrSpec_2$.

With these two operators we can already write complex protocols, as we will see afterwards with some examples.

Another important aspect that we want to introduce, and that was widely discussed in [11], is the *switch* interaction. Such an event can take place when our events include communicative ones. During an interaction protocol execution, an agent can ask to another to change its protocol; this can happen only in specific circumstances and is an important feature in order to support runtime protocol switch.

3 Template Global Types

To be able to write more complex protocols, we extended the previous work [11] introducing Template Global Types. The main difference of Template Global Types w.r.t. Constrained Global Types is the presence of parameters inside the protocol definition.

Template Global Types are a sort of "meta-formalism" in the sense they cannot be directly used as they are. Indeed, they are templates which must be applied to some arguments in order to obtain plain Constrained Global Types. Parameters are present only in the Template Global Type definition: when Template Global Types are actually used either for runtime verification or for protocol driven behavior generation, all terms must be ground, i.e. variables must have already been instantiated.

In order to better explain this new formalism, we introduce some examples.

Here is how we would have represented a client-server protocol with Constrained Global Types[3].

```
                              SERVER =
(receive_request(client1),0):(serve_request(client1),0):SERVER.
```

An example of correct trace would be

```
receive_request(client1): serve_request(client1):
          receive_request(client1):...
```

where ... indicates that the trace is infinite: Constrained Global Types use coinduction to easily represent endless sequences.

The `SERVER` protocol is a client-server protocol made up by a loop in which the server receives a `serve_request` from the client1 replying with a `receive_request`.

Global types can be easily expressed as a set of Prolog equations like the one defining SERVER.

If we would like to change the client we should modify the protocol. This is not very convenient because our protocols support switch interactions and we could take advantage of this feature in order to change, for instance, the clients that communicate with the server.

Using instead Template Global Types we can avoid this problem, defining the protocol as follows:

[3] The reader should ignore the 0 placed as second argument in the event type definition: the number is required to synchronize sequences present in different *fork* branches; in the examples in this paper no synchronization is necessary and the argument is fixed to 0.

```
                                SERVER =
(receive_request(var(1)),0):(serve_request(var(1)),0):SERVER.
```

In this way we have written a generic protocol where we can change the involved agents simply changing the domain of parameter $var(1)$.

The domain of $var(1)$ is set during the application stage. In fact, a Template Global Type must be "applied" in order to turn into a "normal" Constrained Global Type, which can be used as described in our previous work. In particular, after the application stage, the obtained Constrained Global Type can be projected. The projection of a Constrained Global Type is still a Constrained Global Type, where events not involving the agents in the given set are removed.

When we say "project on a set of agents", we mean that, being the protocol a static and global representation of our system, there are some parts of it that are interesting for some agents but not for others. In general, an agent is not present in all the protocol points and it is for this reason that, before allowing an agent A to "execute" a protocol, we project it onto A, in order to remove all events which do not involve A from the protocol instance that A will need to execute.

Let us consider a more complex example:

```
                               SERVER1 =
 (receive_request(client1),0):(serve_request(client1),0):SERVER1,
                               SERVER2 =
 (receive_request(client2),0):(serve_request(client2),0):SERVER2,
                               SERVER3 =
 (receive_request(client3),0):(serve_request(client3),0):SERVER3,
               SERVER = SERVER1|(SERVER2|SERVER3).
```

In this example, we have three protocol branches, each of which is similar to the simple protocol described before, combined using a "fork" operator. The first branch involves `client1`, the second `client2`, the third `client3`. As we can see we have to write the same piece of code many times and above all it is impossible to change agents at runtime, for example during a protocol switch.

Instead, using Template Global Type we may write:

```
                               SERVERT =
  (receive_request(var(1)),0):(serve_request(var(1)),0):SERVERT,
      SERVER = finite_composition(fork, SERVERT, [var(1)]).
```

The construct $finite\_composition$ is used to compose many times a Constrained Global Type with a chosen operator, in this case the fork operator. If we iterate the $var(1)$ parameter on the set containing $\{client1, client2, client3\}$ we get the same results as before, without $var(1)$. The great advantage of this approach, is that the set over which $var(1)$ ranges can be decided at runtime,

hence allowing the agents to implement a (limited) form of **dynamic protocol generation**.

In order to better explain the application and projecting phases, we can see (with a short piece of *pseudocode*) some sample calls:

```
SERVERT =
        (receive_request(var(1)),0):
        (serve_request(var(1)),0):
        SERVERT,
SERVER = finite_composition(fork, SERVERT, [var(1)]),
apply(SERVER,
      [t(var(1), [client1, client2, client3])],
      INSTANTIATEDSERVER),
project(INSTANTIATEDSERVER, [agent1], PROJECTEDSERVER).
```

The *apply* predicate instantiates the `SERVER` protocol returning its instantiation in `INSTANTIATEDSERVER` variable. Afterwards, the obtained "normal" Constrained Global Type without parameters can be projected; in our example, the projection is on an agent called *agent1*.

After the application and projection phases we obtain a "customized" protocol driven agent. This agent will have to only choose what to do during its execution on the basis of what is expected by the protocol; the respect of the global protocol is guaranteed because each agent directly derives from it via projection, and each agent is guided by the same interpreter, that interrogates the Prolog library which implements both the protocol definition and the "apply", "project", and all the other predicates which are necessary in order to know what is expected from protocol and what is not.

The original contribution described in this paper lies in the design of Template Global Types and in the implementation of the apply predicate. The other predicates were already implemented. Also, we integrated the Template Global Types mechanism into both Jason and JADE.

4 Integration inside Jason

Before trying the integration in JADE, we considered a more linear and modular implementation in Jason.

Jason is an interpreter for an extended version of AgentSpeak, where each agent implements the BDI approach; it is a useful framework in order to quickly create MASs architectures using a Prolog-like language.

In case of Jason, the implementation was almost straightforward because it directly supports Prolog code. Hence, it was not necessary to add an interface between Jason and the Prolog engine, since the Prolog library defining apply, project, and all the other predicates necessary for implementing a protocol-driven behavior, could be used by Jason almost "as it is" (with definitely minor syntactic changes).

The integration of our approach into JADE was instead more challenging. We had many different design choices, but in the end we had to opt for the architecture in which most of the work is done in Prolog, and JADE is almost passive. The reason for this choice, better explained later, is that JADE cannot directly manipulate the representation of cyclic protocols, because of limitations of the Java-Prolog interface, hence we had to relegate all the operations on the protocols into the Prolog code, only task of calling Prolog predicates, without taking any decision.

5 Integration inside JADE

JADE is a Java framework where each agent must extend a common Java class (*Agent* class). We created a class called *AgentProtocolDriven* that extends it. Each new agent must extend the latter and override some methods:

- *setup*, method dedicated to initialize the Prolog engine with all predicates necessary to the agent;
- *react*, method dedicated to the agent reaction after a message reception that is expected by the protocol;
- *unexpected*, method dedicated to the agent reaction after a message reception that is unexpected by protocol;
- *select_messages*, method used by the agent in order to select which message to send between those expected by the protocol.

Each agent's setup method must recall the inherited method of its parent (*AgentProtocolDriven* class) in which the main behavior implementing the interpreter's body is created and added. This is a *Cyclic Behavior* which is executed any time the agent is selected by the JADE schedule. It is like a loop, and in each round the agent can check if can do something coherently with the protocol.

The parent's setup method does not only create a behavior but it cares about instantiation and projection of the protocol by calling the Prolog predicate: *instantiate_template_and_project*.

Below we show the Prolog code corresponding to this predicate. It can be easily seen that the code can be broken down into three basic components (as already seen in Figure 1):

- Protocol representation
- Protocol instantiation
- Protocol projection

```
/* Predicate that manage the instantiation
   of a Template Global Type */
instantiate_template_and_project
 (Name, ActualParameters, MyName, ProjectedAgents) :-
```

```
   /* Get the Template Global Type from the library,
      this protocol can have parameter variables */
   trace_expr_template(Name, GlobalType),
   /* Preprocessing phase where all the syntactic sugar
      and parameter variables are removed */
   apply(GlobalType, ActualParameters, InstantiatedTemplate),
   /* Project protocol on this agent */
   project(MyName, InstantiatedTemplate,
           ProjectedAgents, ProjectedGlobalType),
   /* Update current state of protocol */
   clean_and_record(MyName,
                    current_state(ProjectedGlobalType)).
```

The above code is invoked in Java as follows.

```
// Instantiate and project the protocol
new Query(
  "instantiate_template_and_project(" +
  protocolName + "," +
  protocolParameters + "," +
  getLocalName() + "," +
  "[" + getLocalName() + "])"
).hasSolution();
```

After this sequence of instructions, inside the Prolog engine the projected protocol of our agent is correctly instantiated and the interpreter can follow it.

To know the actions allowed by the protocol at a given time (namely, which messages the agent can send, which one it is allowed to receive), the agent queries the Prolog library where all the important pieces of information, like the current state of protocol, are maintained; to do this, in JADE, we have to use a Java library called JPL, that makes communication between a Java program and the SWI Prolog engine possible. So, the JADE agents interpreter can, step by step, ask to Prolog what the agent can or cannot do.

To receiving or sending a message is explicitly set a priori with other parameters, all by reading a configuration file.

5.1 Message reception

When a message is received by an agent, it is put in a queue. When a message is selected from the queue, in order to check if it is expected by the protocol in the current state we have created a predicate in Prolog that returns a list containing all messages that agent can receive.

```
/* Messages that agent can receive according to
   current state of the protocol */
inMsgs(MyName, ListToReceive) :-
 /* Current state of the agent protocol */
 recorded(MyName, current_state(LastState), _),
 /* Find all possible next state
```

```
    where agent is the receiver */
 findall(
  msg(SenderV, MyName, PerformativeV, ContentV, NewStateV),
  next(
   0, LastState,
   msg(SenderV, MyName, PerformativeV, ContentV), NewStateV,
   0, MyName),
  ListToReceive).

/* If Msg is allow in this state of protocol
   move to new state and save it */
move_to_next(MyName, Msg) :-
 /* Current state of the agent protocol */
 recorded(MyName, current_state(LastState), Ref),
 /* Try to do a step,
    if it is valid in the current state */
 next(0, LastState, Msg, NewState, 0, MyName),
 erase(Ref),
 /* Update current state of protocol */
 recorda(MyName, current_state(NewState)).
```

The JADE agent should only call *move_to_next* and execute the *react* method if move to next does not fail, and the *unexpected* method otherwise.

5.2 Messages sending

When an agent wants to send a message it must check which messages are allowed by the protocol in the current state. In order to do this, the Jade agent can call the *outMsgs* Prolog predicate that returns all messages that it can send in the current state of protocol.

```
/* Messages that agent can send according to
   current state of the protocol */
outMsgs(MyName, ListToSend) :-
  /* Current state of the agent protocol */
  recorded(MyName, current_state(LastState), _),
  /* Find all possible next state
     where agent is the sender */
  findall(
   msg(MyName, ReceiverV, PerformativeV, ContentV),
   next(
    0, LastState,
    msg(MyName, ReceiverV, PerformativeV, ContentV), _,
    0, MyName),
   ListToSend).
```

The Jade agent should only select one message from the list of messages returned by the predicate using the select messages method.

5.3 Problems encountered only with JADE

The interpreter implementation in JADE was more complicated than in Jason, indeed we found many more different problems.

The main problems can be summarized in two specific cases:

- the JPL library does not support cyclic terms;
- SWI Prolog assert predicate does not allow a cyclic term as argument.

It is easy to note that the second problem result from a lack of SWI Prolog in the management of cyclic terms.

In order to solve the first problem, we had to create "super predicates", which are simply collections of predicates, to ensure that all intermediate executions are made within Prolog and no cyclic term is returned to JADE.

The second problem was solved instead using another predicate inside SWI Prolog; indeed, the *record* predicate supports cyclic terms and has a behavior similar to the *assert* predicate.

6 Related Work

A large part of the state of the art analysis presented in this section was published in [11].

Our work falls in the research area on self-adaptive systems which spun off from the wider area of distributed systems, be them based on web services, software agents, robots, or on other autonomous entities that need to react to unforeseen changes during their execution. Many surveys have been conducted to identify the main features of self-adaptive MASs [12, 15, 22, 25, 26] and interesting and original solutions have been proposed by the research community.

Proposals for standardizing the concepts involved in the self-adaptation process include a meta-model to describe intelligent adaptive systems in open environments [16] and a taxonomy of adaptive agent-based collaboration patterns [7], for their analysis and exploitation in the area of autonomic service ensembles. An analysis of linguistic approaches for self-adaptive software is presented in [24].

The approaches closer to ours focus on formalizing protocols that the agents may use during their life, including specific protocols to deal with unforeseen events: in these approaches agents are usually free to choose, from a bunch of usable protocols, which one they prefer, maintaining in this way the freedom to autonomously self adapt to the new situation but ensuring at the same time that a feasible interaction pattern is followed. Our work can be included in this research field, where we can speak of "protocol enforcement" or "protocol-driven agents".

As far as self-adaptiveness of protocol-driven agents is concerned, the main sources of inspiration were [8, 9, 20, 21]. In [8] the authors propose a dynamic self-monitoring and self-regulating approach based on norms to express properties

which allow agents to control their own behavior. In [20] and [21] agents operating in open and heterogeneous MASs dynamically select protocols, represented in FIPA AUML, in order to carry collaborative tasks out. Since the selection is performed locally to the agent, some errors may occur in the process. The proposed mechanism provides the means for detecting and overcoming them.

Comparison. To the best of our knowledge, there are no approaches similar to ours presented in the MAS literature.

In Fornara at al [13, 14, 19] the authors discuss their approach based on Normative MAS. An artificial institution catches the institutional events and verifies them with respect to a normative specification. As a result, protocol specifications are a special case with respect to a normative specification. So, even if the approach is different the aim is similar, i.e. to deal with open multiagent systems and monitor their correctness w.r.t. a specification.

Normative system approaches offer other advantages for multi agent systems because agents may integrate their practical reasoning with reasoning about the normative specification, although, also our protocol-driven agents could reason about trace expressions which are a First Class Entities.

Other closely related proposals are those by Criado et al. [10] and Baldoni et al. [4, 3, 17, 5]; in these papers, the authors suggest a way to implement a monitoring mechanism by exploiting the A&A metamodel and by reifying commitment-based protocols into artifacts. The proposal is implemented both on top of Cartago and Jade and on top of Jason/JaCaMo. However, our work is different from theirs, because – at least from the Runtime Verification application – our approach is less invasive, in fact, it works with each possible MASs architecture and not with only customized implementations.

If we consider our previous work, before upgrading with Template Global Types, one reason why our approach was different from others, was that the projection function took protocol specifications and returned protocol specifications expressed in the same language. Usually, projection functions return either agent stubs/code (common in the MAS community) or protocol specifications in a language suitable for expressing the agent local viewpoint, different from the language for expressing the global one (common in the session types community). Having a unique formalism for protocol specification both at the global and at the local level is a simpler and more uniform approach.

In this paper we have shown the benefits of using parameters inside protocol specifications; in this way we have made protocols much more generic and flexible, also moving a step towards dynamic protocol generation.

7 Conclusions

In this paper we have presented our proposal to make the management of protocols more flexible and to move a step forward their dynamic generation. Two working prototypes exist, demonstrating the feasibility of our approach. While

integrating our parametric protocol-driven agents into Jason was easy because of its native support to Prolog, integrating them into JADE was not. However, that attempt – which, although not trivial, was successful – makes us confident in the possibility to integrate our approach into almost any agent framework, given that an interface between the framework language and Prolog is provided.

References

1. D. Ancona, M. Barbieri, and V. Mascardi. Constrained global types for dynamic checking of protocol conformance in multi-agent systems. In S. Y. Shin and J. C. Maldonado, editors, *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*, pages 1377–1379. ACM, 2013.
2. D. Ancona, S. Drossopoulou, and V. Mascardi. Automatic generation of self-monitoring MASs from multiparty global session types in Jason. In M. Baldoni, L. A. Dennis, V. Mascardi, and W. Vasconcelos, editors, *Declarative Agent Languages and Technologies X - 10th International Workshop, DALT 2012, Valencia, Spain, June 4, 2012, Revised Selected Papers*, volume 7784 of *Lecture Notes in Computer Science*, pages 76–95. Springer, 2012.
3. M. Baldoni, C. Baroglio, and F. Capuzzimati. A commitment-based infrastructure for programming socio-technical systems. *ACM Trans. Internet Technol.*, 14(4):23:1–23:23, Dec. 2014.
4. M. Baldoni, C. Baroglio, and F. Capuzzimati. Typing multi-agent systems via commitments. In F. Dalpiaz, J. Dix, and M. van Riemsdijk, editors, *Engineering Multi-Agent Systems*, volume 8758 of *Lecture Notes in Computer Science*, pages 388–405. Springer International Publishing, 2014.
5. M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio. Programming with commitments and goals in JaCaMo+. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1705–1706, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.
6. D. Briola, V. Mascardi, and D. Ancona. Distributed runtime verification of JADE and Jason multiagent systems with prolog. In L. Giordano, V. Gliozzi, and G. L. Pozzato, editors, *Proceedings of the 29th Italian Conference on Computational Logic, Torino, Italy, June 16-18, 2014.*, volume 1195 of *CEUR Workshop Proceedings*, pages 319–323. CEUR-WS.org, 2014.
7. G. Cabri, M. Puviani, and F. Zambonelli. Towards a taxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles. In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 508–515, 2011.
8. C. Chopinaud, A. El Fallah-Seghrouchni, and P. Taillibert. Automatic generation of self-controlled autonomous agents. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, pages 755–758, 2005.
9. M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Self-adaptive monitors for multiparty sessions. In *22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2014*, pages 688–696. IEEE, 2014.
10. N. Criado, E. Argente, P. Noriega, and V. J. Botti. Reasoning about constitutive norms in BDI agents. *Logic Journal of the IGPL*, 22(1):66–93, 2014.

11. A. F. Davide Ancona, Daniela Briola and V. Mascardi. Global protocols as first class entities for self-adaptive agents. *AAMAS2015*, 2015.
12. G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-organization in multi-agent systems. *Knowl. Eng. Rev.*, 20(2):165–189, 2005.
13. N. Fornara, F. Vigan, and M. Colombetti. Agent communication and artificial institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.
14. N. Fornara, F. Vigan, M. Verdicchio, and M. Colombetti. Artificial institutions: a model of institutional reality for open multiagent systems. *Artificial Intelligence and Law*, 16(1):89–105, 2008.
15. M.-P. Gleizes. Self-adaptive complex systems. In M. Cossentino, M. Kaisers, K. Tuyls, and G. Weiss, editors, *Multi-Agent Systems*, volume 7541 of *Lecture Notes in Computer Science*, pages 114–128. Springer Berlin Heidelberg, 2012.
16. T. Juan and L. Sterling. The ROADMAP meta-model for intelligent adaptive multi-agent systems in open environments. In P. Giorgini, J. Müller, and J. Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 53–68. Springer Berlin Heidelberg, 2004.
17. C. B. M. Baldoni and F. Capuzzimati. Reasoning about social relationships with Jason. *Autonomous Agents and Multi-Agent Systems*, 2014.
18. V. Mascardi and D. Ancona. Attribute global types for dynamic checking of protocols in logic-based multiagent systems. *TPLP*, 13(4-5-Online-Supplement), 2013.
19. D. Okouya, N. Fornara, and M. Colombetti. An infrastructure for the design and development of open interaction systems. In M. Cossentino, A. El Fallah Seghrouchni, and M. Winikoff, editors, *Engineering Multi-Agent Systems*, volume 8245 of *Lecture Notes in Computer Science*, pages 215–234. Springer Berlin Heidelberg, 2013.
20. J. G. Quenum, S. Aknine, O. Shehory, and S. Honiden. Dynamic protocol selection in open and heterogeneous systems. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Hong Kong, China, 18-22 December 2006*, pages 333–341, 2006.
21. J. G. Quenum, F. Ishikawa, and S. Honiden. Protocol selection alongside service selection and composition. In *2007 IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA*, pages 719–726, 2007.
22. R. de Lemos, H. Giese, H. A. Müller, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32, 2013.
23. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. R. Lesser and L. Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, pages 312–319. The MIT Press, 1995.
24. G. Salvaneschi, C. Ghezzi, and M. Pradella. An analysis of language-level support for self-adaptive software. *ACM Trans. Auton. Adapt. Syst.*, 8(2):7:1–7:29, 2013.
25. D. Weyns and M. Georgeff. Self-adaptation using multiagent systems. *Software, IEEE*, 27(1):86–91, 2010.
26. F. Zambonelli, N. Bicocchi, G. Cabri, L. Leonardi, and M. Puviani. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference on*, pages 108–113, 2011.

Leveraging Commitments and Goals in Agent Interaction

Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, Roberto Micalizio

Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino (Italy)
{firstname.lastname}@unito.it

Abstract. Modeling and regulating interactions among agents is a critical step in the development of Multiagent Systems (MASs). Some recent works assume a normative view, and suggest to model interaction protocols in terms of obligations. In this paper we propose to model interaction protocols in terms of goals and commitments, and show how such a formalization promotes a deliberative process inside the agents. In particular, we take a software engineering perspective, and balance the use of commitments against obligations inside interaction protocols. The proposal is implemented via JaCaMo+, an extension to JaCaMo, in which Jason agents can interact while preserving their deliberative capabilities by exploiting commitment-based protocols, reified by special CArtAgO artifacts. The paper shows how practical rules relating goals and commitments can be almost directly encoded as Jason plans to be used as building blocks in agent programming.

Keywords: Social Computing, Agent Programming, Commitments and Goals, Agents & Artifacts, JaCaMo

1 Introduction

Many researchers claim that an effective way to approach the design and development of a MAS consists in conceiving it as a structure composed of four main entities: *Agents*, *Environment*, *Interactions*, and *Organization* [32,18,19]. Such a separation of concerns enjoys many advantages from a software engineering point of view, since it enables a modular development of code that eases code reuse and maintainability. Currently, there are many frameworks that support designers and programmers in realizing one of these components (e.g., [8,10,25,11,29]). To the best of our knowledge, JaCaMo [9] is the the most complete among the well-established proposals, providing a thorough integration of the three components agents, environments, and organizations into a single programming framework. Another work along this direction is [7], which posits that agents and environments should be linked and interconnected through standard interfaces to fully leverage each of them.

A recent extension to JaCaMo [32] further enriches the framework by introducing an interaction component. The interaction component allows regulating

both agent interactions and the interactions between agents and environment. More precisely, an interaction component encodes –in an automaton-like shape– a *protocol*, in which states represent protocol steps, and transitions between states are associated with (undirected) *obligations* that can assume three forms: actions performed by the agents in the environment, messages that an agent sends to another agent, and events that an agent can perceive (i.e., events emitted from objects in the environment). Such protocols provide a *guideline* of how a given organizational goal should be achieved.

Interaction components, as defined in [32], however, present also some drawbacks. Works such as [15] show the importance, for the agents to be autonomous, to reason about the social consequences of their actions by exploiting *constitutive norms* that link the agents' actions to their respective social meanings. However, an interaction component operates as a coordinator that, by relying on obligations, issues commands about what an agent has to do, and when. This impedes agents from reasoning on the normative effects of their actions. On the one hand, the obligations are not constitutive norms while, on the other hand, the social meaning of such commands is not known to the agents but only implicitly encoded within the protocol. Agents lose part of their deliberative power since, once they join an interaction component, they have no other choice but deciding whether satisfying or not those obligations they are in charge of, while the rationale behind these obligations remains hidden to them. Consequently, this approach does not suit those situations where interaction is not subject to an organizational guideline, such as in the case when interaction is among agents and each agent decides what is best for itself [31], or when guidelines amount to declarative, underspecified constraints that still leave agents the freedom to take strategic decisions on their behavior.

Although we substantially agree with [32] about the importance of explicitly capturing the agents' interactions with appropriate abstractions, we also note that organization-driven guidelines, presented in that work, are but a kind of interaction; we thus propose a complementary approach which better supports the deliberative capabilities of the agents. Indeed, when organizational goals are not associated with corresponding guidelines, agent deliberation is crucial for the achievement of goals. An agent, in fact, has to act not only upon its own goals, but also upon what interactions could be necessary for achieving these goals. In other terms, an agent has to discover how to obtain a goal by interacting with others, i.e. to establish when to create an engagement, and which (sub)goals should be achieved first in order to fulfill its engagements. It is important to underline that when agents can fully exploit their deliberative capabilities, they can take advantage of opportunities (*flexibility*), and can find alternative ways to get their goals despite unexpected situations that may arise (*robustness*).

We claim that whenever guidelines are missing, the interactions among the agents should be supported by the very fundamental notions of goal and engagement. For this reason, we propose in this paper to complement the interaction protocol in [32], and more in general organizational and normative approaches [17,20,23,16], with an *interaction artifact* that can be used by the agents as a

common ground. Our interaction artifacts encode the notion of engagement as *social commitment* [26]. The choice of commitments stems by the fact that, differently from obligations, commitments are taken by an agent as a result of an internal deliberative process. They can be directly manipulated by the agents, and they have proved to be very effective in modeling (directed) social relationships. In addition, a recent work by Telang et al. [28] shows how goals and commitments are strongly interrelated. Commitments are therefore evidence of the capacity of an agent to take responsibilities autonomously. Citing Singh [27], an agent would become a debtor of a commitment based on the agent's own communications: either by directly saying something or having another agent communicate something in conjunction with a prior communication of the debtor. That is, there is a causal path from the establishment of a commitment to prior communications by the debtor of that commitment. By contrast, obligations can result from a deliberative process which is outside the agent; this is the case of the interaction component in [32]. This is the reason why we believe that the introduction of a deliberative process on constitutive rules that rely on obligations would not really support the agents' autonomy.

Practically, the proposal relies on the JaCaMo platform [9], and hence we dubbed it JaCaMo+: Jason agents engage commitment-based interactions which are reified as CArtAgO artifacts. CArtAgO is a framework based on the A&A meta-model [30,24] which extends the agent programming paradigm with the first-class entity of artifact: a resource that an agent can use, and that models working environments. It provides a way to define and organize workspaces, that are logical groups of artifacts, that can be joined by agents at runtime. The environment is itself programmable and encapsulates services and functionalities, making it active. JaCaMo+ artifacts represent the *interaction social state* and provide the roles agents enact. The use of artifacts enables the implementation of monitoring functionalities for verifying that the on-going interactions respect the commitments and for detecting violations and violators.

The paper extends and details the approach introduced in [3], and is organized as follows. Section 2 introduces some basic notions about goals and commitments. Section 3 discusses the extensions to JaCaMo that have been introduced in JaCaMo+; Section 4 shows, by exemplifying the FIPA Contract Net Protocol, how agents can be programmed by means of patterns encoding the interplay between goals and commitments.

2 Basic Notions

A social commitment models the directed relation between two agents: a *debtor* and a *creditor*, that are both aware of the existence of such a relation and of its current state: A commitment $\mathsf{C}(x, y, s, u)$ captures that agent x (debtor) commits to agent y (creditor) to bring about the consequent condition u when the antecedent condition s holds. Antecedent and consequent conditions are conjunctions or disjunctions of events and commitments. Unlike obligations, commitments are manipulated by agents through the standard operations *create*,

cancel, *release*, *discharge*, *assign*, *delegate* [26]. A commitment is autonomously taken by a debtor towards a creditor on its own initiative, instead of dropping from an organization, like obligations. This preserves the autonomy of the agents and is fundamental to harmonize deliberation with goal achievement. The agent does not just react to some obligations, but it rather includes a deliberative capacity by which it creates engagements towards other agents while it is trying to achieve its goals (or to the aim of achieving its goals). Since debtors are expected to satisfy their engagements, commitments satisfy the requirement in [14] of having a normative value, providing social expectations on the agents' behaviors, as well as obligations. Commitments also satisfy the requirement in [17] that when parties are autonomous, social relationships cannot but concern the *observable* behavior of the agents themselves.

Commitment-based protocols assume that a (notional) *social state* is available and inspectable by all the involved agents. The social state traces which commitments currently exist between any two agents, and the states of these commitments according to the commitments lifecycle. By relying on the social state, an agent can deliberate to create further commitments, or to bring about a condition involved in some existing commitment. Most importantly, commitments can be used by agents in their practical reasoning together with beliefs, intentions, and *goals*. In particular, Telang et al. [28] point out that goals and commitments are one another complementary: A commitment specifies how an agent relates to another one, and hence describes what an agent is willing to bring about for another agent. On the other hand, a goal denotes an agent's proattitude towards some condition; that is, a state of the world that the agent should achieve. An agent can create a commitment towards another agent to achieve one of its goals; but at the same time, an agent determines the goals to be pursued relying on the commitments it has towards others: A commitment is satisfied when the related goal is achieved. Note that, similarly to commitments, goals have their own lifecycle that evolves according to the actions performed by the agents (leading to the achievement or unfulfillment of goals), but also to the decisions of the agents to purse, suspend, or drop the goals themselves.

In [28], a goal G is formalized as $\mathsf{G}(x, p, r, q, s, f)$, where x is the agent pursuing G, p is a precondition that must be satisfied before G can become Active, r is an invariant condition that is true when G becomes Active and holds until the achievement of G, q is a post-condition (effect) that becomes true when G is successfully achieved, and finally, s and f are the success and failure conditions, respectively. In the following sections we will show how such a formalization can be mapped into Jason plans, and how it turns out to be useful for the agent programming purpose.

3 JaCaMo+

JaCaMo [9] is a platform integrating Jason (as an agent programming language), CArtAgO (as a realization of the A&A meta-model [30]), and Moise (as a support to the realization of organizations). In this section we shortly describe how

JaCaMo+ is obtained by extending the CArtAgO and Jason components of the standard JaCaMo.

3.1 Extending CArtAgO

Exploiting [1], JaCaMo+ enriches CArtAgO's artifacts with an explicit representation of commitments and of commitment-based protocols. The resulting class of artifacts reifies the execution of commitment-based protocols, including the social state of the interaction, and enables Jason agents both to be notified about the social events and to perform practical reasoning also about the other agents. This is possible thanks to the *social expectations* raised by commitments. Since an artifact is a programmable, active entity, it can act as a monitor of the in progress interaction. The artifact can therefore detect violations that it can ascribe to the violator without the need of agent introspection.

Specifically, a JaCaMo+ artifact encodes a commitment protocol, that is structured into a set of roles. By enacting a role, an agent gains the rights to perform social actions, whose execution has public social consequences, expressed in terms of commitments. If an agent tries to perform an action which is not associated with the role it is enacting, the artifact raises an exception that is notified to the violator. On the other hand, when an agent performs a protocol action that pertains to its role, the social state is updated accordingly by adding new commitments, or by modifying the state of existing commitments.

In CArtAgO, the Java annotation[1] *@OPERATION* marks a public operation that agents can invoke on the artifact. In JaCaMo+, a method tagged with *@OPERATION* corresponds to a protocol action. We also add the annotation *@ROLE* to specify which roles are enabled to use that particular action. Another extension is an *explicit representation of the social state*, which is maintained within the artifact. By *focusing* on an artifact, an agent registers to be notified of events that are generated inside the artifact. Note that all events that amount to the execution of protocol actions/messages are recorded as facts in the social state. This is done for the sake of a greater degree of decoupling between actions/events/messages and their effects [5,6]. In particular, when the social state is updated, the JaCaMo+ artifact provides such information to the focusing JaCaMo+ agents by exploiting proper *observable properties*. Agents are, thus, constantly aligned with the social state.

3.2 Extending Jason

Jason [10] implements in Java, and extends, the agent programming language AgentSpeak(L). Jason agents have a BDI architecture. Each has a belief base, and a plan library. It is possible to specify *achievement* (operator '!') and *test* (operator '?') goals. Each plan has a triggering event (causing its activation),

[1] Annotations, a form of metadata, provide data about a program that is not part of the program itself. See https://docs.oracle.com/javase/tutorial/java/annotations/

which can be either the addition or the deletion of some belief or goal. The syntax is inherently declarative. In JaCaMo, the beliefs of Jason agents can also change due to operations performed by other agents on the CArtAgO environment, whose consequences are automatically propagated. We extend the *Jason* component of JaCaMo by allowing the specification of plans whose triggering events involve commitments. JaCaMo+ represents a commitment as a term $cc(debtor, creditor, antecedent, consequent, status)$ where *debtor* and *creditor* identify the involved agents (or agent roles), while *antecedent* and *consequent* are the commitment conditions. *Status* is the commitment state (the set being defined in the commitments life cycle [21]). Commitments operations (e.g. create, see Section 2) are realized as *internal operations* of the new class of artifacts we added to CArtAgO. Thus, commitment operations cannot be invoked directly by the agents, but the protocol actions will use them as primitives to modify the social state.

A Jason plan is specified as:

$$triggering\_event : \langle context \rangle \leftarrow \langle body \rangle$$

where the *triggering_event* denotes the events the plan handles, the *context* specifies the circumstances when the plan could be used, the *body* is the course of action that should be taken. In a Jason plan specification, commitments can be used wherever beliefs can be used. Otherwise than beliefs, their assertion/deletion can only occur through the artifact, in consequence to a social state change. The following template shows a Jason plan triggered by the addition of a commitment in the social state:

$$+cc(debtor, creditor, antecedent, consequent, status) : \langle context \rangle \leftarrow \langle body \rangle.$$

More precisely, the plan is triggered when a commitment, that unifies with the one in the plan head, appears in the social state. The syntax is the standard for Jason plans. *Debtor* and *creditor* are to be substituted by the proper roles. The plan may be devised so as to change the commitment status (e.g. the debtor will try to satisfy the comment), or it may be devised so as to allow the agent to react to the commitment presence (e.g., collecting information). Similar schemas can be used for commitment deletion and for the addition/deletion of social facts. Further, commitments can also be used in contexts and in plans as test goals ($?cc(\dots)$), or achievement goals ($!cc(\dots)$). Addition or deletion of such goals can, as well, be managed by plans; for example:

$$+!cc(debtor, creditor, antecedent, consequent, status) : \langle context \rangle \leftarrow \langle body \rangle.$$

The plan is triggered when the agent creates an achievement goal concerning a commitment. Consequently, the agent will act upon the artifact so as to create the desired social relationship. After the execution of the plan, the commitment $cc(debtor, creditor, antecedent, consequent, status)$ will hold in the social state, and will be projected onto the belief bases of all agents focusing on the artifact.

4 Programming in JaCaMo+

In this section we show how Jason agents can be easily programmed by considering a commitment-based protocol as a guideline for the programmer: We first present a programming approach which exploits the practical rules by Telang et al. [28], and then we exemplify the approach implementing the initiator and participant agents of the well-known Contract-Net Protocol (CNP).

4.1 Practical Rules as Programming Code-Blocks

For both goals and commitments, [28] defines lifecycles, and operations through which the state of a goal, or a commitment, evolves over time. The relationship between goals and commitments is formalized in terms of *practical* rules, which capture patterns of pragmatic reasoning. They include: (1) rules from goals to commitments to capture how commitments evolve when the state of some goals change; and (2) rules from commitments to goals to capture how a goal evolves when the corresponding commitment changes in the social state. These rules can be easily encoded in JaCaMo+, and used by a programmer as templates for implementing Jason agents. In the following we discuss four examples of rules that will be used in the CNP scenario.

Goal rules. This JaCaMo+ template tackles the case when a goal $G = \mathsf{G}(x, p, r, q, s, f)$ appears in the knowledge base of agent x; namely, x wants to achieve the success condition s, and hence an appropriate plan is triggered:

```
+!G : p
  <-?r
    ⟨body⟩ /*plan achieving condition s*/
    ?q.
```

Differently from [28], in JaCaMo+ we explicitly mention a plan of actions (the body) to achieve the success condition s. When x can satisfy G autonomously (no interaction is needed), conditions s and q coincide. Instead, when x cannot satisfy G (or it is not convenient for x to achieve G autonomously), the body will involve an interaction with another agent and, as we will see, conditions q and s will differ. Note that, in JaCaMo+ we can also specify a plan to be triggered when the failure condition is reached:

```
-!G : f
  <-
    ⟨body⟩. /*plan handling failure condition f*/
```

The following three templates reflect namesake rules in [28].

Entice. Agent x can achieve G with the help of agent y: x creates an offer to agent y such that, if y brings about s (success condition of G), then x will engage into achieving a condition u of interest for y. Such an offer is naturally modeled as the commitment $C(x, y, s, u)$. The JaCaMo+ template is:

```
+!G : p
  <-?r
    social_action;
    ?cc(x, y, s, u, CONDITIONAL).
```

The body of the rule consists of a *social action*; namely, a protocol action offered by the artifact x is focused on, and whose meaning is the creation of a commitment $C = cc(x, y, s, u, CONDITIONAL)$. This commitment will push agent y to bring about the success condition s associated with G, thus this is a special case of goal activation. Note that the post condition of this rule corresponds to a test on the existence of the commitment C; agent x can verify, by inspecting the social state, that the commitment really exists.

Deliver. If commitment $C(x, y, s, u)$ becomes detached, then debtor x activates a goal $G_1 = \mathsf{G}(x, p, r, q, u, f)$ to bring about the consequent. In JaCaMo+:

```
+cc(x, y, s, u, DETACHED) : context
   <- !G1 ;
      ?cc(x, y, s, u, SATISFIED).
```

It is worth noting the test goal at the end of the rule: It allows x to verify that after the achievement of G_1, its corresponding commitment is now satisfied.

Detach. When a conditional commitment $C_1(y, x, s', t)$, appears in the social state, the creditor x activates a goal $G_2 = \mathsf{G}(x, p', r', q', s', f')$ to bring about the commitment antecedent. The JaCaMo+ template is:

```
+cc(y, x, s', t, CONDITIONAL) : context
   <- !G2 ;
      ?cc(y, x, s', t, DETACHED).
```

Note that, as in the previous case, agent x can verify that, after the satisfaction of goal G_2, the corresponding commitment is now detached.

4.2 JaCaMo+ Contract Net Protocol

As in [32], we assume that agents are assigned with institutional goals, defined in the Moise layer, to be achieved via the well-known Contract Net Protocol (CNP) protocol. We show how CNP can be implemented in JaCaMo+ by exploiting the templates introduced above. CNP (see Table 1) involves two roles: initiator (i) and participant (p). An agent playing the initiator role calls for proposals from agents playing the participant role. A participant makes a proposal if interested. Proposals can be accepted or rejected by initiator. *Accept*, *done*, and *failure* do not amount to commitment operations, but impact on the progression of commitment states, e.g., *accept* causes the satisfaction of the commitment created by *cfp*. Listing 1.1 reports an excerpt of the JaCaMo+ *CNP protocol artifact*

Table 1. CNP: actions and their social meaning.

initiator (i):	participant (p):
cfp: create(C(*i, p, propose, accept* $\vee$ *reject*))	*propose:* create(C(*p, i, accept, done* $\vee$ *failure*))
reject: release(C(*p, i, accept, done* $\vee$ *failure*))	*refuse:* release(C(*i, p, propose, accept* $\vee$ *reject*))
accept: "commitment progression"	*done:* "commitment progression"
	failure: "commitment progression"

implementation.

```
1  @OPERATION
2  @ROLE(name="initiator")
3  public void cfp(String task) {
4    RoleId initiator =
5       getRoleIdByPlayerName(getOpUserName());
6    this.defineObsProperty("task", task,
7                   initiator.getCanonicalName());
8    RoleId dest = new RoleId("participant");
9    createAllCommitments(new Commitment(initiator,
10     dest,"propose","accept OR reject"));
11   assertFact(new Fact("cfp", initiator, task));
12 }
13 @OPERATION
14 @ROLE(name="participant")
15 public void propose(String prop, int cost, String init) {
16   Proposal p = new Proposal(prop, cost);
17   // ...
18   defineObsProperty("proposal",
19      p.getProposalContent(),p.getCost(),
20      participant.getCanonicalName());
21   createCommitment(new Commitment(participant,
22      initiator,"accept", "done OR failure"));
23   assertFact(new Fact("propose", participant, prop));
24   actualProposals++;
25   if (actualProposals == numberMaxProposals) {
26     // ...
27     createCommitment(new Commitment(initiator,
28              groupParticipant,"true", "accept OR reject"));
29   }
```

Listing 1.1. The CNP artifact in JaCaMo+.

cfp (line 3) is a protocol action, realized as a CArtAgO operation (CArtAgO Java annotation @OPERATION, line 1). It can be executed only by an *initiator* (JaCaMo+ Java annotation @ROLE(name="initiator"), line 2). It publishes the task for the interaction session as an observable property of the artifact (line 6). All agents focusing on the artifact will have this information added to their belief bases. The social effect of *cfp* is the creation (line 9) of as many commitments as participants to the interaction, and of a social fact (line 11), that tracks the call made by the initiator. These effects will be broadcast to all focusing agents. *Accept* pertains to the initiator. It asserts a social fact, *accept*, which causes the satisfaction of one of the commitments created at line 9 towards a specific participant. *Propose* counts the received proposals and, when their number is sufficient, signals this fact to the initiator by the creation of a commitment (line 21) towards the group of participants. Below, the *JaCaMo+* initiator program:

```
1  /* Initial goals */
2  !startCNP.
3  /* Plans */
4  +!startCNP : true
5    <-  makeArtifact("cnp","cnp.Cnp",[],C);
6        focus(C);
7        enact("initiator").
8  +enacted(Id,"initiator",Role_Id)
9    <-  +enactment_id(Role_Id);
10       !solveTask("task-one").
11 +!solveTask(Task) /*ENTICE*/
12   : enactment_id(My_Role_Id)
13   <- +task(Task);
14      cfp(Task);
15      ?cc(My_Role_Id, Part_Role_Id, "propose",
```

```
16          "(accept or reject)", "CONDITIONAL").
17 +cc(My_Role_Id, "participant", "true", /*DELIVER*/
18          "(accept OR reject)", "DETACHED")
19    : enactment_id(My_Role_Id)
20      <-!acceptORreject;
21        ?cc(My_Role_Id, _, "true",
22            "(accept OR reject)", "SATISFIED").
23 +!acceptORreject
24    : not evaluated
25      <- +evaluated;
26      .findall(proposal(Content,Cost,Id),
27          proposal(Content,Cost,Id),Proposals);
28      .count(proposal(Content,Cost,Id),
29                  ProposalsNumb);
30      .min(Proposals,
31        proposal(Proposal,Cost,Winner_Role_Id));
32      +winner(Winner_Role_Id);
33      accept(Winner_Role_Id);
34      ?cc(My_Role_Id, Winner_Role_Id,
35           "true", "(accept OR reject)", "DETACHED").
36 %... action 'reject' for all other proposals ...
37 +done(Participant_role_id, Result)
38    : winner(Participant_role_id);
39    <- .print("Task resolved: ",Result).
40 +failure(Participant_role_id)
41    : winner(Participant_role_id);
42    <- .print("Task failed by ",Participant_role_id).
```

Listing 1.2. The initiator agent code in JaCaMo+.

The first ten lines are about the setting up of the environment. In this implementation, the initiator agent first creates the Cnp artifact (line 5), and then enact the initiator role (line 7). In general, however, the artifact could already be available, and an agent could just focus on it, and enact the initiator role. Note that the artifact notifies the agent the success of the enactment by asserting an *enacted* belief in the social state; note also that the agent receives a unique identifier, Role_Id, that will be used within the social state throughout the subsequent interactions (i.e., commitments will mention such an identifier).

After these preliminary steps, the initiator tries to reach the goal of having task-one performed: solveTask("task-one")[2]. This situation maps with the *ENTICE* rule from [28]; we, thus, follow the JaCaMo+ template associated with such a rule: see lines 11 - 16. The initiator, driven by its goal, performs the social action *cfp*, and thereby creates a commitment towards any participant that is focusing (or will focus) on that specific artifact. The execution of such action (which is performed by the initiator by its own initiative) modifies the social state; consequently, this modification is notified to the other focussing agents who will be in condition of taking this new social relationship into account in their own deliberative activity. The test goal concluding the rule allows the initiator to verify that at least one commitment has actually been created; namely, the entice has changed the social state.

Since the initiator has created a commitment, it must be ready to bring about the consequent of such a commitment whenever the antecedent will become true. The initiator must therefore contain a *DELIVER*-template plan; see lines 17-22

[2] To improve the readability of the code, we have simplified the notation in the Jason program by abstracting goals with simple labels.

in which the initiator, activated by the detachment of the commitment previously created with the *cfp* social action, starts a plan that will satisfy the commitment itself. The plan, acceptORreject (lines 23-33), first selects the best proposal, and then performs a social action accept towards the winner agent, and a social action reject towards any other participant that has not been selected.

The two last plans, done (line 37) and failure (line 40), are used by the initiator to monitor the actual completion of the task with either success or failure.

Let us now consider the participant side.

```
1  /* Initial goals */
2  !participate.
3  /* Plans */
4  +!participate : true
5    <-  focusWhenAvailable("cnp");
6        enact("participant").
7  +enacted(Id,"participant",My_Role_Id)
8    <-  +enactment_id(My_Role_Id).
9  +cc(Initiator_Role_Id, My_Role_Id, /*DETACH*/
10     "propose", "(accept OR reject)","CONDITIONAL")
11   :enactment_id(My_Role_Id)
12       & task(Task, Initiator_Role_Id)
13   <- !setup_proposal(Task, Initiator_Role_Id);
14       ?cc(Initiator_Role_Id, My_Role_Id,
15          "true", "(accept OR reject)","DETACHED").
16 +!setup_proposal(Task, Initiator_Role_Id)
17   : enactment_id(My_Role_Id)
18   <- !prepare_proposal(Task, Prop, Cost);
19       propose(Prop, Cost, Initiator_Role_Id);
20      +my_proposal(Prop, Cost, Initiator_Role_Id);
21      ?cc(My_Role_Id, Initiator_Role_Id, "accept",
22          "(done OR failure)", "CONDITIONAL").
23 +cc(My_Role_Id, Initiator_Role_Id, /*DELIVER*/
24        "true", "(done OR failure)", "DETACHED")
25   : enactment_id(My_Role_Id) &
26        accept(My_Role_Id)
27   <- ?my_proposal(Prop, Cost, Initiator_Role_Id);
28     !doneORfailure(Prop, Cost, Initiator_Role_Id).
29     ?cc(My_Role_Id, Initiator_Role_Id,
30        "true", "(done OR failure)", "SATISFIED").
31 +!doneORfailure(Prop, Cost, Initiator_Role_Id)
32  <- !compute_result(Prop, Cost, Result);
33    if (Result == "fail"){
34     failure(Initiator_Role_Id);
35    }
36    else {
37      done(Result, Initiator_Role_Id);
38    }.
39 +!compute_result(Prop, Cost, Result)
40   <- ⟨plan computing the result⟩.
```

Listing 1.3. The participant agent code in JaCaMo+.

A participant waits for calls for proposal by means of the CArtAgO basic operation *focusWhenAvailable* (line 5). A participant, thus, must be able to react whenever a new commitment of the form $cc(initiator, participant, propose, accept \vee reject)$ pops up in the social state. This behavior corresponds to the *DETACH* template, that is encoded in the JaCaMo+ plan in lines 9-15. In particular, the participant triggers a plan, setup_proposal that will satisfy the antecedent of the commitment. Such a plan, in fact, will include the social action propose (line 19). Note that the effect of action propose is twofold: (1) it asserts a fact "propose" in the social state, and hence satisfies the antecedent of

the triggering commitment; and (2) it also creates a new commitment from the participant to the initiator (see the protocol definition in Table 1), of the form $cc(p, i, accept, done \vee failure)$. This second commitment states that the participant is committed to carry out the task in case the initiator accepts its proposal. Thus, since the participant creates a commitment, it must also be ready to bring about the consequent of that commitment when the antecedent holds, and hence also the participant has a *DELIVER*-template plan in its program: see lines 23-30. In the specific case, the participant will activate a plan, doneORfailure, whose body will include the computation of a solution for the task at hand, and also the social actions done or failure depending on the, respectively, positive or negative result of the computation.

4.3 Final Remarks

One of the strongest points of JaCaMo+ is the *decoupling* between the design of the agents and the design of the interaction – that builds on the decoupling between computation and coordination done by coordination models like tuple spaces. Agent behavior is built upon agent goals and on its engagements with other agents, which are both the result of its deliberative process. For instance, in CNP the initiator becomes active when the commitments that involve it as a debtor, and which bind it to accept or reject the proposals, are detached. It is not necessary to specify nor to manage, inside the agent, such things as deadlines or counting the received proposals: the artifact is in charge of these aspects.

The decoupling allows us to change the definition of the artifact without the need of changing the agents' implementation. The Cnp class in Listing 1.1 detaches the commitments when a certain number of proposals is received. We can substitute such a class with class CnpTimer, which detaches commitments when a given deadline expires. This modification does not have any impact on the agents, whose programs remain unchanged, but for the line in which an agent focuses on (or creates) an artifact; e.g., for the initiator, the only change occurs in line 5 (see the following listing), in which the initiator creates a different type of artifact reifing the CNP protocol (the participant case is similar).

```
1 /* Initial goals */
2 !startCNP.
3 /* Plans */
4 +!startCNP : true
5   <-  makeArtifact("cnp","cnp.CnpTimer",[],C);
6       focus(C);
7       enact("initiator").
```

Listing 1.4. The initiator code, using CnpTimer.

Table 2 compares JaCaMo (with interaction [32]), with JaCaMo+ along some important characteristics that a MAS should feature. Let us discuss these dimensions, with a particular attention to those where the two platforms differ from one another. JaCaMo and JaCaMo+ do not equally support *autonomy*, in the sense that an agent can autonomously selects its own duties. JaCaMo with interaction just offers an agent to follow a predetermined path (a guideline) through which the agent has to fulfill a precise pattern of obligations. JaCaMo+, instead,

	JaCaMo with Interaction	JaCaMo+
Autonomous selection of obligations	✗	✓
Maintainability	✓	✓
Monitoring Support	✓	✓
Modular Definition of Protocols	✗	✓
Flexibility	✗	✓
Robustness	✗	✓
Interaction not spread across the agents code	✓	✓

Table 2. Comparison among JaCaMo with interaction and JaCaMo+.

offers an agent a tool, the interaction artifact, through which it can communicate with other agents and act together with others. The choice, however, of how and when been involved into an interaction remains within the scope of the agents. The adoption of commitments, in fact, assures that an agent assumes the responsibility for a task only when, by its own choice, performs a specific action on the interaction artifact. This has an impact on the property of flexibility and robustness. An interaction that is structured based on obligations only hinders agents when they need to adapt to unforeseen conditions (flexibility) or when they need to react to unwanted situations (robustness). The agent, in fact, is not free to delegate obligations, schedule them differently, etc. All the agent can do is to perform the actions that, instructed by the interaction protocol, resolve its obligations.

Protocols in [32] aim at defining guidelines to the use of resources in an organization. This, however, limits the modularity of *interaction* protocols because protocols depend on operations that are defined in the organization and there is no explicit association of which actions pertain to which roles. Thus, for instance, a participant may execute a cfp and the interaction artifact would allow it to do so. JaCaMo+ interaction protocols, instead, include the definitions of the needed operations, and specify which of them will empower the various role players. For both proposals the interaction logic is captured by the artifact and is not spread across the agent codes. Both include functionalities for monitoring the on-going interaction. In [32] the normative structure is leveraged to this aim, while in JaCaMo+ this can be done inside each of the protocol artifacts.

5 Conclusions

In this paper we presented JaCaMo+, and extension to JaCaMo that enables social behaviors into its agents. We started from the interaction protocols based on obligations proposed in [32]. These protocols are suitable for modeling interactions among different elements of a MAS (i.e., not only interactions between agents, but also between agents and objects). However, obligation-based protocols reduce agent interactions to messages that an agent is obliged to send to another agent; that is, social relationships among agents are not handled directly. In other words, an obligation-based protocol can be adopted only in an

organization that gives guidelines about how interactions should be carried on, but it is not applicable in those organizations where similar guidelines are not available.

To cope with these more challenging situations, our intuition is to define an interaction in terms of goals and commitments. Commitments, in fact, are at the right level of abstraction for modeling directed relationships between agents. Moreover, since commitments have a normative power, they enable the agents to reason about the behavior of others; a commitment creates expectations in the creditor about the behavior that the debtor will assume in the near future.

Note that our view is also backed up by the practical rules discussed in [28], which highlight how goals and commitments are each other related. In particular, in this paper we have proposed to use the same rules as a sort of methodology for programming the Jason agents. An initial implementation of our proposal is provided by the JaCaMo+ platform. The tests (which involve from 5 to 100 agents) show that it scales up quite well, despite the introduction of commitments, but a more thorough testing will be performed in the near future.

The shift from obligations to commitments is beneficial in many respects. First of all, the autonomy of the agents is better supported because, although charged with goals to be achieved, they are free in deciding how to fulfill their goals. It follows that agents are *deliberative*, and this paves the way to self-* applications, including the ability to autonomously take advantage from opportunities, and the ability of properly reacting to unexpected events (self-adaptation). For instance, by finding a way for accomplishing an organizational goal taking into account the current state of the MAS, which is hardly foreseeable at design time. Moreover, the interplay between goals and commitments opens the way to the integration of self-governance mechanisms into organizational contexts. Thus, our concluding claim is that directly addressing social relationships increases the robustness of the whole MAS.

In the future, we intend to investigate how agents can leverage on their deliberative capabilities, and use it not only to program interactions, but to plan social interactions. Moreover, the modular nature of the implementation facilitates the development of extensions for tackling richer, data-aware contexts [12,22,13]. We are also interested in tackling, in the implementation, a more sophisticate notion of social context and of enactment of a protocol in a social context [4], as well as to introduce a typing system along the line of [2].

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments, which helped improving the paper.

References

1. Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. Social computing in JaCaMo. In *Proc. of ECAI*, volume 263 of *Frontiers in Artificial Intelligence*

and Applications, pages 959–960. IOS Press, 2014.
2. Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. Typing Multi-Agent Systems via Commitments. In F. Dalpiaz, J. Dix, and M. B. van Riemsdijk, editors, *Post-Proc. of the 2nd International Workshop on Engineering Multi-Agent Systems, EMAS 2014, Revised Selected and Invited Papers*, number 8758 in LNAI, pages 388–405. Springer, 2014.
3. Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Programming with Commitments and Goals in JaCaMo+ (Extended Abstract). In *Proc. of 14th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015*, July 4th-8th 2015.
4. Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, and Munindar P. Singh. Composing and Verifying Commitment-Based Multiagent Protocols. In M. Wooldridge and Q. Yang, editors, *Proc. of 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, Buenos Aires, Argentina, July 25th-31th 2015.
5. Matteo Baldoni, Cristina Baroglio, Elisa Marengo, and Viviana Patti. Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach. *ACM Trans. on Intelligent Sys. and Tech., Special Issue on Agent Communication*, 4(2):22:1–22:25, March 2013.
6. Matteo Baldoni, Cristina Baroglio, Viviana Patti, and Elisa Marengo. Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach (Extended Abstract). In M. Wooldridge and Q. Yang, editors, *Proc. of 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, Buenos Aires, Argentina, July 25th-31th 2015.
7. Tristan M. Behrens, Koen V. Hindriks, and Jürgen Dix. Towards an environment interface standard for agent platforms. *Annals of Mathematics and Artificial Intelligence*, 61(4):261–295, 2011.
8. Fabio L. Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, 2007.
9. Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747 – 761, 2013.
10. Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.
11. Frances M. T. Brazier, Barbara M. Dunin-Keplicz, Nick R. Jennings, and Jan Treur. Desire: Modelling Multi-Agent Systems in a Compositional Formal Framework. *Int. J. of Cooperative Information Systems*, 06(01):67–94, March 1997.
12. Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1):85–130, 2013.
13. Amit K. Chopra and Munindar P. Singh. Cupid: Commitments in relational algebra. In *Proc. of the 29th AAAI Conf*, pages 2052–2059. AAAI Press, 2015.
14. Rosaria Conte, Cristiano Castelfranchi, and Frank Dignum. Autonomous Norm Acceptance. In *ATAL*, volume 1555 of *LNCS*, pages 99–112. Springer, 1998.
15. Natalia Criado, Estefania Argente, Pablo Noriega, and Vicent Botti. Reasoning about constitutive norms in BDI agents. *Logic Journal of IGPL*, 22(1):66–93, 2014.
16. Natalia Criado, Estefania Argente, Pablo Noriega, and Vicent Botti. Reasoning about norms under uncertainty in dynamic environments. *International Journal of Approximate Reasoning*, 2014.
17. Mehdi Dastani, Davide Grossi, John-Jules Ch. Meyer, and Nick A. M. Tinnemeier. Normative Multi-agent Programs and Their Logics. In *KRAMAS*, volume 5605 of *LNCS*, pages 16–31. Springer, 2008.

18. Yves Demazeau. From interactions to collective behaviour in agent-based systems. In *In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*, 1995.
19. Frodi Hammer, Alireza Derakhshan, Yves Demazeau, and Henrik Hautop Lund. A multi-agent approach to social human behaviour in children's play. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 403–406. IEEE Computer Society, 2006.
20. Felipe Meneguzzi and Michael Luck. Norm-based behaviour modification in BDI agents. In *AAMAS (1)*, pages 177–184. IFAAMAS, 2009.
21. Felipe Meneguzzi, Pankaj R. Telang, and Munindar P. Singh. A first-order formalization of commitments and goals for planning. In *AAAI*. AAAI Press, 2013.
22. Marco Montali, Diego Calvanese, and Giuseppe De Giacomo. Verification of data-aware commitment-based multiagent system. In *Proc. of AAMAS*, pages 157–164. IFAAMAS/ACM, 2014.
23. Daniel Okouya, Nicoletta Fornara, and Marco Colombetti. An infrastructure for the design and development of open interaction systems. In M. Cossentino, A. El Fallah Seghrouchni, and M. Winikoff, editors, *Post-Proc. of the 2nd International Workshop on Engineering Multi-Agent Systems, EMAS 2014, Revised Selected and Invited Papers*, number 8245 in LNAI, pages 215–234. Springer, 2013.
24. Andrea Omicini, Alessandro Ricci, and Mirko Viroli. Artifacts in the a&a meta-model for multi-agent systems. *JAAMAS*, 17(3):432–456, 2008.
25. Andrea Omicini and Franco Zambonelli. TuCSoN: a coordination model for mobile information agents. In *Proc. of IIIS*, pages 177–187. IDI – NTNU, Trondheim (Norway), 8–9 June 1998.
26. Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
27. Munindar P. Singh. Commitments in multiagent systems some controversies, some prospects. In Fabio Paglieri, Luca Tummolini, Rino Falcone, and Maria Miceli, editors, *The Goals of Cognition. Essays in Honor of Cristiano Castelfranchi*, chapter 31, pages 601–626. College Publications, London, 2011.
28. Pankaj R. Telang, Neil Yorke-Smith, and Munindar P. Singh. Relating Goal and Commitment Semantics. In *Proc. of ProMAS*, volume 7212 of *LNCS*, pages 22–37. Springer, 2012.
29. Alexander Thiele, Thomas Konnerth, Silvan Kaiser, Jan Keiser, and Benjamin Hirsch. Applying JIAC V to Real World Problems: The MAMS Case. In *MATES*, volume 5774 of *LNCS*, pages 268–277. Springer, 2009.
30. Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *JAAMAS*, 14(1):5–30, 2007.
31. Pinar Yolum and Munindar P. Singh. Commitment Machines. In *Intelligent Agents VIII, 8th Int. WS, ATAL 2001*, volume 2333 of *LNCS*, pages 235–247. Springer, 2002.
32. Maicon R. Zatelli and Jomi F. Hübner. The Interaction as an Integration Component for the JaCaMo Platform. In F. Dalpiaz, J. Dix, and M. B. van Riemsdijk, editors, *Post-Proc. of the 2nd International Workshop on Engineering Multi-Agent Systems, EMAS 2014, Revised Selected and Invited Papers*, number 8758 in LNAI, pages 431–450. Springer, 2014.

Evaluating Compliance: From LTL to Abductive Logic Programming

Marco Montali[1], Federico Chesani[2], Marco Gavanelli[3], Evelina Lamma[3], and Paola Mello[2]

[1] Free University of Bozen-Bolzano
Piazza Domenicani 3, 39100 – Bolzano (Italy)
montali@inf.unibz.it
[2] University of Bologna
V.le Risorgimento 2, 40136 – Bologna (Italy)
{ federico.chesani | paola.mello }@unibo.it
[3] University of Ferrara
Via Saragat 1, 44122 – Ferrara (Italy)
{ marco.gavanelli | evelina.lamma }@unife.it

Abstract. The *compliance verification* task amounts to establishing if the execution of a system, given in terms of observed events, does respect a given property. In the past both the frameworks of Temporal Logics and Logic Programming have been extensively exploited to assess compliance. In this work we review the LTL and the Abductive Logic Programming frameworks in the light of compliance evaluation, and formally investigate the relationship between the two approaches. We define a notion of compliance within each approach, and then we show that an arbitrary LTL formula can be expressed in SCIFF, by providing an automatic translation procedure from LTL to SCIFF which preserves compliance.

Keywords: Linear Temporal Logic, Abductive Logic Programming, Compliance Verification, Business Process Management.

1 Introduction

Linear Temporal Logic (LTL) [12] specifications are traditionally used for expressing the properties that a reactive system should exhibit (or avoid), and are exploited by model checking tools for formal verification (e.g., [15,9]). Recently, LTL has been also used to describe the system under study itself, in fields like Business Process Management (BPM) and Service Oriented Computing (SOC): e.g., the DECLARE system [24] and the ConDec language [23,20] adopt LTL to model business processes, business rules and policies.

In these domains, a relevant task is to assess *compliance*, usually defined as checking if an implementation faithfully meets the requirements of a specification. The LTL models correspond to linear Kripke structures representing the

execution traces (i.e., sequences of events) occurred during a specific instantiation of the system, while entailment becomes a compliance evaluator w.r.t. a *regulatory specification* expressed as an LTL formula. Such approach has been used, for example, in [6] for static compliance verification of BPMN business processes, and in [1] for auditing event logs.

Recently, Logic Programming (LP) based approaches have been applied for specification and verification of normative systems [5,14], web services [25,7] and business processes as well [11,20]. The LP framework nicely meets the advantages of a declarative, first-order specification, grounded on a model-based semantics, and equipped with an operational proof procedure. Abductive Logic Programming (ALP, [17]), in particular, integrates abductive reasoning into LP, supporting an hypothesis-making mechanism.

In [2] we have defined the abductive proof procedure named SCIFF, originally developed for specification and verification of open societies of "computees" (a sort of agents), and later applied to normative systems [4,8], web service interaction [3,21] and BPM [22,20]. SCIFF specifications are given in terms of integrity constraints linking occurring events to expectations about the future course of events, and the declarative semantics has been given in terms of compliance of a given trace with respect to a SCIFF specification.

In this paper we investigate the relation between the LTL-based approach and the SCIFF framework, showing that if we focus on the compliance task, an LTL model can be (formally and correctly) translated into a SCIFF one. Starting from the seminal work in [13] about Separated Normal Forms (SNF) for LTL formulae, we define proper mapping functions and show how any LTL formula can be expressed within the SCIFF formalism. Then, we formally define the notion of compliance in both the approaches, we identify a tight equivalence relation, and we prove how such equivalence is indeed maintained when moving from the LTL approach to the SCIFF-based one.

2 Linear Temporal Logic

In this section, we provide a brief introduction to (propositional) Linear-time Temporal Logic (LTL), in particular w.r.t. the notion of *compliance*; the interested reader can refer to [12] for a more general introduction.

LTL formulae are built up from *atomic propositions*, whose truth values change over time. The LTL *time structure* $\mathcal{F}$, also called *frame*, models a single, linear timeline; formally, $\mathcal{F}$ is a totally ordered set $(\mathcal{K}, \prec)$ [12].

Definition 1 (LTL model). *Let $\mathcal{P}$ be the set of all atomic propositions in the system. An LTL model $\mathcal{M}$ for $\mathcal{P}$ is a triple $(\mathcal{K}, \prec, \upsilon)$ where $\upsilon : \mathcal{P} \rightarrow 2^{\mathcal{K}}$ is a* function *which maps each proposition in $\mathcal{P}$ to the set of time instants at which the proposition holds.*

We are interested in systems characterized by dynamics consisting of a stream of events. In this respect, each proposition represents a possible event that may occur in an instance of the system. More specifically, a proposition $e \in \mathcal{P}$ is

true in a certain state if at that state the event denoted by e occurs. Under this interpretation, LTL models correspond to execution traces.

Definition 2 (LTL execution trace). *Given a set $\mathcal{E}$ of atomic propositions (representing possible events), an* LTL execution trace $\mathcal{T}_\mathcal{L}$ *is an LTL model having $(\mathbb{N}, <)$ as time structure and $\mathcal{E}$ as the set of atomic propositions. In particular, $\mathcal{T}_\mathcal{L} = (\mathbb{N}, <, v_{occ})$, where $v_{occ} : \mathcal{E} \to 2^\mathbb{N}$ is a valuation function mapping each event $e \in \mathcal{E}$ to the set of all time instants $i \in \mathbb{N}$ at which e occurs.*

We will use the following abbreviations: $\mathcal{T}_\mathcal{L}(i)$ will denote the i-th state of $\mathcal{T}_\mathcal{L}$, i.e. the subset $\{e \in \mathcal{E} \mid i \in v_{occ}(e)\}$.

2.1 Syntax of LTL

LTL formulae are defined by using (i) *atomic propositions*, i.e., events, together with the two special constants $true$ and $false$; (ii) *classical propositional connectives*, i.e., $\neg$, $\wedge$, $\vee$ and $\Rightarrow$; (iii) *temporal operators*, i.e., $\bigcirc$ (next time), $\mathcal{U}$ (until), $\Diamond$ (eventually), $\Box$ (globally) and $\mathcal{W}$ (weak until). An LTL formula is recursively defined as: each event $e \in \mathcal{E}$ is a formula; if φ and ψ are formulae, then $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\psi$, and $\varphi \mathcal{U} \psi$ are formulae. Other LTL formulae can be defined as abbreviations:

- $\varphi \vee \psi \triangleq \neg(\neg\varphi \wedge \neg\psi)$ and $\varphi \Rightarrow \psi \triangleq \neg\varphi \vee \psi$;
- $true \triangleq \neg\varphi \vee \varphi$ and $false \triangleq \neg true$;
- $\Diamond\varphi \triangleq true \mathcal{U} \varphi$, $\Box\varphi \triangleq \neg\Diamond\neg\varphi$ and $\psi \mathcal{W} \varphi \triangleq \psi \mathcal{U} \varphi \vee \Box\psi$.

2.2 Semantics of LTL and Compliance

The semantics of LTL is given w.r.t. an LTL execution trace, and w.r.t. a specific state. We will use $\models_\mathcal{L}$ to denote the logical *entailment* in the LTL setting. $\mathcal{M}, i \models_\mathcal{L} \varphi$ means that φ is true at time i in model $\mathcal{M}$. $\models_\mathcal{L}$ is defined by induction on the structure of the formulae[4]:

$(\mathcal{T}_\mathcal{L} \models_\mathcal{L} \varphi)$ iff $(\mathcal{T}_\mathcal{L}, 0 \models_\mathcal{L} \varphi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} e)$ iff $e \in \mathcal{T}_\mathcal{L}(i)$ (i.e., $i \in v_{occ}(e)$);
$(\mathcal{T}_\mathcal{L}, i \not\models_\mathcal{L} e)$ iff $e \notin \mathcal{T}_\mathcal{L}(i)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \neg\varphi)$ iff $(\mathcal{T}_\mathcal{L}, i \not\models_\mathcal{L} \varphi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \varphi \wedge \psi)$ iff $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \varphi)$ and $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \varphi \vee \psi)$ iff $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \varphi)$ or $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \varphi \Rightarrow \psi)$ iff $(\mathcal{T}_\mathcal{L}, i \not\models_\mathcal{L} \varphi)$ or $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \bigcirc\varphi)$ iff $(\mathcal{T}_\mathcal{L}, i+1 \models_\mathcal{L} \varphi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi \mathcal{U} \varphi)$ iff $\exists k \geq i$ s.t. $(\mathcal{T}_\mathcal{L}, k \models_\mathcal{L} \varphi)$ and $\forall i \leq j < k$ $(\mathcal{T}_\mathcal{L}, j \models_\mathcal{L} \psi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \Diamond\varphi)$ iff $\exists j \geq i$ s.t. $(\mathcal{T}_\mathcal{L}, j \models_\mathcal{L} \varphi)$;
$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \Box\varphi)$ iff $\forall j \geq i$ $(\mathcal{T}_\mathcal{L}, j \models_\mathcal{L} \varphi)$;

[4] For the sake of readability, we explicitly show the semantics of $\Diamond$, $\Box$ and $\mathcal{W}$, even if their meaning can be obtained from the semantics of $\mathcal{U}$ and $\Box$.

$(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi \mathcal{W} \varphi)$ iff either $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \psi \mathcal{U} \varphi)$ or $(\mathcal{T}_\mathcal{L}, i \models_\mathcal{L} \Box \psi)$.

When LTL is employed to formalize compliance rules, the declarative semantics selects those events that must be contained (or avoided) in certain states so as to fulfil them, separating compliant traces from non-compliant ones. In this respect, $\models_\mathcal{L}$ plays the role of a *compliance evaluator*.

Definition 3 (LTL Compliance). *An LTL trace $\mathcal{T}_\mathcal{L}$ is* compliant *with a LTL formula φ if and only if $\mathcal{T}_\mathcal{L}$ entails φ:*

$$\textsc{cmp}_{\text{LTL}}(\mathcal{T}_\mathcal{L}, \varphi) \triangleq \mathcal{T}_\mathcal{L} \models_\mathcal{L} \varphi.$$

When LTL formulae are used to express business constraints/rules of a regulatory model, as for example in the ConDec language [23], then the LTL formula used for compliance is the conjunction of all formulae contained in the regulatory model. From an operational viewpoint, the compliance of a formula φ w.r.t. a $\mathcal{T}_\mathcal{L}$ is verified by means of model checking algorithms.

3 The SCIFF Framework

In the following we will briefly recap the main features of the SCIFF framework. The interested reader can refer to [2] for a detailed and comprehensive presentation. A SCIFF specification $\mathcal{S}$ is an Abductive Logic Program $\langle \mathcal{KB}, \mathcal{A}, \mathcal{IC} \rangle$ [17] where: (i) $\mathcal{KB}$ is a (static) knowledge base (a Logic Program [19]); (ii) $\mathcal{A}$ is a special set of predicates, called *abducibles*; two special abducibles, namely $\mathbf{E}/2$ and $\mathbf{EN}/2$, are used to represent the *expectations*; (iii) $\mathcal{IC}$ is a set of SCIFF integrity constraints, relating happened events with expectations.

Roughly speaking, given a goal $\mathcal{G}$, abductive reasoning looks for a set of literals Δ built from predicates $\mathcal{A}$ such that the goal is entailed by the program $\mathcal{KB} \cup \Delta$, and the set of integrity constraints $\mathcal{IC}$ is entailed too. The set Δ is referred to as an *abductive explanation* (see Definition 6).

Three special predicates are used to model happened events and positive/negative expectations. Happened events are denoted by using the (non abducible) predicate $\mathbf{H}(Ev, T)$, where Ev is a term representing the occurred event, while T explicitly represents the time at which the event occurred. In the remainder of this paper we will assume the time domain relies on natural numbers.

Definition 4 (SCIFF Execution Trace). *A SCIFF* execution trace $\mathcal{T}$ *(or simply a SCIFF trace) is a set of positive ground $\mathbf{H}(E, T)$ atoms.*

A specific execution of the system under study is called an *instance*, and it is formally identified by the SCIFF specification modeling the system and by the execution trace produced during the instance execution.

Definition 5 (SCIFF Instance). *Given a SCIFF specification $\mathcal{S} = \langle \mathcal{KB}, \mathcal{A}, \mathcal{IC} \rangle$ and a trace $\mathcal{T}$, $\langle \mathcal{S}, \mathcal{T} \rangle$ is an* instance *of $\mathcal{S}$.*

Positive and negative expectations model expected and forbidden events. They are represented by $\mathbf{E}(Ev, T)$ and $\mathbf{EN}(Ev, T)$, where Ev is a term describing the event, and T is a term or a variable. The intended meaning is that event Ev is expected to occur/not occur at time T.

SCIFF Integrity Constraints (IC) are mainly used to relate happened events with expectations. They are *body* $\rightarrow$ *head* rules, where *body* contains a conjunction of happened events, general abducibles, and defined predicates, while *head* contains a disjunction of conjunctions of expectations, general abducibles, and defined predicates. When the body is matched with events and abducibles, the IC is triggered, and expectations occurring in the head are assumed (abduced).

Definition 6 (Abductive explanation Δ). *Given a SCIFF instance $\langle \mathcal{S}, \mathcal{T} \rangle$, a set $\Delta \subseteq \mathcal{A}$ is an* abductive explanation *for $\langle \mathcal{S}, \mathcal{T} \rangle$ if and only if*

$$\text{Comp}\,(\mathcal{KB} \cup \mathcal{T} \cup \Delta) \cup \text{CET} \cup T_{\mathcal{X}} \models \mathcal{IC}$$

where Comp *is the (two-valued) completion of a theory [18],* CET *stands for Clark Equational Theory [10] and $T_{\mathcal{X}}$ is the CLP constraint theory [16], parametrized by the domain $\mathcal{X}$.*

We remind for completeness that CET is provided by the following axioms:

- $c \neq c'$ $\quad c, c'$ any pair of distinct constants
- $f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m)$ $\quad f, g$ any pair of distinct functors
- $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \ldots x_n = y_n$
- $f(x_1, \ldots, x_n) \neq c$ $\quad f$ any functor, c any constant
- $\tau(x) \neq x$ $\quad \tau(x)$ any term structure in which x is free.
- $x = y \rightarrow [W(x) \leftrightarrow W(y)]$ $\quad W$ any formula

together with the usual rules of reflexivity, symmetry and transitivity for equality. Fixing a CLP theory corresponds to instantiate the parameter $\mathcal{X}$ and the set of allowed constraints. Therefore, different structures can be chosen without affecting the notion of SCIFF's abductive explanation. We will instantiate such a parameter to $\mathbb{N}$, with linear equations and dis-equations. The theory of constraints $T_{\mathbb{N}}$ defines the symbols $+, -, *, /, =, >, <, \geq, \ldots$ with the usual meanings (e.g., $1 < 2 + 2$ is evaluated to *true*).

Remark 1 (Abductive explanations and sub-specifications). If Δ is an abductive explanation for $\langle \mathcal{S}, \mathcal{T} \rangle$, then Δ is an abductive explanation also for $\langle \mathcal{S}' = \langle \mathcal{KB}, \mathcal{A}, \mathcal{IC}' \rangle, \mathcal{T} \rangle$, where $\mathcal{IC}' \subseteq \mathcal{IC}$.

Being able to generate hypotheses might not be enough: in specific domains like, e.g., legal reasoning, a further step of verification of the hypotheses against the observed events (available data) is mandatory. Hence, the SCIFF framework provides also an *hypotheses-confirmation* mechanism, based on the formal notions of *fulfillment* and *violation*. First of all, expectations must be **E**-*consistent*: the same event cannot be expected and prohibited at the same time.

Definition 7 (E-consistency). *An abducible set Δ is* **E**-consistent *iff for each event e and for each time t it holds that $\{\mathbf{E}(e,t), \mathbf{EN}(e,t)\} \nsubseteq \Delta$*

The relationship between expectations and happened events is instead captured by the notions of *fulfillment* and *violation*.

Definition 8 ($\mathcal{T}$-Fulfillment). *Given a SCIFF trace $\mathcal{T}$ and an abducible set Δ, Δ is $\mathcal{T}$-*fulfilled *iff for each event e and for each time t: $\mathbf{E}(e,t) \in \Delta \rightarrow \mathbf{H}(e,t) \in \mathcal{T}$ and $\mathbf{EN}(e,t) \in \Delta \rightarrow \mathbf{H}(e,t) \notin \mathcal{T}$*

Definition 9 ($\mathcal{T}$-Violation). *Given a SCIFF trace $\mathcal{T}$ and an abducible set Δ, Δ is $\mathcal{T}$-*violated *iff it exists at least one event e and time t such that: $\mathbf{E}(e,t) \in \Delta \wedge \mathbf{H}(e,t) \notin \mathcal{T}$, or $\mathbf{EN}(e,t) \in \Delta \wedge \mathbf{H}(e,t) \in \mathcal{T}$*

Given an abductive explanation Δ, fulfillment acts as a classifier that separates the legal/correct execution traces with respect to Δ from the wrong ones.

Definition 10 (Compliance in SCIFF). *A trace $\mathcal{T}$ is* compliant *with a SCIFF specification $\mathcal{S}$ if and only if there exists an abducible set Δ such that:*

1. *Δ is an abductive explanation for $\langle \mathcal{S}, \mathcal{T} \rangle$;*
2. *Δ is* **E***-consistent;*
3. *Δ is $\mathcal{T}$-fulfilled.*

If this is the case, we write $\text{CMP}_{\text{SCIFF}}{}^{\Delta}(\mathcal{T}, \mathcal{S})$ or simply $\text{CMP}_{\text{SCIFF}}(\mathcal{T}, \mathcal{S})$.

4 Relating LTL and SCIFF

LTL and SCIFF rely on different logics, but when capturing regulatory models they both act as compliance evaluators, capturing the same idea of compliance. To capture some similarity w.r.t. compliance, we propose a mapping between LTL and SCIFF. First of all, we need to provide a mapping between an LTL trace $\mathcal{T}_{\mathcal{L}}$ and the corresponding SCIFF trace $\mathcal{T}$ (and vice versa).

Definition 11 (Trace mapping). *Given an LTL trace $\mathcal{T}_{\mathcal{L}} = (\mathbb{N}, <, v_{occ})$ and the set of atomic propositions $\mathcal{E}$, we map any possible pair (e,i) into a corresponding SCIFF event $\mathbf{H}(e,i)$, where $e \in \mathcal{E}$ and $i \in \mathbb{N}$.*

A trace mapping $\mathtt{tm}$ *is a transformation which maps an arbitrary LTL trace $\mathcal{T}_{\mathcal{L}}$ onto a corresponding SCIFF one, by applying the event mapping to each proposition belonging to $\mathcal{T}_{\mathcal{L}}$, i.e. to each $e \in \mathcal{E}$ and for each $i \in v_{occ}(e)$:*

$$\mathtt{tm}(\mathcal{T}_{\mathcal{L}}) = \{\mathbf{H}(e,i) | e \in \mathcal{E}, i \in v_{occ}(e)\}$$

Example 1. Let us consider an LTL execution trace $\mathcal{T}_{\mathcal{L}} = (\mathbb{N}, <, v_{occ})$, where $\mathcal{E} = \{a, b, c, d\}$ is the set of propositional events and v_{occ} is defined as follows:

$$v_{occ}(a) = \{0, 1\} \qquad v_{occ}(b) = \{2\} \qquad v_{occ}(c) = \{3\} \qquad v_{occ}(d) = \emptyset$$

Then $\mathtt{tm}(\mathcal{T}_{\mathcal{L}}) = \{\ \mathbf{H}(a,0),\ \mathbf{H}(a,1),\ \mathbf{H}(b,2),\ \mathbf{H}(c,3)\ \}$

The inverse translation, which starts from a SCIFF execution trace and produces a corresponding LTL trace, will be denoted by $\mathtt{tm}^{-1}$.

Thanks to the trace mapping function $\mathtt{tm}$, it is possible to evaluate whether the "same" execution trace complies with an LTL and a SCIFF specification: if the two models agree, then they express in some sense "equivalent" prescriptions w.r.t. the trace. Generalizing, if such an agreement is valid for all the possible execution traces, then the two specifications are *behaviorally equivalent*.

Definition 12 (Behavioural equivalence w.r.t. compliance). *A SCIFF specification $\mathcal{S}$ and an LTL formula φ are* behaviorally equivalent w.r.t. compliance *($\varphi \overset{c}{\rightsquigarrow} \mathcal{S}$) if and only if* for each *LTL trace $\mathcal{T}_{\mathcal{L}}$ it holds that:*

$$\textsc{cmp}_{\text{LTL}}\left(\mathcal{T}_{\mathcal{L}}, \varphi\right) \Longleftrightarrow \textsc{cmp}_{\text{SCIFF}}\left(\mathtt{tm}(\mathcal{T}_{\mathcal{L}}), \mathcal{S}\right).$$

We might notice that Definition 12 does not pose any constraint on the SCIFF specification $\mathcal{S}$: indeed, only the trace $\mathcal{T}_{\mathcal{L}}$ is somehow constrained by the application of the mapping function $\mathtt{tm}$.

5 On the Expressiveness of SCIFF

We show now that an arbitrary LTL formula can be expressed in SCIFF by providing an automatic translation procedure from LTL to SCIFF which preserves the compliance equivalence. To this end, we exploit the Separated Normal Form (SNF) for LTL formulae.

5.1 A Separated Normal Form for LTL Formulae

Fisher and colleagues [13] introduced SNF to express an arbitrary LTL formula by adopting a conjunction of three-basic forms, while preserving satisfiability.

Definition 13 (SNF Formula [13]). *An LTL formula φ is in* SNF *iff φ is a conjunction of formulas of the following forms:*

$$\mathbf{start} \implies \bigvee_c l_c \qquad \textit{(an initial LTL-clause)}$$

$$\Box\Big(\bigwedge_a k_a \implies \bigcirc \bigvee_d l_d\Big) \qquad \textit{(a step LTL-clause)}$$

$$\Box\Big(\bigwedge_b k_b \implies \Diamond l\Big) \qquad \textit{(a sometime LTL-clause)}$$

where k_i and l_j are literals (i.e., atomic propositions or negation of atomic propositions) and **start** *is a special symbol true only at the initial time (i.e., whose valuation function is the set $\{0\}$). In this case, we say that φ is an SNF formula.*

Definition 14 (LTL to SNF translation [13]). *$\mathtt{snf}$ is a function which translates an arbitrary LTL formula to a corresponding SNF formula.*

During the transformation, new proposition symbols are introduced to rename complex sub-formulae. Hence, we distinguish between propositions used to represent activities/events, and those used for renaming.

Definition 15 (Proposition symbols, renaming and event sets). *Given an LTL formula φ, $\mathcal{P}(\varphi)$ is the set of proposition symbols contained in φ. Given an SNF formula σ s.t. $\sigma = \mathtt{snf}(\varphi)$, it holds that $\mathcal{P}(\sigma) = \mathcal{E}(\sigma) \cup \mathcal{R}(\sigma)$, where:*

1. event set $\mathcal{E}(\sigma)$ *is the set of atomic propositions contained in the original LTL formula φ, which denote events ($\mathcal{E}(\sigma) = \mathcal{P}(\varphi)$)*
2. renaming set $\mathcal{R}(\sigma)$ *is the set of atomic propositions used for renaming during the transformation.*

Example 2. Let us consider LTL "precedence" formula stating that the send_receipt activity can be executed only after having executed the pay activity:

$$\varphi = \neg send\_receipt \; \mathcal{W} \; pay$$

Hence, $\mathcal{P}(\varphi) = \{pay, send\_receipt\}$. The SNF translation of φ is:

$$\begin{aligned} \sigma = \mathtt{snf}\,[\neg send\_receipt \; \mathcal{W} \; pay] = \\ = \mathbf{start} \Rightarrow \mathbf{x} \wedge \begin{cases} \mathbf{start} \Rightarrow (\neg \mathbf{x} \vee \neg send\_receipt \vee pay) \wedge \\ \mathbf{true} \Rightarrow \bigcirc (\neg \mathbf{x} \vee \neg send\_receipt \vee pay) \wedge \\ \mathbf{start} \Rightarrow (\neg \mathbf{x} \vee \mathbf{y} \vee pay) \wedge \\ \mathbf{true} \Rightarrow \bigcirc (\neg \mathbf{x} \vee \mathbf{y} \vee pay) \wedge \\ \mathbf{y} \Rightarrow \bigcirc (\neg send\_receipt \vee pay) \wedge \\ \mathbf{y} \Rightarrow \bigcirc (\mathbf{y} \vee pay) \end{cases} \end{aligned}$$

Therefore, $\mathcal{R}(\sigma) = \{\mathbf{start}, \mathbf{x}, \mathbf{y}, \mathbf{true}\}$.

5.2 Translation from SNF Formulae to SCIFF

We now provide a syntactic procedure which translates an arbitrary SNF formula to SCIFF, and prove that such a translation preserves compliance.

Definition 16 ($\mathcal{IC}$-mapping). *An $\mathcal{IC}$-mapping $\mathtt{icm}$ is a function which translates an SNF formula to a set of SCIFF integrity constraints, defined as*[5]*:*

[5] Abducible predicates will be represented as **bold** terms.

$$\mathtt{icm}\left[\bigwedge_i \varphi_i\right] \triangleq \bigcup_i \mathtt{icm}[\varphi_i]$$

$$\mathtt{icm}\left[\mathbf{start} \implies \bigvee_c l_c\right] \triangleq \mathtt{icm}[start, 0] \rightarrow \bigvee_c \mathtt{icm}[l_c, 0]\,.$$

$$\mathtt{icm}\left[\Box\left(\bigwedge_a k_a \implies \bigcirc \bigvee_d l_d\right)\right] \triangleq \bigwedge_a \mathtt{icm}[k_a, T] \rightarrow \bigvee_d \left(\mathtt{icm}[l_d, T_2] \wedge T_2 = T + 1\right).$$

$$\mathtt{icm}\left[\Box\left(\bigwedge_a k_a \implies \Diamond l\right)\right] \triangleq \bigwedge_a \mathtt{icm}[k_a, T] \rightarrow \mathtt{icm}[l, T_2] \wedge T_2 \geq T.$$

$$\mathtt{icm}[start, 0] \triangleq \mathbf{occ}(start, 0)$$

$$\mathtt{icm}[true, T] \triangleq \mathbf{true}(T)$$

$$\mathtt{icm}[a, T] \triangleq \mathbf{occ}(a, T)$$

$$\mathtt{icm}[\neg a, T] \triangleq \mathbf{not\_occ}(a, T)$$

Where a stands for a generic propositional symbol. The $\mathcal{IC}$-mapping maps the presence of a certain proposition in a given state onto an abducible $\mathbf{occ}/2$, stating that the proposition *occurs* in that state. Conversely, the absence of the proposition is mapped onto an abducible $\mathbf{not\_occ}/2$.

Definition 17 ($\mathcal{S}$-mapping sm). *Given an SNF formula φ and a set $\mathcal{V} \subseteq \mathcal{P}(\varphi)$ of proposition symbols, the $\mathcal{S}$-*mapping *sm translates φ to a SCIFF specification depending on $\mathcal{V}$. sm is defined as:*

$$\mathtt{sm}: \qquad \varphi, \mathcal{V} \longmapsto \langle \emptyset, \{\mathbf{E}/2, \mathbf{EN}/2, \mathbf{true}/1, \mathbf{occ}/2, \mathbf{not\_occ}/2\}, \mathcal{IC}) \rangle$$

where

$$\begin{aligned}
\mathcal{IC} = \mathtt{icm}(\varphi) \cup \{ & \\
& true \rightarrow \mathbf{occ}(start, 0). && (S)\\
& true \rightarrow \mathbf{true}(0). && (T_1)\\
& \mathbf{true}(T) \rightarrow \mathbf{true}(T_2) \wedge T_2 = T + 1. && (T_2)\\
& \forall p \in \mathcal{P}(\varphi), p \neq start, \mathbf{true}(T) \rightarrow \mathbf{occ}(p, T) \vee \mathbf{not_occ}(p, T). && (2V)\\
& \mathbf{occ}(X, T) \wedge \mathbf{not_occ}(X, T) \rightarrow \bot. && (C)\\
& \mathbf{H}(X, T) \wedge X \in \mathcal{V} \rightarrow \mathbf{occ}(X, T). && (O)\\
& \mathbf{occ}(X, T) \wedge X \in \mathcal{V} \rightarrow \mathbf{E}(X, T). && (E_1)\\
& \mathbf{not_occ}(X, T) \wedge X \in \mathcal{V} \rightarrow \mathbf{EN}(X, T). \ \} && (E_2)
\end{aligned}$$

$\mathcal{S}$-mapping applies $\mathcal{IC}$-mapping and then augments the obtained constraints with further general rules. Such rules capture specific aspects of the LTL semantics:

– (S) translates the special **start** symbol, which is introduced by SNF and is true only at the initial state (i.e., at time point 0).

– (T_1) and (T_2) formalize the LTL *true* atom, which is implicitly subject to the formula $\Box(true)$. To this aim, the **true** abducible is introduced, using an initial rule (T_1) and a recursive rule.
– $(2V)$ and (C) are used to model the two-valued semantics of LTL, i.e., that in each state either a proposition is either true or false. We exclude the symbol "start", which is introduced by Fisher et al. as a special symbol holding only in the initial state.
– (O), (E_1) and (E_2) relate the (not) occurrence of each proposition in each state with the SCIFF concepts of happened events and expectations.

The next theorem states that $\mathtt{sm}$ preserves compliance: an arbitrary SNF formula can be translated to a *behaviourally equivalent* SCIFF specification.

Theorem 1 (SCIFF can express SNF formulae). *Given an SNF formula σ and the SCIFF specification $\mathcal{S} = \mathtt{sm}[\sigma, \mathcal{P}(\sigma)]$, it holds that $\sigma \overset{c}{\rightsquigarrow} \mathcal{S}$.*

Proof. Since LTL and SCIFF share the same semantics for logical symbols AND($\wedge$), OR ($\vee$), and implication($\Rightarrow$ in LTL and $\rightarrow$ in SCIFF), we will focus only on the simplest SNF-forms, consisting of single proposition symbols (instead of conjunctions/disjunctions).

$\sigma = (\mathbf{start} \Rightarrow l)$
If l is a positive literal, say, $l = a$, each compliant LTL execution trace $\mathcal{T}_\mathcal{L}$ must satisfy the property that $a \in \mathcal{T}_\mathcal{L}(0)$, because **start** always holds in state 0. The obtained $\mathcal{S}$ contains the corresponding IC

$$\mathtt{icm}[\mathbf{start} \Rightarrow a] = \mathbf{occ}(start, 0) \rightarrow \mathbf{occ}(a, 0).$$

By taking into account also the two general ICs (S) and (E_1), all abductive explanations of $\mathcal{S}$ must expect a at time point 0, i.e., they must contain $\mathbf{E}(a, 0)$. Therefore, each compliant trace $\mathcal{T}$ must contain $\mathbf{H}(a, 0)$. By considering the trace mapping function $\mathtt{tm}$, this is exactly the same property required for compliant LTL traces, and therefore compliance is preserved by switching from σ to $\mathcal{S}$ or vice-versa. The case in which l is a negative literal, say, $l = \neg a$, can be proven in a similar way.

$\sigma = (k \Rightarrow \bigcirc l)$
Let us consider a first case where both k and l are positive literals, and focus on one side of the equivalence ($\overset{c}{\rightsquigarrow}$); the other side can be proven in a very similar way. To disprove $\overset{c}{\rightsquigarrow}$, one must find an execution trace $\mathcal{T}_\mathcal{L}$ which is compliant with σ, but whose corresponding trace $\mathcal{T}$ is not compliant with $\mathcal{S} = \mathtt{sm}[\sigma, \mathcal{P}(\sigma)]$. Notice that, by Definition 17 (applying (O) and (E_1)), $\mathcal{S}$ explicitly foresees that in case k happens at a time t, then l is expected to happen at time $t_2, t_2 = t + 1$. Hence, to violate $\mathcal{S}$, $\mathcal{T}$ must contain, for a certain time t the event $\mathbf{H}(k, t)$, while $\mathbf{H}(l, t_2) \notin \mathcal{T}$. By applying the $\mathtt{tm}^{-1}$ function on this trace, one obtains a $\mathcal{T}_\mathcal{L}$ which obeys the following properties: (1) $k \in \mathcal{T}_\mathcal{L}(t)$, and (2) $l \notin \mathcal{T}_\mathcal{L}(t + 1)$. The second property in particular implies that $\mathcal{T}_\mathcal{L}$ is not compliant with σ, hence the initial hypothesis does not

hold. The other side of the implication ($\not\rightsquigarrow$) can be proved in the same way, exploiting again the characteristics of the `tm` function. This same proving schema can be applied also to the case where k is a positive literal, and l is a negative literal: the only difference is that $\mathcal{S}$ will contain a negative expectation **EN**, rather than a positive one as before.

Let us now consider the case in which k is a negative literal, say $k = \neg a$, and l is a positive literal, say $l = b$; again, the case in which l is a negative literal can be proven in the same way. Each compliant $\mathcal{T}_\mathcal{L}$ trace must obey the following property: $\forall\ t,\ a \in \mathcal{T}_\mathcal{L}(t) \vee b \in \mathcal{T}_\mathcal{L}(t+1)$. The IC obtained by the application of `icm` is $\mathbf{not\_occ}(a, T) \rightarrow \mathbf{occ}(b, T_2) \wedge T_2 = T + 1$. For each time t, if a happens at time t then rule (O) states that $\mathbf{occ}(a, t)$ is abduced, rule (C) prevents $\mathbf{not\_occ}(a, t)$ to be abduced and thus the IC does not trigger. If, conversely, a does not happen at time t, by rule $(2V)$ we can have two options. In the first, $\mathbf{occ}(a, t)$ is abduced, which imposes that also $\mathbf{E}(a, t)$ is abduced (rule E_1); since a does not happen at time t, this assumption is not fulfilled. In the second, $\mathbf{not\_occ}(a, t)$ is abduced, the IC triggers, abducing $\mathbf{occ}(b, t + 1)$, which in turn triggers (E_1), imposing that b is expected to happen at time $t + 1$. Therefore, each SCIFF compliant execution trace $\mathcal{T}$ must satisfy that $\forall\ t, \mathbf{H}(a, t) \in \mathcal{T} \vee \mathbf{H}(b, t + 1) \in \mathcal{T}$, which is equivalent, under `tm`, to the property on LTL traces.

$\sigma = (k \Rightarrow \Diamond l)$

This case of a simple sometime LTL-clause trivially follows from the discussion made for the previous LTL-clause. The only difference is that the constraint $T_2 = T + 1$ is substituted by $T_2 \geq T$ in this more general case.

Having proven that `sm` preserves compliance for each SNF basic form, we must prove that the translation preserves compliance when applied to a conjunction of these forms. This is straightforward, because a trace complies with a SCIFF specification if *all* the integrity constraints are respected.

5.3 Translation of Arbitrary LTL Formulae to SCIFF

We now demonstrate that also an arbitrary LTL formula can be encoded in SCIFF preserving compliance. The main technical problem is that the SNF translation introduces new symbols (used for renaming complex sub-formulae) which do not represent events. At the SNF level, the distinction between concrete events and renaming symbols gets lost, and therefore the SCIFF specification produced by applying in cascade the SNF and the `sm` translation does not preserve compliance w.r.t. the original LTL formula: positive expectations are imposed also on renaming symbols, which however do not appear in the original LTL formula.

To overcome this issue, the intuitive idea is to restrict the translation `sm` function only to events. The first step is therefore to define, in both settings, a suitable trace projection, which filters an execution trace by maintaining only certain symbols (in particular, the ones which correspond to events).

Definition 18 (SCIFF trace projection). *Given a SCIFF execution trace $\mathcal{T}$ and a set $\mathcal{V}$ of predicate symbols, the* trace projection *of $\mathcal{T}$ on $\mathcal{V}$ ($\mathcal{T}|_\mathcal{V}$) is the*

subset of $\mathcal{T}$ containing only events taken from $\mathcal{V}$:

$$\mathcal{T}|_{\mathcal{V}} \triangleq \{\mathbf{H}(e,t) \mid \mathbf{H}(e,t) \in \mathcal{T} \wedge e \in \mathcal{V}\}$$

Definition 19 (LTL trace projection). *Given an LTL execution trace $\mathcal{T}_{\mathcal{L}} = (\mathbb{N}, <, v_{occ})$ and a set $\mathcal{V}$ of proposition symbols, the* trace projection *of $\mathcal{T}_{\mathcal{L}}$ on $\mathcal{V}$ ($\mathcal{T}_{\mathcal{L}}|_{\mathcal{V}}$) is the projection of $\mathcal{T}_{\mathcal{L}}$ containing only events taken from $\mathcal{V}$:*

$$\mathcal{T}_{\mathcal{L}}|_{\mathcal{V}} = (\mathbb{N}, <, {v_{occ}}') \text{ s.t. } {v_{occ}}'(e) \triangleq \begin{cases} v_{occ}(e) & \text{if } e \in \mathcal{V}; \\ \emptyset & \text{otherwise.} \end{cases}$$

Lemma 1 (Commutativity between trace projection and trace mapping). *For each LTL execution trace $\mathcal{T}_{\mathcal{L}}$ and for each set of proposition symbols $\mathcal{V}$*

$$\mathtt{tm}[\mathcal{T}_{\mathcal{L}}|_{\mathcal{V}}] = \mathtt{tm}[\mathcal{T}_{\mathcal{L}}]\,|_{\mathcal{V}}$$

Proof. From the definitions of trace mapping (Def. 11) and of trace projection (Def. 18 and 19).

We now briefly recall one of the main results presented in [13], which proves that SNF preserves satisfiability, i.e., in our setting, that it preserves compliance. Lemma 2 reviews the satisfiability result by explicitly taking into account execution traces. In particular, it states that execution traces compliant respectively with an LTL formula and its corresponding SNF are exactly the same if we restrict the comparison only to concrete events.

Theorem 2 (SNF preserves satisfiability [13]). *An LTL formula φ is satisfiable iff $\mathtt{snf}(\varphi)$ is satisfiable.*

Lemma 2 (Compliance preservation via extended traces, adapted from [13]). *For each LTL formula φ, it holds that*

$$\forall\ \mathcal{T}_{\mathcal{L}}\ \text{CMP}_{\text{LTL}}\left(\mathcal{T}_{\mathcal{L}}, \mathtt{snf}[\varphi]\right) \Longrightarrow \text{CMP}_{\text{LTL}}\left(\mathcal{T}_{\mathcal{L}}|_{\mathcal{E}(\mathtt{snf}[\varphi])}, \varphi\right)$$

$$\forall\ \mathcal{T}_{\mathcal{L}}\ \text{CMP}_{\text{LTL}}\left(\mathcal{T}_{\mathcal{L}}, \varphi\right) \Longrightarrow \exists \mathcal{T}'_{\mathcal{L}} \text{ s.t. } \mathcal{T}_{\mathcal{L}} = \mathcal{T}'_{\mathcal{L}}|_{\mathcal{E}(\mathtt{snf}[\varphi])} \wedge \text{CMP}_{\text{LTL}}\left(\mathcal{T}'_{\mathcal{L}}, \mathtt{snf}[\varphi]\right)$$

where we remember that (by Definition 15) $\mathcal{E}\left(\mathtt{snf}\left[\varphi\right]\right) = \mathcal{P}(\varphi)$. With such preliminaries, it is possible to prove that each LTL formula is translatable to a SCIFF specification, preserving compliance.

Theorem 3 (SCIFF can express LTL). *Given an arbitrary LTL formula φ and the SCIFF specification $\mathcal{S} = \mathtt{sm}[\mathtt{snf}[\varphi], \mathcal{P}(\varphi)]$, it holds that $\mathcal{S} \overset{c}{\rightsquigarrow} \varphi$.*

Proof. Let us denote $\sigma = \mathtt{snf}\,[\varphi]$. From Def. 12, and by remembering that the event set of σ contains all the proposition symbols of φ ($\mathcal{P}(\varphi) = \mathcal{E}(\sigma)$), one has to prove that

$$\forall \mathcal{T}_{\mathcal{L}},\ \text{CMP}_{\text{LTL}}(\mathcal{T}_{\mathcal{L}}, \varphi) \Longleftrightarrow \text{CMP}_{\text{SCIFF}}\left(\mathtt{tm}\left[\mathcal{T}_{\mathcal{L}}\right], \mathtt{sm}\left[\sigma, \mathcal{E}(\sigma)\right]\right)$$

We will prove firstly one way of the implication ($\Longrightarrow$), and then the opposite direction ($\Longleftarrow$). Both the proofs are organized in the same way: by applying the results obtained in Lemma 1, Lemma 2, and Theorem 1, the problem of proving a formula is reduced to prove another, simpler formula. Hence, each proof starts with a diagram that shows how each previous result is applied to a formula, and then the simpler formula is proved.
($\Longrightarrow$) Let us consider the following schema:

$$\begin{array}{ccc} \forall\, \mathcal{T}_{\mathcal{L}},\ \textsc{cmp}_{\text{LTL}}(\mathcal{T}_{\mathcal{L}}, \varphi) & \xRightarrow{(*)} & \textsc{cmp}_{\text{SCIFF}}\left(\mathtt{tm}\left[\mathcal{T}_{\mathcal{L}}\right], \mathtt{sm}\left[\sigma, \mathcal{E}(\sigma)\right]\right) \\ \Big\Downarrow \text{Lemma 2} & & \Big\Uparrow (\dagger) \\ \exists \mathcal{T}'_{\mathcal{L}},\ \mathcal{T}_{\mathcal{L}} = \mathcal{T}'_{\mathcal{L}}|_{\mathcal{E}(\sigma)} \wedge \textsc{cmp}_{\text{LTL}}(\mathcal{T}'_{\mathcal{L}}, \sigma) & \xRightarrow{\text{Theorem 1}} & \textsc{cmp}_{\text{SCIFF}}\left(\mathtt{tm}\left[\mathcal{T}'_{\mathcal{L}}\right], \mathtt{sm}\left[\sigma, \mathcal{P}(\sigma)\right]\right) \end{array}$$

The schema shows that proving $(*)$ reduces to prove $(\dagger)$, i.e., we prove that

$$\textsc{cmp}_{\text{SCIFF}}\left(\mathtt{tm}\left[\mathcal{T}'_{\mathcal{L}}\right], \mathtt{sm}\left[\sigma, \mathcal{P}(\sigma)\right]\right) \Longrightarrow \textsc{cmp}_{\text{SCIFF}}\left(\mathtt{tm}\left[\mathcal{T}'_{\mathcal{L}}|_{\mathcal{E}(\sigma)}\right], \mathtt{sm}\left[\sigma, \mathcal{E}(\sigma)\right]\right) \quad (\dagger)$$

By taking into account abducible sets, Def. 15 and Lemma 1, $(\dagger)$ becomes:

$$\textsc{cmp}_{\text{SCIFF}}{}^{\Delta}\left(\mathcal{T}, \mathcal{S}^{\mathcal{ER}}\right) \Longrightarrow \textsc{cmp}_{\text{SCIFF}}{}^{\Delta'}\left(\mathcal{T}|_{\mathcal{E}(\sigma)}, \mathcal{S}^{\mathcal{E}}\right) \quad (\ddagger)$$

where $\mathcal{S}^{\mathcal{ER}} = \mathtt{sm}\left[\sigma, \mathcal{E}(\sigma) \cup \mathcal{R}(\sigma)\right]$, $\mathcal{S}^{\mathcal{E}} = \mathtt{sm}\left[\sigma, \mathcal{E}(\sigma)\right]$ and $\mathcal{T} = \mathtt{tm}\left[\mathcal{T}'_{\mathcal{L}}\right]$. To prove $(\ddagger)$, we demonstrate that

$$\Delta' = \Delta \setminus \{\mathbf{E}(e,t) | e \in \mathcal{R}(\sigma)\} \setminus \{\mathbf{EN}(e,t) | e \in \mathcal{R}(\sigma)\}$$

obeys the three properties required by the Definition 10 of SCIFF compliance:

1. Δ' is an abductive explanation for $\mathcal{S}^{\mathcal{E}}_{\mathcal{T}|_{\mathcal{E}(\sigma)}}$. The only difference between $\mathcal{S}^{\mathcal{E}}$ and $\mathcal{S}^{\mathcal{ER}}$ is that, for the first specification, rules (O), (E_1) and (E_2) of Def. 17 do not trigger for events outside $\mathcal{E}(\sigma)$ (in particular, they do not trigger for events inside $\mathcal{R}(\sigma)$). From Remark 1, Δ is therefore a suitable abductive explanation for $\mathcal{S}^{\mathcal{E}}$ too. Furthermore, being (E_1) and (E_2) the only constraints involving positive and negative expectations concerning elements in $\mathcal{R}(\sigma)$, it is not required for an abductive explanation to contain them anymore.
2. Δ' is $\mathbf{E}$-consistent, because $\Delta' \subseteq \Delta$ and Δ is $\mathbf{E}$-consistent.
3. Δ' is $\mathcal{T}|_{\mathcal{E}(\sigma)}$-fulfilled. Since $\mathcal{T}|_{\mathcal{E}(\sigma)}$ is a projection of $\mathcal{T}$, $\Delta' \subseteq \Delta$ and Δ is $\mathcal{T}$-fulfilled, no negative expectation in Δ' can be violated by $\mathcal{T}|_{\mathcal{E}(\sigma)}$. Positive expectations concerning elements in $\mathcal{E}(\sigma)$ are maintained in Δ', and so are the corresponding happened events after the trace projection. Positive expectations concerning elements in $\mathcal{R}(\sigma)$ are removed from Δ when obtaining Δ', and therefore the application of the trace projection, which rules out happened events concerning elements in $\mathcal{R}(\sigma)$, does not affect fulfillment.

($\Longleftarrow$) We move then to prove the other way of the double implication stated in this theorem. Again, let us consider the following schema:

$$\begin{array}{ccc}
\textsc{cmp}_{\text{LTL}}\left(\mathtt{tm}^{-1}\left[\mathcal{T}\right],\varphi\right) & \xleftarrow{(**)} & \forall\ \mathcal{T},\ \textsc{cmp}_{\text{SCIFF}}\left(\mathcal{T},\mathtt{sm}\left[\sigma,\mathcal{E}(\sigma)\right]\right) \\
\Uparrow\ \text{Lemma 2, then Lemma 1} & & \Downarrow(\S) \\
\textsc{cmp}_{\text{LTL}}\left(\mathtt{tm}^{-1}\left[\mathcal{T}'\right],\sigma\right) & \xleftarrow{\text{Theorem 1}} & \begin{array}{c}\exists\ \mathcal{T}',\ \mathcal{T}=\mathcal{T}'|_{\mathcal{E}(\sigma)} \\ \wedge\textsc{cmp}_{\text{SCIFF}}\left(\mathcal{T}',\mathtt{sm}\left[\sigma,\mathcal{P}(\sigma)\right]\right)\end{array}
\end{array}$$

The schema shows that proving $(**)$ reduces to prove $(\S)$, i.e. we prove that

$$\forall\ \mathcal{T},\ \textsc{cmp}_{\text{SCIFF}}{}^{\Delta}\left(\mathcal{T},\mathcal{S}^{\mathcal{E}}\right)\Longrightarrow\exists\ \mathcal{T}',\ \mathcal{T}=\mathcal{T}'|_{\mathcal{E}(\sigma)}\wedge\textsc{cmp}_{\text{SCIFF}}{}^{\Delta'}\left(\mathcal{T}',\mathcal{S}^{\mathcal{ER}}\right)\quad(\S)$$

where $\mathcal{S}^{\mathcal{ER}}=\mathtt{sm}\left[\sigma,\mathcal{E}(\sigma)\cup\mathcal{R}(\sigma)\right]$ and $\mathcal{S}^{\mathcal{E}}=\mathtt{sm}\left[\sigma,\mathcal{E}(\sigma)\right]$.

First of all, it is worth noting that $\mathcal{S}^{\mathcal{ER}}$ extends $\mathcal{S}^{\mathcal{E}}$ by imposing that rules (O), (E_1) and (E_2) can be also triggered by **occ**/**not_occ** abducibles involving symbols in $\mathcal{R}(\sigma)$, generating a larger set of expectations. Since $\mathcal{T}'\supseteq\mathcal{T}$, an abductive explanation Δ' can be therefore found for $\mathcal{S}^{\mathcal{ER}}$ by extending Δ with the new generated expectations: $\Delta'=\Delta\cup\Delta^{\mathbf{E}}_{\mathcal{R}}\cup\Delta^{\mathbf{EN}}_{\mathcal{R}}$, where $\Delta^{\mathbf{E}}_{\mathcal{R}}$ and $\Delta^{\mathbf{EN}}_{\mathcal{R}}$ respectively represent the inserted positive and negative expectations.

Δ' is **E**-consistent. Indeed, since $\Delta^{\mathbf{E}}_{\mathcal{R}}$ and $\Delta^{\mathbf{EN}}_{\mathcal{R}}$ contain only expectations generated by rules (E_1) and (E_2), by construction we have:

$$\begin{array}{l}\forall\ \mathbf{E}(a,t),\qquad \mathbf{E}(a,t)\in\Delta^{\mathbf{E}}_{\mathcal{R}}\Rightarrow\mathbf{occ}(a,t)\in\Delta' \\ \forall\ \mathbf{EN}(a,t),\ \mathbf{EN}(a,t)\in\Delta^{\mathbf{EN}}_{\mathcal{R}}\Rightarrow\mathbf{not\_occ}(a,t)\in\Delta'\end{array}\quad(\S\S)$$

Let us suppose by absurdum that there exist a, t (with $a\in\mathcal{R}(\sigma)$) s.t. $\mathbf{E}(a,t)\in\Delta^{\mathbf{E}}_{\mathcal{R}}$ and $\mathbf{EN}(a,t)\in\Delta^{\mathbf{EN}}_{\mathcal{R}}$. In this case, $(\S\S)$ would state that $\mathbf{occ}(a,t)\in\Delta'$ and $\mathbf{not\_occ}(a,t)\in\Delta'$. This would violate rule (C), making impossible that Δ' is an abductive explanation.

An execution trace $\mathcal{T}^*$ compliant with $\mathcal{S}^{\mathcal{ER}}$ can be therefore built as follows:

$$\mathcal{T}^*=\mathcal{T}\cup\mathcal{T}^{\mathcal{R}},\text{ where }\mathbf{H}(a,t)\in\mathcal{T}^{\mathcal{R}}\Leftrightarrow\mathbf{E}(a,t)\in\Delta^{\mathbf{E}}_{\mathcal{R}}$$

Under this choice:

1. Δ' is left untouched by $\mathcal{T}^*$. Indeed, the only impact of $\mathcal{T}^{\mathcal{R}}$ on the ICs of $\mathcal{S}^{\mathcal{ER}}$ is to trigger rule (O), generating corresponding **occ** abducibles. However, from $(\S\S)$ we know that all these abducibles are already contained in Δ'.
2. Δ' is $\mathcal{T}^*$-fulfilled by construction.
3. $\mathcal{T}^*|_{\mathcal{E}(\sigma)}=\mathcal{T}$, because all the happened events contained in $\mathcal{T}^{\mathcal{R}}$ involve symbols belonging to $\mathcal{R}(\sigma)$, and are therefore ruled out by applying the projection.

6 Discussion and Conclusion

In this work we compare the framework SCIFF with the widely adopted LTL, from the viewpoint of the compliance verification task. To this end, we have proposed a formal notion of compliance for each one of the approaches, and defined the equivalence of the two notions. Then we provide an automatic translation between LTL-based and SCIFF-based specifications. We prove that such translation preserves the notion of compliance, w.r.t. the defined equivalence.

LTL-based techniques for verification have a number of strengths and weaknesses, as well as the SCIFF framework that inherits advantages and limits of LP approaches. An important result of this work is to better clarify the links between the two techniques: this opens up to the possibility of an integrated approach based on Computational Logic, where the best of both worlds (LTL and SCIFF) can be coherently exploited. E.g., LP-based approaches support the use of variables and constraints over them, allowing to model systems where also data and data relations/constraints are taken into account.

A limit of the presented approach stems from the semi-decidability issues of (refutation-based) logic programming. SCIFF inherits such characteristics: as a consequence, not any possible LTL specification could be directly reasoned about in SCIFF. From an operational viewpoint the problem can be avoided by restricting to a significant fragment of LTL, and provide ad-hoc translations for which termination is guaranteed. Alternatively, it is possible to notice that a number of applications inherently require finite traces, like e.g. business processes [23], that are developed to reach a business goal (such as delivery a product) in a finite number of steps. Automatic translation of any LTL formula within a finite trace semantics into a SCIFF corresponding model is matter of ongoing work.

References

1. van der Aalst, W., de Beer, H., van Dongen, B.: Process Mining and Verification of Properties: An Approach based on Temporal Logic. In: Meersman, R., Tari, Z. (eds.) Proceedings of the OTM 2005 Confederated International Conferences CoopIS, DOA, and ODBASE. LNCS, vol. 3760, pp. 130–147. Springer (2005)
2. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. ACM Transactions on Computational Logic 9(4), 29:1–29:43 (Aug 2008)
3. Alberti, M., Gavanelli, M., Lamma, E., Chesani, F., Mello, P., Montali, M.: An abductive framework for a-priori verification of web services. In: Bossi, A., Maher, M.J. (eds.) Procs. of PPDP 2006, July 10-12, 2006, Venice, Italy. pp. 39–50. ACM, New York, USA (2006)
4. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P., Sartor, G.: Mapping deontic operators to abductive expectations. Computational & Mathematical Organization Theory 12(2-3), 205–225 (2006)
5. Artikis, A., Sergot, M., Pitt, J.: Specifying Norm-Governed Computational Societies. ACM Transactions on Computational Logic 10(1), 1–42 (2009)
6. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) 6th Intl. Conf. BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer (2008)

7. Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Reasoning about interaction protocols for customizing web service selection and composition. Journal of Logic and Algebraic Programming 70(1), 53–73 (2007)
8. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment Tracking via the Reactive Event Calculus. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 91–96 (2009)
9. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: Nusmv 2: An opensource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV. Lecture Notes in Computer Science, vol. 2404, pp. 359–364. Springer (2002)
10. Clark, K.L.: Negation as Failure. In: Gallaire, H., Minker, J. (eds.) Logic and Data Bases, pp. 293–322. Plenum Press (1978)
11. De Nicola, A., Missikoff, M., Proietti, M., Smith, F.: A logic-based method for business process knowledge base management. In: Bergamaschi, S., Lodi, S., Martoglia, R., Sartori, C. (eds.) 8th Italian Symposium on Advanced Database Systems. pp. 170–181. Rimini, Italy (2010)
12. Emerson, E.A.: Temporal and Modal Logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics. Elsevier and MIT Press (1990)
13. Fisher, M., Dixon, C., Peim, M.: Clausal Temporal Resolution. ACM Transactions on Computational Logic 2(1), 12–56 (2001)
14. Fornara, N., Colombetti, M.: Specifying artificial institutions in the event calculus. In: Dignum, V. (ed.) Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. pp. 335–366. IGI Global (2009)
15. Holzmann, G.: The model checker spin. Software Engineering, IEEE Transactions on 23(5), 279–295 (1997)
16. Jaffar, J., Maher, M.J., Marriott, K., Stuckey, P.J.: The semantics of constraint logic programs. J. Log. Program. 37(1-3), 1–46 (1998)
17. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive Logic Programming. Journal of Logic and Computation 2(6), 719–770 (1993)
18. Kunen, K.: Negation in logic programming. J. Log. Program. 4(4), 289–308 (1987)
19. Lloyd, J.W.: Foundations of Logic Programming. Springer, 2nd edn. (1987)
20. Montali, M.: Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach, LNBIP, vol. 56. Springer (2010)
21. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. ACM Transactions on the Web 4(1) (2010)
22. Montali, M., Torroni, P., Chesani, F., Mello, P., Alberti, M., Lamma, E.: Abductive logic programming as an effective technology for the static verification of declarative business processes. Fundamenta Informaticae 102(3-4), 325–361 (2010)
23. Pesic, M., van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In: Eder, J., Dustdar, S. (eds.) Procs. of the BPM 2006 Workshops. LNCS, vol. 4103, pp. 169–180. Springer (2006)
24. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: Full Support for Loosely-Structured Processes. In: Procs. IEEE EDOC 2007. pp. 287–300. IEEE Computer Society (2007)
25. Roman, D., Kifer, M.: Semantic Web Service Choreography: Contracting and Enactment. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T.W., Thirunarayan, K. (eds.) Procs. ISWC 2008. LNCS, vol. 5318, pp. 550–566. Springer (2008)

Towards a tableau-based procedure for PLTL based on a multi-conclusion rule and logical optimizations

Mauro Ferrari[1], Camillo Fiorentini[2], Guido Fiorino[3]

[1] DiSTA, Univ. degli Studi dell'Insubria, Via Mazzini, 5, 21100, Varese, Italy
[2] DI, Univ. degli Studi di Milano, Via Comelico, 39, 20135 Milano, Italy
[3] DISCO, Univ. degli Studi di Milano-Bicocca, Viale Sarca, 336, 20126, Milano, Italy

Abstract. We present an ongoing work on a proof-search procedure for Propositional Linear Temporal Logic (PLTL) based on a one-pass tableau calculus with a multiple-conclusion rule. The procedure exploits logical optimization rules to reduce the proof-search space. We also discuss the performances of a Prolog prototype of our procedure.

1 Introduction

In recent years, we have introduced new tableau calculi and logical optimization rules for propositional Intuitionistic Logic [4] and propositional Gödel-Dummett Logic [7]. As an application of these results, we have implemented theorem provers for these logics [3, 7] which outperform their competitors. The above quoted calculi and optimizations are the result of a deep analysis of the Kripke semantics of the logic at hand. In this paper, we apply such a semantical analysis to PLTL. In particular, we present a refutation tableau calculus and some logical optimizations for PLTL and we briefly discuss a prototype Prolog implementation of the resulting proof-search procedure.

As for related work, our tableau calculus lies in the line of the one-pass calculi based on sequents and tableaux calculi [14, 2, 10], whose features are suitable for automated deduction. We also cite as related the approaches based on sequent calculi discussed in [12, 13] and the natural deduction based proof-search techniques discussed in [1]. The results in [8, 15] are based on resolution, thus they are related less to our approach.

2 Tableau calculus and replacement rules

We consider the language based on a denumerable set of propositional variables $\mathcal{V}$, the logical constants $\top$ (true), $\bot$ (false), $\neg$ and $\vee$ and the modal operators $\circ$ (next) and $\mathbf{U}$ (until). We define $\Box A$ as $\neg(\top \mathbf{U} \neg A)$. Given a set of formulas $\mathcal{S}$, we denote with $\circ S$ the set $\{\circ A \mid A \in \mathcal{S}\}$.

PLTL is semantically characterized by rooted linearly ordered Kripke models; formally, a PLTL-*model* is a structure $\mathcal{K} = \langle P, \leq, \rho, V\rangle$ where P is the set of

$$\frac{\mathcal{S}, A\mathbf{U}B}{\mathcal{S}, B \mid \mathcal{S}, A, \neg B, \circ(A\mathbf{U}B)}\mathbf{U} \qquad \frac{\mathcal{S}, \neg(A\mathbf{U}B)}{\mathcal{S}, \Box\neg B \mid \mathcal{S}, \neg A, \neg B \mid \mathcal{S}, A, \neg B, \circ\neg(A\mathbf{U}B)}\neg\mathbf{U}$$

$$\frac{\mathcal{S}, \neg \circ A}{\mathcal{S}, \circ\neg A}\neg\circ \quad \frac{\mathcal{S}, A \vee B}{\mathcal{S}, A \mid \mathcal{S}, B}\vee \quad \frac{\mathcal{S}, \neg(A \vee B)}{\mathcal{S}, \neg A, \neg B}\neg\vee \quad \frac{\mathcal{S}, \neg\neg A}{\mathcal{S}, A}\neg\neg \quad \frac{\mathcal{S}, \neg\top}{\mathcal{S}, \bot}\neg\top \quad \frac{\mathcal{S}, \neg\bot}{\mathcal{S}, \top}\neg\bot$$

$$\frac{\mathcal{T}, \circ\mathcal{A}, \circ\mathcal{B}}{\mathcal{A}, \mathcal{B}^+, \circ\mathcal{B} \mid \mathcal{A}, \mathcal{H}_1 \mid \ \ldots \ \mid \mathcal{A}, \mathcal{H}_m}\mathrm{Lin}$$

$\mathcal{T} \subseteq \mathcal{V} \cup \{\neg p \mid p \in \mathcal{V}\} \cup \{\top, \bot\}$, $\mathcal{A}$ is a possibly empty set,
$\mathcal{B} = \{U_1, \ldots, U_m\}$ is a possibly empty set, with $U_i = A_i\mathbf{U}B_i$ or $U_i = \neg(A_i\mathbf{U}B_i)$
$\mathcal{B}^+ = \{U_i^+ | U_i \in \mathcal{B}\}$, where

$$\begin{array}{rclcrcl} (A\mathbf{U}B)^+ & = & A & & (A\mathbf{U}B)^- & = & B \\ (\neg(A\mathbf{U}B))^+ & = & \neg B & & (\neg(A\mathbf{U}B))^- & = & \neg A, \neg B \end{array}$$

$$\mathcal{H}_i \ = \ \{\circ U_1, U_1^+, \ldots, \circ U_{i-1}, U_{i-1}^+\} \cup \{U_i^-\} \cup \{U_{i+1}, \ldots, U_m\} \ (i = 1, \ldots, m)$$

Fig. 1. The tableau calculus for PLTL

worlds, $\leq$ is a linear well-order with minimum ρ and no maximum element, V is a function associating with every world $\alpha \in P$ the set of propositional variables satisfied in α. Given $\alpha \in P$, the *immediate successor* of α, denoted by α', is the minimum of the $<$-successors of α. The satisfiability of a formula A in a world α of $\mathcal{K}$, written $\mathcal{K}, \alpha \Vdash A$ (or simply $\alpha \Vdash A$), is defined as follows:

- for $p \in \mathcal{V}$, $\alpha \Vdash p$ iff $p \in V(\alpha)$; $\alpha \Vdash \top$; $\alpha \nVdash \bot$;
- $\alpha \Vdash \neg A$ iff $\alpha \nVdash A$; $\alpha \Vdash A \vee B$ iff $\alpha \Vdash A$ or $\alpha \Vdash B$;
- $\alpha \Vdash \circ A$ iff $\alpha' \Vdash A$;
- $\alpha \Vdash A\mathbf{U}B$ iff $\exists \beta \geq \alpha : \beta \Vdash B$ and $\forall \gamma : \alpha \leq \gamma < \beta$, $\gamma \Vdash A$.

The following properties can be easily proved. The latter one follows by the fact that $\leq$ is a well-order, hence, if B is satisfiable in some $\gamma \geq \alpha$, there exists the minimum $\gamma^* \geq \alpha$ satisfying B.

- $\alpha \Vdash \Box A$ iff $\forall \beta \geq \alpha, \beta \Vdash A$;
- $\alpha \nVdash A\mathbf{U}B$ iff $(\forall \gamma \geq \alpha, \gamma \nVdash B)$ or $(\exists \beta \geq \alpha : \beta \nVdash A$ and $\forall \gamma : \alpha \leq \gamma \leq \beta, \gamma \nVdash B)$.

A set of formulas $\mathcal{S}$ is *satisfiable in* α ($\mathcal{K}, \alpha \Vdash \mathcal{S}$) if every formula of $\mathcal{S}$ is satisfiable in α; $\mathcal{S}$ *is satisfiable* if it is satisfiable in some world of a PLTL-model. The rules of the tableau calculus $\mathbf{T}$ for PLTL are given in Fig. 1. The peculiar rule of $\mathbf{T}$ is the rule Lin inspired by the multiple-conclusion rule for Gödel-Dummett Logic DUM presented in [6, 7]. DUM is semantically characterized by intuitionistic linearly ordered Kripke models; the multi-conclusion rule for DUM simultaneously treats a set of implicative formulas while Lin simultaneously treats a set of modal formulas. We remark, that the number of conclusions of rule Lin depends on the number of formulas in $\mathcal{B}$; if $\mathcal{B}$ is empty, Lin has $\mathcal{A}$ as only conclusion.

The rules of $\mathbf{T}$ are sound is the sense that, if the premise of a rule is satisfiable then one of its conclusions is satisfiable. We briefly discuss, by means

$$\frac{\mathcal{S}, \Box A}{\mathcal{S}[\top/A], \Box A}\text{R-}\Box \qquad \frac{\mathcal{S}, \Box\neg A}{\mathcal{S}[\bot/A], \Box\neg A}\text{R-}\Box\neg$$

$$\frac{\mathcal{S}, A}{\mathcal{S}\{\top/A\}, A}\text{R-}cl \qquad \frac{\mathcal{S}, \neg A}{\mathcal{S}\{\bot/A\}, \neg A}\text{R-}cl\neg$$

$$\frac{\mathcal{S}}{\mathcal{S}[\top/p]}\preceq^+ \text{ if } p\preceq^+ \mathcal{S} \qquad \frac{\mathcal{S}}{\mathcal{S}[\bot/p]}\preceq^- \text{ if } p\preceq^- \mathcal{S}$$

For for $l \in \{+,-\}$, $p\preceq^l \mathcal{S}$ iff $p\preceq^l H$ for every $H \in \mathcal{S}$ where, $p\preceq^l H$ is defined as follows:

- $p\preceq^+ p$ and $p\preceq^- \neg p$ and $p\preceq^l H$, if $H \in (\mathcal{V}\setminus\{p\})\cup\{\top,\bot\}$;
- $p\preceq^l (A\vee B)$ iff $p\preceq^l A$ and $p\preceq^l B$;
- $p\preceq^l (A\mathbf{U}B)$ iff $p\preceq^l A$ and p does not occur in B;
- $p\preceq^l \neg(A\mathbf{U}B)$ iff $p\preceq^l B$ and p does not occur in A;
- if $A \neq B\mathbf{U}C$, then $p\preceq^+ \neg A$ iff $p\preceq^- A$ and $p\preceq^- \neg A$ iff $p\preceq^+ A$;
- $p\preceq^l \circ A$ iff $p\preceq^l A$;

Fig. 2. Optimization rules for PLTL

of an example, the soundness of rule Lin. The application of rule Lin to $\circ\mathcal{B} = \{\circ(A_1\mathbf{U}B_1), \circ\neg(A_2\mathbf{U}B_2)\}$ generates as conclusions the sets:

$$\mathcal{C} = \{A_1, \neg B_2\} \cup \circ\mathcal{B}\,,\ \mathcal{H}_1 = \{B_1, \neg(A_2\mathbf{U}B_2)\}\,,\ \mathcal{H}_2 = \{\circ(A_1\mathbf{U}B_1), A_1, \neg A_2, \neg B_2\}.$$

Let us assume that $\alpha \Vdash \circ\mathcal{B}$; we show that at least one of the conclusions is satisfiable. We have $\alpha' \Vdash A_1\mathbf{U}B_1$ and $\alpha' \Vdash \neg(A_2\mathbf{U}B_2)$. Note that:

- $\alpha' \Vdash A_1\mathbf{U}B_1 \ \Rightarrow\ \exists\beta_1 \geq \alpha' : \beta_1 \Vdash B_1$ and $\forall\gamma : \alpha' \leq \gamma < \beta_1, \gamma \Vdash A_1$.
- $\alpha' \Vdash \neg(A_2\mathbf{U}B_2) \ \Rightarrow\ \begin{cases} \text{(i) } \forall\gamma \geq \alpha', \gamma \nVdash B_2 \text{ or} \\ \text{(ii) } \exists\beta_2 \geq \alpha' : \beta_2 \nVdash A_2 \text{ and } \forall\gamma : \alpha' \leq \gamma \leq \beta_2, \gamma \nVdash B_2 \end{cases}$

If (i) holds either $\alpha' < \beta_1$ and $\alpha' \Vdash \mathcal{C}$, or $\alpha' = \beta_1$ and $\alpha' \Vdash \mathcal{H}_1$. Now, let us suppose that (ii) holds; then:

- if $\alpha' < \beta_1$ and $\alpha' < \beta_2$, then $\alpha' \Vdash \mathcal{C}$;
- if $\alpha' = \beta_1$, then $\alpha' \Vdash \mathcal{H}_1$;
- if $\alpha' < \beta_1$ and $\alpha' = \beta_2$, then $\alpha' \Vdash \mathcal{H}_2$.

The notions of *proof-table* and *branch* are defined as usual. A set $\mathcal{S}$ of formulas is *closed* if it either contains $\bot$ or it contains a formula A and its negation. Branches of a proof-table are generated alternating *saturation phases*, where rules different from Lin are applied as long as possible, and applications of rule Lin. We remark that, at the end of a saturation phase, we get a set of formulas which only contains literals, $\top$, $\bot$ and formulas prefixed with $\circ$. If during the saturation phase a closed set is generated the construction of the branch is aborted. During branch construction loops can be generated, hence a loop-checking mechanism is needed to assure termination. A *loop* is a sequence of consecutive sets of formulas $\mathcal{S}_1, \ldots, \mathcal{S}_n$ in a branch such that $\mathcal{S}_1 = \mathcal{S}_n$ and $\mathcal{S}_i \neq \mathcal{S}_{i+1}$ for every $1 \leq i < n$. Whenever, during a branch construction, a loop is detected the

branch construction is aborted. A loop is *closed* if there exist $i \in \{1, \ldots, n-1\}$ and $A\mathbf{U}B \in \mathcal{S}_i$ ($\neg(A\mathbf{U}B) \in \mathcal{S}_i$, respectively) such that $B \notin \mathcal{S}_j$ ($\{\neg A, \neg B\} \not\subseteq \mathcal{S}_j$, respectively) for every $1 \leq j < n$. A loop is *open* if it is not closed. A branch is *closed* if it contains a closed set of formulas or a closed loop and *open* otherwise. The proof of the completeness theorem for $\mathbf{T}$ is based on a procedure extracting a PLTL-model satisfying $\mathcal{S}$ from an open branch starting with $\mathcal{S}$.

Although multi-conclusion rules as Lin can generate a huge number of branches, as discussed in [7], theorem provers using these kind of rules can be effective.

To improve the performances of the above procedure we exploit the optimization rules depicted in Fig. 2 which are inspired by the rules presented in [11, 4]. In rules R-$\Box$ and R-$\Box\neg$ (R stands for Replacement), $\mathcal{S}[B/A]$ denotes formula substitution, that is the set of formulas obtained by replacing every occurrence of A in S with B. In rules R-*cl* and R-*cl*$\neg$, $\mathcal{S}\{B/A\}$ denotes *partial* formula substitution, that is the set of formulas obtained by replacing every occurrence of A in $\mathcal{S}$ which is not under the scope of a modal connective with B. As for rules $\preceq^+$ and $\preceq^-$, they can be applied if the propositional variable p has *constant polarity in* $\mathcal{S}$ ($p \preceq^l \mathcal{S}$). We remark that rules $\preceq^+$ and $\preceq^-$ are the PLTL version of the rules $\mathbf{T}$-permanence and $\mathbf{T}\neg$-permanence of [4].

All the rules of Fig. 2 have the property that the premise is satisfiable iff the conclusion is. In the proof search procedure we apply the optimization rules and the usual boolean simplification rules [11, 4] at every step of a saturation phase.

3 Implementations and performances

To perform some experiments on the benchmark formulas for PLTL, we have implemented β, a theorem prover written in Prolog[4]. At present β is a very simple prototype that implements $\mathbf{T}$ and the rules in Fig. 2. On the third column of the table in Fig. 4 we report the performances of β. For every family of formulas in the benchmark we indicate the number of formulas of the family solved within one minute. All tests were conducted on a machine with a 2.7GHz Intel Core i7 CPU with 8GB memory. All the optimizations rules we have described are effective in speeding-up the deduction. Indeed, without the described optimizations, timings of β are some order of magnitude greater and almost no formula is decided within one minute. In the fourth column of Fig. 4 we report the figures related to `PLTL`, an OCaml prover based on the one-pass tableau calculus of [14]. Although in general `PLTL` outperforms β, there are families where our prototype is more efficient than `PLTL` and this is encouraging for further research.

To conclude, we have presented our ongoing research on automated deduction for PLTL. In this note we have discussed a new proof-theoretical characterization of PLTL based on a multiple-conclusion rule and some optimization rules useful to cut the size of the proofs. As regards the future work, we aim to apply to the case of PLTL other optimizations introduced in the context of Intuitionistic logic as the permanence rules of [4] and the optimizations exploiting the notions of *local formula* [3] and *evaluation* [5].

[4] Available at `http://www2.disco.unimib.it/fiorino/beta.tgz`

Family	Status	β	PLTL
lift	sat.	76	42
anzu-amba	sat.	18	38
acacia-demo-v3	sat.	12	12
anzu-genbuf	sat.	28	26
rozier counters	sat.	35	65

Family	Status	β	PLTL
lift	unsat.	0	8
schuppan-O1	unsat.	27	10
schuppan-O2	unsat.	7	10
schuppan-phltl	unsat.	5	3

Fig. 3. Comparison between β and PLTL

References

1. A. Bolotov, O. Grigoriev, and V. Shangin. Automated natural deduction for propositional linear-time temporal logic. In *TIME (2007)*, pages 47–58. IEEE Computer Society, 2007.
2. K. Brünnler and M. Lange. Cut-free sequent systems for temporal logic. *Journal of Logic and Algebraic Programming*, 76(2):216–225, 2008.
3. M. Ferrari, C. Fiorentini, and G. Fiorino. fCube: An efficient prover for intuitionistic propositional logic. In C. G. Fermüller et al., editor, *LPAR-17*, volume 6397 of *LNCS*, pages 294–301. Springer, 2010.
4. M. Ferrari, C. Fiorentini, and G. Fiorino. Simplification rules for intuitionistic propositional tableaux. *ACM Transactions on Computational Logic (TOCL)*, 13(2):14:1–14:23, 2012.
5. M. Ferrari, C. Fiorentini, and G. Fiorino. An evaluation-driven decision procedure for G3i. *ACM Transactions on Computational Logic (TOCL)*, 6(1):8:1–8:37, 2015.
6. G. Fiorino. Tableau calculus based on a multiple premise rule. *Information Sciences*, 180(19):371–399, 2010.
7. G. Fiorino. Refutation in Dummett logic using a sign to express the truth at the next possible world. In T. Walsh, editor, *IJCAI 2011*, pages 869–874. IJCAI/AAAI, 2011.
8. M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic (TOCL)*, 2(1):12–56, 2001.
9. J. Gaintzarain, M. Hermo, P. Lucio, and M. Navarro. Systematic semantic tableaux for PLTL. *Electronic Notes in Theoretical Computer Science*, 206:59–73, 2008.
10. J. Gaintzarain, M. Hermo, P. Lucio, M. Navarro, and F. Orejas. Dual systems of tableaux and sequents for PLTL. *Journal of Logic and Algebraic Programming*, 78(8):701–722, 2009.
11. F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In Harrie de Swart, editor, *TABLEAUX'98*, volume 1397 of *LNCS*, pages 217–232. Springer-Verlag, 1998.
12. B. Paech. Gentzen-systems for propositional temporal logics. In E. Börger et al., editor, *CSL'88*, volume 385 of *LNCS*, pages 240–253. Springer, 1988.
13. R. Pliuskevicius. Investigation of finitary calculus for a discrete linear time logic by means of infinitary calculus. In J. Barzdins et al., editor, *Baltic Computer Science*, volume 502 of *LNCS*, pages 504–528. Springer, 1991.
14. S. Schwendimann. A new one-pass tableau calculus for PLTL. In H. C. M. de Swart, editor, *TABLEAUX'98*, volume 1397 of *LNCS*, pages 277–291. Springer, 1998.
15. M. Suda and C. Weidenbach. Labelled superposition for PLTL. In N. Bjørner et al., editor, *LPAR-18*, volume 7180 of *LNCS*, pages 391–405. Springer, 2012.

Ontoceramic: an OWL ontology for ceramics classification

Domenico Cantone[1], Marianna Nicolosi-Asmundo[1], Daniele Francesco Santamaria[1], and Francesca Trapani[2]

[1] Department of Mathematics and Computer Science, University of Catania
email: {cantone,nicolosi}@dmi.unict.it, daniele.f.santamaria@gmail.com
[2] Department of Humanistic Sciences, University of Catania
email: patercolo@alice.it

Abstract. In this note, we describe Ontoceramic, an OWL 2 ontology for cataloguing and classifying ancient ceramics. Ontoceramic has been defined through a synergic effort of computer scientists and archaeologists, by taking into account the most important papers in the field. It has been designed with the purpose of efficiently addressing significant problems concerning knowledge management about ceramics such as, for instance, classification by shape and type, and analysis of findings by their components. Ontoceramic implements CIDOC CRM, the standard ontology for describing concepts and relationships used in cultural heritage documentation, and LinkedGeoData for describing locations.

1 Introduction

In the last decades, the task of efficiently organize the classification process of archaeological findings – with particular focus on ceramics – has become more and more relevant for scholars and researchers in the field [10]. Currently, in fact, archaeological findings cataloguing and classification are often performed either by traditional methods, like hard-copy archives, or by standard digital techniques, like relational databases. However, such methods have severe drawbacks. For instance, they are often based on tools mainly developed and maintained locally; hence, they usually store partial data, rarely shared with the whole scientific community. This results in an incoherent use of information. Also, they do not support flexible data-management and information retrieval algorithms due to the lack of advanced reasoning means such as logical inferencing, consistency and soundness check.

Semantic web is a vision of the World Wide Web in which information carries an explicit meaning, so it can be automatically processed and integrated by machines, and data can be accessed and modified at a global level, resulting in increased coherence and dissemination of knowledge. Moreover, by means of automated reasoning procedures, it is possible to extract implicit information present in data, thus permitting to gain a deeper knowledge of the domain. Informally, in the context of computer science, an *ontology* defines a set of representational primitives (classes and attributes) with which to model a domain of knowledge or discourse

[11]. In the last years, the potential of ontologies has been recognized by archaeologists [1, 9]. Some projects have been undertaken concerning either single typologies of archaeological findings or several different materials related to each other [4, 7].
In this contribution we describe Ontoceramic [12, 3], an *OWL 2* (Ontology Web Language 2) defined by a synergic effort between computer scientists and archaeologists as a first step to overcome the problem of efficiently mechanize the task of correctly cataloguing ceramics and to make such knowledge easily retrievable by scholars and researchers in the field.
This initial definition of the ontology takes into account the most important papers in the field [4] and strongly relies on ICCD (Istituto Centrale per il Catalogo e la Documentazione) data sheets. The latter choice is motivated by the need of easily importing data from relational databases currently used in archaeological institutions (i.e., universities, museums, superintendences of cultural heritage). Moreover, ontoceramic implements the ontology CIDOC CRM [9] that provides a formal structure and definitions for describing concepts and relationships used in cultural heritage documentation, and furnishes the "semantic glue" needed to integrate different sources of information, such as the ones published by museums, libraries, and archives. Ontoceramic also implements LinkedGeoData [13], a large ontology for spatial knowledge base. It consists of more than 90 classes, 33 object properties, and 20 data properties. It includes a number of SWRL (Semantic Web Rule Language) rules[1] allowing several reasoning tasks on the knowledge domain in a short time.
The expressive power of the language underlying Ontoceramic has been studied in [12, 2]. In particular, in [12], we defined an *OWL 2* profile constructed from a decidable fragment of set theory and proved that the computational complexity of the consistency problem for Ontoceramic knowledge bases is NP-complete.
Ontoceramic has been developed using the *Protégé* editor[2] and classified by the Hermit,[3] Pellet,[4] and FaCT++[5] reasoners.

2 Ontoceramic

Ontoceramic aims at covering different aspects of the ceramic classification and cataloguing problem.
To begin with, Ontoceramic helps in identifying unambiguously the location of findings. One can have many locations to consider for a specific finding such as province, region, state, and so on. Locations are introduced by means of a taxonomy of OWL classes. Association between locations is performed by means of a taxonomy of object-properties. The

[1] http://www.w3.org/Submission/SWRL/
[2] http://protege.stanford.edu
[3] http://hermit-reasoner.com
[4] https://www.w3.org/2001/sw/wiki/Pellet
[5] http://owl.cs.manchester.ac.uk/tools/fact/

path allowed is shown in Fig. 2, where double-hoop entities are optional. Classes and object-properties involved are shown in Fig. 1. Reasoning tasks involving places are strengthened using SWRL rules of the type of the ones showed in Fig. 3. The entity "Localisation" is equivalent to "LinkedGeoData:Place"; equivalences between subclasses of "Localisation" and of "Place" are not reported here for space reasons. The object-property "hasLocalisation" is a subproperty of the CIDOC property "P54_has_current_permanent_location".

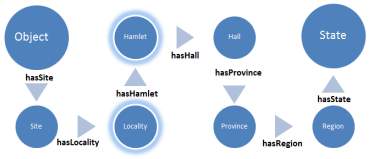


Fig. 2. Allowed path for locations.

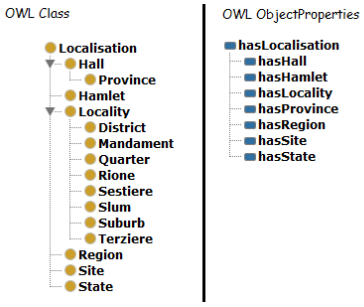


Fig. 1. Classes and properties for locations.

OntoCeramic Rules
Locality(y), hasLocalisation(?x,?y) → hasLocality(?x,?y)
Region(y), hasLocalisation(?x,?y) → hasRegion(?x,?y)
State(y), hasLocalisation(?x,?y) → hasState(?x,?y)
Hamlet(y), hasLocalisation(?x,?y) → hasHamlet(?x,?y)
Hall(y), hasLocalisation(?x,?y) → hasHall(?x,?y)
Site(y), hasLocalisation(?x,?y) → hasSite(?x,?y)
Province(y), hasLocalisation(?x,?y) → hasProvince(?x,?y)

Fig. 3. SWRL rules for reasoning with locations.

Ontoceramic is also able to handle components belonging to a specific object. For example, a cup can be found broken in three disjoint parts that may require distinct descriptions. Usually, in hard-copy versions of ceramic catalogues, components of a object are included in a descriptive field of the archive. Thus, information about each component can be extracted manually by the users or by means of search keys. In Ontoceramic, instead, objects as well as their fragments are considered as entities. Such classes are subclasses of the CIDOC "E22_Man-Made_Object". Each fragment is associated to the object it belongs to by the object-property "IsFragmentOf". Analogously, each object is associated to its components by the object-property "hasFragment" (notice that "hasFragment" and "IsFragmentOf" are inverses of each other) and, in particular, also by its sub-properties. Subproperties of "hasFragment" are intended to relate an object with one of its fragments, taking care of its correct functionality in the object. Thanks to this construction, one

can precisely describe every part of an object. “isFragmentOf” is a sub-property of the CIDOC “P46_is_composed_of”. Types of fragments that can be implemented and also extended are described in Fig. 4. Object-properties which associate objects to their fragments are shown in the taxonomy illustrated in Fig. 5.

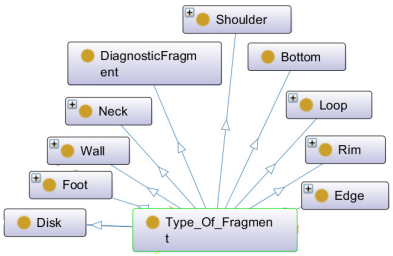


Fig. 4. Classes of fragments.

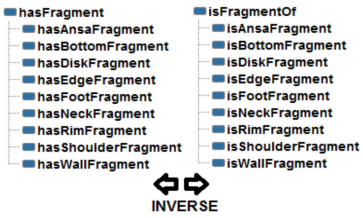


Fig. 5. Properties for fragments.

The class of an object is structured in a taxonomy as shown in Fig. 6. To assign a class to an object, the object-property “hasClass” is provided, having “Object” as domain and “Class” as range. The type of an object is represented in the “Object” taxonomy. “Class” is a subclass of CIDOC “E25_Man-Made_Feature”.

Sample and sector of a finding are represented by the classes “Sample” and “Sector”, respectively. Instances of these classes are associated to an object by means of the “hasSample” and “hasSector” object-properties.

Ontoceramic can also specify the color of a finding using the Munsell Color System by means of the data-property “hasColor”. This property is endowed with three sub-data-properties, namely “hasChroma”, “hasHue”, and “hasValue”, for Munsell chroma, hue, and value, respectively. In addition, one can provide the discovery date and a general additional description of the object under consideration using, respectively, the “hasSiteDate” and “hasGeneralDescription” data-properties.

For objects and fragments one can specify their measurements. For example, thickness of an object can be represented by the generic data-property “hasThickness” and by its specific subproperties. For instance, the data-property “hasWallThickness” is used when we indicate the thickness of the object wall, “hasBottomThickness”, instead, is used when we indicate the thickness of the foot, and so on. The hierarchy of these properties is shown in Fig. 7.

It is possible to specify whether a fragment can be physically associated with another fragment to compose a unique object. In this case the “isFittedWith” object-property is applied. One can indicate the number of the box and the number of the sheet of the hard-copy archive of an object description using a “nonNegativeInteger” value in “hasBox” and “hasSheet” data-properties, respectively.

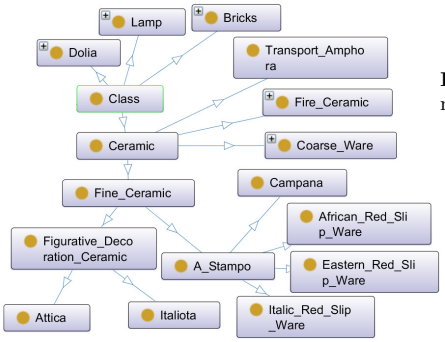


Fig. 6. Taxonomy of Classes.

Fig. 7. Data-Properties for measurement.

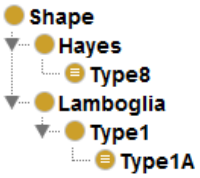


Fig. 8. Taxonomy for Shapes.

To solve a problem concerning the management of the shape of a finding, the "Shape" taxonomy is provided, as shown in Fig. 8. The shape of an object is represented by the "Shape" class. Instances of this class and of its subclasses are associated to an object by means of the "hasShape" object-property. Every instance of the class "Shape" can be uniquely identified by the properties "hasFirstShapeDescriptor", "hasSecondShapeDescriptor", "hasThirdShapeDescriptor", which are subproperties of the "hasShapeDescriptor" data-property. For each data-property, a string value can be specified: these values will be the keys for the "Shape" instances. These properties can be used as additional human-readable descriptors. At present, Ontoceramic supports only few types of "shape" and of "shape type", but one can add an arbitrary number of these classes and possibly assert equivalences among them. Currently, there is no world-wide agreement on the use of a specific nomenclature to indicate the shape and the type of an object. The "Shape" taxonomy is an attempt to face this problem providing a class for each type of shape and several classes for each specific shape; where required, an equivalence relation can be established among the shape classes or their sub-classes, to identify shapes which are identical with respect to the classification system but which have been called with different names. For example, in Fig. 8 "Lamboglia 1A" and "Hayes 8" are represented as equivalent. The class "Shape" is a subclass of CIDOC "E25_Man-Made_Feature".

3 Conclusions and future work

In this preliminary work we have presented an ontology which takes advantage of semantic web technologies in order to accomplish several tasks related to ancient ceramics cataloguing and classification such as reasoning on location, shape, and type of findings. We are currently populating Ontoceramic with datasets of ceramics coming from excavations located in eastern Sicily. We also plan to include support for stratigraphic excavations, bibliographic references management including authors and revisors, and identification of the production factory.

References

1. Angelis S., Benardou A., Chatzidiakou N., Constantopoulos P., Dallas C., Hughes L. M., Papachristopoulos L., Papaki E., Pertsas V., Documenting and reasoning about research on ancient Corinthia using the NeDiMAH Methods Ontology (NeMO), *Computer Applications and Quantitative Methods in Archaeology (CAA)*, 2015.
2. Cantone D., Longo C., Nicolosi-Asmundo M., Santamaria D. F., Web ontology representation and reasoning via fragments of set theory. To appear in *Proc. of the 9th International Conference on Web Reasoning and Rule Systems*, 2015.
3. Cantone D., Nicolosi-Asmundo M., Santamaria D. F., Trapani F., An ontology for ceramics cataloguing, *Computer Applications and Quantitative Methods in Archaeology (CAA)*, 2015.
4. Corti L., *I beni culturali e la loro catalogazione*. Bruno Mondadori, Milano 2003.
5. DBpedia, http://dbpedia.org.
6. M. Doerr. *The CIDOC CRM - An ontological approach to semantic interoperability of metadata*. AI Magazine 24(3): 75-92 (2003).
7. Felicetti A., Scarselli T., Mancinelli M. L., Niccolucci F., Mapping ICCD Archaeological Data to CIDOC CRM: the RA Schema, *Practical Experiences with CIDOC CRM and its Extensions (CRMEX)*, Malta, 26 September 2013.
8. Gašević D., Djurić, D., Devedžić V., *Model Driven Engineering and Ontology Development (2nd ed.)*. Springer. pp. 194. ISBN 978-3-642-00282-3, 2009.
9. Letricot R., Szabados A.V., L'ontologie CIDOC CRM appliquée aux objets du patrimoine antique. In *Archeologia e Calcolatori*, supplemento 5, 2014, pp. 257-272.
10. Moscati P., *Archeologia e società dell'informazione*, in http://www.treccani.it/enciclopedia/archeologia-e-societa-dell-informazione_(XXI_Secolo).
11. Encyclopedia of Database Systems, Ling Liu and M. Tamer zsu (Eds.), Springer-Verlag, 2009.
12. Santamaria D. F., *A Set-Theoretical Representation for OWL 2 Profiles*, LAP Lambert Academic Publishing, ISBN 978-3-659-68797-6, 2015.
13. Stadler C., Lehmann J., Höffner K., Auer S. (2012), LinkedGeoData: A Core for a Web of Spatial Open Data, *Semantic Web Journal* 3 (4), 333-354.

Abductive Logic Programming for Datalog$^{\pm}$ ontologies

Marco Gavanelli[1], Evelina Lamma[1], Fabrizio Riguzzi[2], Elena Bellodi[1], Riccardo Zese[1], and Giuseppe Cota[1]

[1] Dipartimento di Ingegneria – University of Ferrara
[2] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy [name.surname]@unife.it

Abstract. Ontologies are a fundamental component of the Semantic Web since they provide a formal and machine manipulable model of a domain. Description Logics (DLs) are often the languages of choice for modeling ontologies. Great effort has been spent in identifying decidable or even tractable fragments of DLs. Conversely, for knowledge representation and reasoning, integration with rules and rule-based reasoning is crucial in the so-called Semantic Web stack vision. Datalog$^{\pm}$ is an extension of Datalog which can be used for representing lightweight ontologies, and is able to express the DL-Lite family of ontology languages, with tractable query answering under certain language restrictions. In this work, we show that Abductive Logic Programming (ALP) is also a suitable framework for representing Datalog$^{\pm}$ ontologies, supporting query answering through an abductive proof procedure, and smoothly achieving the integration of ontologies and rule-based reasoning. In particular, we consider an Abductive Logic Programming framework named $\mathcal{S}$CIFF and derived from the IFF abductive framework, able to deal with existentially (and universally) quantified variables in rule heads, and Constraint Logic Programming constraints. Forward and backward reasoning is naturally supported in the ALP framework. The $\mathcal{S}$CIFF language smoothly supports the integration of rules, expressed in a Logic Programming language, with Datalog$^{\pm}$ ontologies, mapped into $\mathcal{S}$CIFF (forward) integrity constraints. The main advantage is that this integration is achieved within a single language, grounded on abduction in computational logic.

1 Introduction

The main idea of the Semantic Web is making information available in a form that is understandable and automatically manageable by machines [21]. Ontologies are engineering artefacts consisting of a vocabulary describing some domain, and an explicit specification of the intended meaning of the vocabulary (i.e., how concepts should be classified), possibly together with constraints capturing additional knowledge about the domain. Ontologies therefore provide a formal and machine manipulable model of a domain, and this justifies their use in the Semantic Web.

In order to realize this vision, the W3C has supported the development of a family of knowledge representation formalisms of increasing complexity for defining ontologies, called Web Ontology Language (OWL). Ontologies are a fundamental component of the Semantic Web, and Description Logics (DLs) are often the languages of choice for modeling them.

Several DL reasoners, such us Pellet [32], RacerPro [19] and HermiT [31], are used to extract implicit information from the modeled ontologies, and most of them implement the tableau algorithm in a procedural language. Nonetheless, some tableau expansion rules are non-deterministic, thus requiring to implement a search strategy in an or-branching search space. A different approach is to provide a Prolog-based implementation for the tableau expansion rules [34].

Extensive work focused on developing tractable DLs, identifying the *DL-Lite* family [14], for which answering conjunctive queries is in AC_0 in data complexity.

In a related research direction, [11] proposed Datalog$^{\pm}$, an extension of Datalog with existential rules for defining ontologies. Datalog$^{\pm}$ can be used for representing lightweight ontologies, and encompasses the DL-Lite family [10]. By suitably restricting the language syntax and adopting appropriate syntactic conditions, also Datalog$^{\pm}$ achieves tractability [9].

In this work we consider the Datalog$^{\pm}$ language and show how ontologies expressed in this language can be also modeled in an Abductive Logic Programming (ALP) framework, where query answering is supported by the underlying ALP proof procedure. ALP has been proved a powerful tool for knowledge representation and reasoning [24], taking advantage from ALP operational support as (static or dynamic) verification tool. ALP languages are usually equipped with a declarative (model-theoretic) semantics, and an operational semantics given in terms of a proof-procedure. Several abductive proof procedures have been defined (both backward, forward, and a mix of the two such), with many different applications (diagnosis, monitoring, verification, etc.). Among them, the IFF abductive proof-procedure [17] was proposed to deal with forward rules, and with non-ground abducibles. This proof procedure has been later extended [4], and the resulting proof procedure, named $\mathcal{S}$CIFF, can deal with both existentially and universally quantified variables in rule heads, and Constraint Logic Programming (CLP) constraints [23]. The resulting system was used for modeling and implementing several knowledge representation frameworks, such as deontic logic [6], normative systems, interaction protocols for multi-agent systems [7], Web services choreographies [2], etc.

Here we concentrate on Datalog$^{\pm}$ ontologies, and show how an ALP language enriched with quantified variables (existential to our purposes) can be a useful knowledge representation and reasoning framework for them. We do not focus here on complexity results of the overall system, which is, however, not tractable.

Forward and backward reasoning is naturally supported by the ALP proof procedure, and the considered $\mathcal{S}$CIFF language smoothly supports the integration of rules, expressed in a Logic Programming language, with ontologies expressed in Datalog$^{\pm}$. In fact, $\mathcal{S}$CIFF allows us to map Datalog$^{\pm}$ ontologies into the forward integrity constraints on which it is based.

In the following, Section 2 briefly introduces Datalog$^{\pm}$. Section 3 introduces Abductive Logic Programming and the $\mathcal{S}$CIFF language, with a mention to its abductive proof procedure. Section 4 shows how the considered Datalog$^{\pm}$ language can be mapped into $\mathcal{S}$CIFF, and the kind of queries that the abductive proof procedure can handle. Section 5 illustrates related work. Section 6 concludes the paper, and outlines future work.

2 Datalog$^{\pm}$

Datalog$^{\pm}$ extends Datalog by allowing existential quantifiers, the equality predicate and the truth constant *false* in rule heads. Datalog$^{\pm}$ can be used for representing lightweight ontologies and is able to express the DL-Lite family of ontology languages [10]. By suitably restricting the language syntax, Datalog$^{\pm}$ achieves tractability [9].

In order to describe Datalog$^{\pm}$, let us assume (i) an infinite set of data constants Δ, (ii) an infinite set of labeled nulls Δ_N (used as "fresh" Skolem terms), and (iii) an infinite set of variables Δ_V. Different constants represent different values (unique name assumption), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ vectors of variables $X_1, \ldots, X_k$ with $k \geq 0$. A relational schema $\mathcal{R}$ is a finite set of relation names (or predicates). A term t is a constant, null or variable. An atomic formula (or atom) has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate, and $t_1, \ldots, t_n$ are terms. A database D for $\mathcal{R}$ is a possibly infinite set of atoms with predicates from $\mathcal{R}$ and arguments from $\Delta \cup \Delta_N$. A conjunctive query (CQ) over $\mathcal{R}$ has the form $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms having as arguments variables $\mathbf{X}$ and $\mathbf{Y}$ and constants (but no nulls). A Boolean CQ (BCQ) over $\mathcal{R}$ is a CQ having head predicate q of arity 0 (i.e., no variables in $\mathbf{X}$).

We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via homomorphisms, which are mappings $\mu : \Delta \cup \Delta_N \cup \Delta_V \rightarrow \Delta \cup \Delta_N \cup \Delta_V$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to term vectors, atoms, sets of atoms, and conjunctions of atoms. The set of all answers to a CQ $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu : \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The answer to a BCQ $q = \exists \mathbf{Y} \Phi(\mathbf{Y})$ over a database D, denoted $q(D)$, is Yes, denoted $D \models q$, iff there exists a homomorphism $\mu : \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{Y})) \subseteq D$, i.e., if $q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency* (or TGD) F is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of F, respectively. Such F is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, there exists an extension h' of h such that $h'(\Psi(\mathbf{X}, \mathbf{Z})) \subseteq D$. We usually omit the universal quantifiers in

TGDs. A TGD is *guarded* iff it contains an atom in its body that involves all variables appearing in the body.

Query answering under TGDs is defined as follows. For a set of TGDs T on $\mathcal{R}$, and a database D for $\mathcal{R}$, the set of models of D given T, denoted $mods(D,T)$, is the set of all (possibly infinite) databases B such that $D \subseteq B$ and every $F \in T$ is satisfied in B. The set of answers to a CQ q on D given T, denoted $ans(q,D,T)$, is the set of all tuples $\mathbf{t}$ such that $\mathbf{t} \in q(B)$ for all $B \in mods(D,T)$. The answer to a BCQ q over D given T is Yes, denoted $D \cup T \models q$, iff $B \models q$ for all $B \in mods(D,T)$.

A Datalog$^{\pm}$ theory may contain also *negative constraints* (or NC), which are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded). The universal quantifiers are usually left implicit.

Equality-generating dependencies (or EGDs) are the third component of a Datalog$^{\pm}$ theory. An EGD F is a first-order formula of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of F, is a conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. We call $X_i = X_j$ the *head* of F. Such F is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. We usually omit the universal quantifiers in EGDs.

The *chase* is a bottom-up procedure for deriving atoms entailed by a database and a Datalog$^{\pm}$ theory. The chase works on a database through the so-called TGD and EGD chase rules.

The TGD chase rule is defined as follows. Given a relational database D for a schema $\mathcal{R}$, and a TGD F on $\mathcal{R}$ of the form $\forall \mathbf{X} \forall \mathbf{Y} \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z} \Psi(\mathbf{X},\mathbf{Z})$, F is *applicable to* D if there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X},\mathbf{Y})$ to atoms of D. Let F be applicable and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N, z_j \notin D$, and z_j lexicographically follows all other labeled nulls already introduced. The result of the application of the TGD chase rule for F is the addition to D of all the atomic formulas in $h_1(\Psi(\mathbf{X},\mathbf{Z}))$ that are not already in D.

The EGD chase rule is defined as follows. An EGD F on $\mathcal{R}$ of the form $\Phi(\mathbf{X}) \rightarrow X_i = X_j$ is *applicable to* a database D for $\mathcal{R}$ iff there exists a homomorphism $h : \Phi(\mathbf{X}) \rightarrow D$ such that $h(X_i)$ and $h(X_j)$ are different and not both constants. If $h(X_i)$ and $h(X_j)$ are different constants in Δ, then there is a *hard violation* of F. Otherwise, the result of the application of F to D is the database $h(D)$ obtained from D by replacing every occurrence of $h(X_i)$ with $h(X_j)$ if $h(X_i)$ precedes $h(X_j)$ in the lexicographic order, and every occurrence of $h(X_j)$ with $h(X_i)$ if $h(X_j)$ precedes $h(X_i)$ in the lexicographic order.

The chase algorithm consists of an exhaustive application of the TGD and EGD chase rules that may lead to an infinite result. The chase rules are applied iteratively: in each iteration (1) a single TGD is applied once and then (2) the EGDs are applied until a fix point is reached. EGDs are assumed to be separable [12]. Intuitively, separability holds whenever: (i) if there is a hard violation of an

EGD in the chase, then there is also one on the database w.r.t. the set of EGDs alone (i.e., without considering the TGDs); and (ii) if there is no hard violation, then the answers to a BCQ w.r.t. the entire set of dependencies equals those w.r.t. the TGDs alone (i.e., without the EGDs).

The two problems of CQ and BCQ evaluation under TGDs and EGDs are LOGSPACE-equivalent [11]. Moreover, query answering under TGDs is equivalent to query answering under TGDs with only single atoms in their heads [9]. Henceforth, we focus only on the BCQ evaluation problem and we assume that every TGD has a single atom in its head. A BCQ q on a database D, a set T_T of TGDs and a set T_E of EGDs can be answered by performing the chase and checking whether the query is entailed by the extended database that is obtained. In this case we write $D \cup T_T \cup T_E \models q$.

Example 1. Let us consider the following ontology for a real estate information extraction system, a slight modification of the one presented in Gottlob et al. [18]:

$F_1 = ann(X, label), ann(X, price), visible(X) \rightarrow priceElem(X)$

If X is annotated as a label, as a price and is visible, then it is a price element.

$F_2 = ann(X, label), ann(X, priceRange), visible(X) \rightarrow priceElem(X)$

If X is annotated as a label, as a price range, and is visible, then it is a price element.

$F_3 = priceElem(E), group(E, X) \rightarrow forSale(X)$

If E is a price element and is grouped with X, then X is for sale.

$F_4 = forSale(X) \rightarrow \exists P\ price(X, P)$

If X is for sale, then there exists a price for X.

$F_5 = hasCode(X, C), codeLoc(C, L) \rightarrow loc(X, L)$

If X has postal code C, and C's location is L, then X's location is L.

$F_6 = hasCode(X, C) \rightarrow \exists L\ codeLoc(C, L), loc(X, L)$

If X has postal code C, then there exists L such that C has location L and so does X.

$F_7 = loc(X, L1), loc(X, L2) \rightarrow L1 = L2$

If X has the locations $L1$ and $L2$, then $L1$ and $L2$ are the same.

$F_8 = loc(X, L) \rightarrow advertised(X)$

If X has a location L then X is advertised.

Suppose we are given the database

$$codeLoc(ox1, central), codeLoc(ox1, south), codeLoc(ox2, summertown)$$
$$hasCode(prop1, ox2), ann(e1, price), ann(e1, label), visible(e1),$$
$$group(e1, prop1)$$

The atomic BCQs $priceElem(e1)$, $forSale(prop1)$ and $advertised(prop1)$ evaluate to true, while the CQ $loc(prop1, L)$ has answers $q(L) = \{summertown\}$. In fact, even if $loc(prop1, z_1)$ with $z_1 \in \Delta_N$ is entailed by formula F_5, formula F_7 imposes that $summertown = z_1$. If F_7 were absent then $q(L) = \{summertown, z_1\}$.

Answering BCQs q over databases and ontologies containing NCs can be performed by first checking whether the BCQ $\Phi(\mathbf{X})$ evaluates to false for each NC of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$. If one of these checks fails, then the answer to the original BCQ q is positive, otherwise the negative constraints can be simply ignored when answering the original BCQ q.

A guarded Datalog$^{\pm}$ ontology is a quadruple (D, T_T, T_C, T_E) consisting of a database D, a finite set of guarded TGDs T_T, a finite set of negative constraints T_C and a finite set of EGDs T_E that are separable from T_T. The data complexity (i.e., the complexity where both the query and the theory are fixed) of evaluating BCQs relative to a guarded Datalog$^{\pm}$ theory is polynomial [9].

In the case in which the EGDs are key dependencies and the TGDs are inclusion dependencies, Calì et al. [13] proposed a backward chaining algorithm for answering BCQ. A *key dependency* κ is a set of EGDs of the form

$$\{r(\mathbf{X}, Y_1, \ldots, Y_m), r(\mathbf{X}, Y_1', ..., Y_m') \rightarrow Y_i = Y_i'\}_{1 \leq i \leq m}$$

A TGD of the form $r_1(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z} r_2(\mathbf{X}, \mathbf{Z})$, where r_1 and r_2 are predicate names and no variable appears more than once in the body nor in the head, is called an *inclusion dependency*. The key dependencies must not interact with the inclusion dependencies, similarly to the semantic separability condition mentioned above for TGDs and EGDs. In this case once it is known that no hard violation occurs, queries can be answered by considering the inclusion dependencies only, ignoring the key dependencies. A necessary and sufficient syntactic condition for non interaction is based on the construction of CD-graphs [13].

3 ALP and its proof procedure

Abductive Logic Programming (ALP, for short) is a family of programming languages that integrate abductive reasoning into logic programming. An ALP program is a logic program, consisting of a set of clauses, that can contain in the body some distinguished predicates, belonging to a set $\mathcal{A}$ and called *abducibles*. The aim is finding a set of abducibles **EXP**, built from symbols in $\mathcal{A}$ that, together with the knowledge base, is an explanation for a given known effect (also called *goal* $\mathcal{G}$):

$$KB \cup \mathbf{EXP} \models \mathcal{G}. \tag{1}$$

Also, **EXP** should satisfy a set of logic formulae, called *Integrity Constraints* IC:

$$KB \cup \mathbf{EXP} \models IC. \tag{2}$$

E.g., a knowledge base might contain a set of rules stating that a person is a nature lover

$$natureLover(X) \leftarrow \mathbf{hasAnimal}(X, Y), \mathbf{pet}(Y).$$
$$natureLover(X) \leftarrow \mathbf{biologist}(X).$$

From this knowledge base one can infer, e.g., that each person who owns a pet is a nature lover. However, in some cases we might have the information that

kevin is a nature lover, and wish to infer more information about him. In such a case we might label predicates **hasAnimal**, **pet** and **biologist** as abducible (in the following, abducible predicates are written in **bold**) and apply an abductive proof procedure to the knowledge base. Two explanations are possible: either there exists an animal that is owned by *kevin* and that is a pet:

$$(\exists Y) \qquad \mathbf{hasAnimal}(kevin, Y), \mathbf{pet}(Y)$$

or *kevin* is a biologist:

$$\mathbf{biologist}(kevin)$$

We see that the computed answer includes abduced atoms, which can contain variables.

Integrity constraints can help reducing the number of computed explanations, ruling out those that are not possible. For example, the following integrity constraint states that to become a biologist one needs to be at least 25 years old:

$$\mathbf{biologist}(X), \mathbf{age}(X, A) \rightarrow A \geq 25$$

We might know that *kevin* is a child, and have a definition of the predicate *child*:

$$child(X) \leftarrow \mathbf{age}(X, A), A < 10.$$

In this example we see the usefulness of constraints as in Constraint Logic Programming [23]: the symbols $<, \geq, ...$ are handled as constraints, i.e., they are not predicates defined in a knowledge base, but they associate a numeric domain to the involved variables and restrict it according to constraint propagation. Now, the goal $natureLover(kevin), child(kevin)$ returns only one possible explanation:

$$(\exists Y)(\exists A) \qquad \mathbf{hasAnimal}(kevin, Y), \mathbf{pet}(Y), \mathbf{age}(kevin, A) \qquad A < 10$$

since the option that *kevin* is a biologist is ruled out. Note that we do not need to know the exact age of *kevin* to rule out the biologist hypothesis.

$\mathcal{S}$CIFF [4] is a language in the ALP class, originally designed to model and verify interactions in open societies of agents [7], and it is an extension of the IFF proof-procedure [17]. As in the IFF language, it considers forward integrity constraints of the form

$$body \rightarrow head$$

where the *body* is a conjunction of literals and the head is a disjunction of conjunctions of literals. While in the IFF the literals can be built only on defined or abducible predicates, in $\mathcal{S}$CIFF they can also be CLP constraints, occurring events (only in the body), or positive and negative expectations.

Definition 1. *A* $\mathcal{S}$CIFF Program *is a pair* $\langle KB, \mathcal{IC} \rangle$ *where KB is a set of clauses and* $\mathcal{IC}$ *is a set of forward rules called* Integrity Constraints *(ICs, for short in the following).*

$\mathcal{S}$CIFF considers a (possibly dynamically growing) set of facts (named event set) **HAP**, that contains ground atoms $\mathbf{H}(Event[,Time])$. This set can grow dynamically, during the computation, thus implementing a dynamic acquisition of events. Some distinguished abducibles are called *expectations*. A *positive expectation*, written $\mathbf{E}(Event[,Time])$ means that a corresponding event $\mathbf{H}(Event[,Time])$ is expected to happen, while $\mathbf{EN}(Event[,Time])$ is a *negative expectation*, and requires events $\mathbf{H}(Event[,Time])$ not to happen. To simplify the notation, we will omit the $Time$ argument from events and expectations when not needed, as it is for our purposes.

While events are ground atoms, expectations can contain variables. In positive expectations all variables are existentially quantified (expressing the idea that a single event is enough to support them), while negative expectations are universally quantified, so that any event matching with a negative expectation leads to inconsistency with the current hypothesis. CLP [23] constraints can be imposed on variables. The computed answer includes in general three elements: a substitution for the variables in the goal (as usual in Prolog), the constraint store (as in CLP), and the set **EXP** of abduced literals.

The declarative semantics of $\mathcal{S}$CIFF includes the classic conditions of abductive logic programming

$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{G}$$
$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{IC}$$

plus specific conditions to support the confirmation of expectations.

Positive expectations are confirmed if

$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathbf{E}(X) \rightarrow \mathbf{H}(X),$$

while negative expectations are confirmed (or better they are not violated) if

$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathbf{EN}(X) \wedge \mathbf{H}(X) \rightarrow \mathit{false}.$$

The declarative semantics of $\mathcal{S}$CIFF also requires that the same event cannot be expected both to happen and not to happen

$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathbf{E}(X) \wedge \mathbf{EN}(X) \rightarrow \mathit{false} \tag{3}$$

The $\mathcal{S}$CIFF proof-procedure is a rewriting system that defines a proof tree, whose nodes represent states of the computation. A set of transitions rewrite a node into one or more children nodes. $\mathcal{S}$CIFF inherits the transitions of the IFF proof-procedure [17], and extends it in various directions. We recall the basics of $\mathcal{S}$CIFF; a complete description is in [4], with proofs of soundness, completeness, and termination. An efficient implementation of $\mathcal{S}$CIFF is described in [5].

Each node of the proof is a tuple $T \equiv \langle R, CS, PSIC, \mathbf{EXP}\rangle$, where R is the resolvent, CS is the CLP constraint store, PSIC is a set of implications (called *Partially Solved Integrity Constraints*) derived from propagation of integrity constraints, and **EXP** is the current set of abduced literals. The main transitions, inherited from the IFF are:

Unfolding replaces a (non abducible) atom with its definitions;
Propagation if an abduced atom $\mathbf{a}(X)$ occurs in the condition of an IC (e.g., $\mathbf{a}(Y) \rightarrow p$), the atom is removed from the condition (generating $X = Y \rightarrow p$);
Case Analysis given an implication containing an equality in the condition (e.g., $X = Y \rightarrow p$), generates two children in logical or (in the example, either $X = Y$ and p, or $X \neq Y$);
Equality rewriting rewrites equalities as in the Clark's equality theory;
Logical simplifications other simplifications like (true $\rightarrow A$) $\Leftrightarrow A$, etc.

$\mathcal{S}$CIFF includes also the transitions of CLP [23] for constraint solving. Finally, in this paper we consider the *generative version* of $\mathcal{S}$CIFF, previously called also g-$\mathcal{S}$CIFF [3], in which also the **H** events in the set **HAP** are considered as abducibles, and can be assumed like the other abducible predicates, beside being provided as input in the event set **HAP**.

4 Mapping Datalog$^{\pm}$ into ALP programs

In this section, we show that a Datalog$^{\pm}$ program can be represented as a set of $\mathcal{S}$CIFF integrity constraints and an event set. $\mathcal{S}$CIFF abductive declarative semantics provides the model-theoretic counterpart to Datalog$^{\pm}$ semantics. Operationally, query answering is achieved bottom-up via the *chase* in Datalog$^{\pm}$, while in the ALP framework it is supported by the $\mathcal{S}$CIFF proof procedure. $\mathcal{S}$CIFF is able to integrate a knowledge base KB, expressed in terms of Logic Programming clauses, possibly with abducibles in their body, and to deal with integrity constraints.

To our purposes, we consider only $\mathcal{S}$CIFF programs with an empty KB, ICs with only conjunctions of positive expectations and CLP constraints (or *false*) in their heads. We show that this subset of the language suffices to represent Datalog$^{\pm}$ ontologies.

We map the finite set of relation names of a Datalog$^{\pm}$ relational schema $\mathcal{R}$ into the set of predicates of the corresponding $\mathcal{S}$CIFF program.

Definition 2. *The τ mapping is recursively defined as follows, where A is an atom,* **M** *can be either* **H** *or* **E**, *and F_1, F_2, ... are formulae:*

$$\begin{aligned}
\tau(Body \rightarrow Head) &= \tau_{\mathbf{H}}(Body) \rightarrow \tau_{\mathbf{E}}(Head) \\
\tau_{\mathbf{H}}(A) &= \mathbf{H}(A) \\
\tau_{\mathbf{E}}(A) &= \mathbf{E}(A) \\
\tau_{\mathbf{M}}(F_1 \wedge F_2) &= \tau_{\mathbf{M}}(F_1) \wedge \tau_{\mathbf{M}}(F_2) \\
\tau_{\mathbf{M}}(false) &= false \\
\tau_{\mathbf{M}}(Y_i = Y_j) &= Y_i = Y_j \\
\tau_{\mathbf{E}}(\exists \mathbf{X}\ A) &= A
\end{aligned}$$

A Datalog$^{\pm}$ database D for $\mathcal{R}$ corresponds to the (possibly infinite) $\mathcal{S}$CIFF event set **HAP**, since there is a one-to-one correspondence between each tuple in D and each (ground) fact in **HAP**. This mapping is denoted as $\mathbf{HAP} = \tau_{\mathbf{H}}(D)$.

Notice that since the $\mathcal{S}$CIFF event set can dynamically grow, new constants can be introduced as a new event occurs (these new constants correspond to those in the set Δ_N of Datalog$^{\pm}$).

A Datalog TGD F of the kind $body \rightarrow head$ is mapped into the $\mathcal{S}$CIFF integrity constraint $IC = \tau(F)$, where the *body* is mapped into conjunctions of $\mathcal{S}$CIFF atoms, and *head* into conjunctions of $\mathcal{S}$CIFF abducible atoms. Existential quantifications of variables occurring in the *head* of the TGD are maintained in the head of the $\mathcal{S}$CIFF IC, but they are left implicit in the $\mathcal{S}$CIFF syntax, while the rest of the variables are universally quantified with scope the entire IC.

Given a set of TGDs T, let us denote the mapping of T into the corresponding set $\mathcal{IC}$ of $\mathcal{S}$CIFF integrity constraints, as $\mathcal{IC} = \tau(T)$.

Recall that for a set of TGDs T on $\mathcal{R}$, and a database D for $\mathcal{R}$, the set of models of D given T, denoted $mods(D, T)$, is the set of all (possibly infinite) databases B such that $D \subseteq B$ and every $F \in T$ is satisfied in B. For any such database B, we can prove that there exists an abductive explanation $\mathbf{EXP} = \tau_{\mathbf{E}}(B)$, $\mathbf{HAP}' = \tau_{\mathbf{H}}(B)$ such that:

$$\mathbf{HAP}' \cup \mathbf{EXP} \models \mathcal{IC}$$

where $\mathbf{HAP}' \supseteq \mathbf{HAP} = \tau_{\mathbf{H}}(D)$, and $\mathcal{IC} = \tau(T)$.

Finally, Datalog$^{\pm}$ negative constraints NC are mapped into $\mathcal{S}$CIFF ICs with head *false*, and equality-generating dependencies EGDs into $\mathcal{S}$CIFF ICs, each one with an equality CLP constraint in its head.

Therefore, informally speaking, the set of models of D given T, $mods(D, T)$, corresponds to the set of all the abductive explanations **EXP** satisfying the set of $\mathcal{S}$CIFF integrity constraints $\mathcal{IC} = \tau(\mathcal{T})$.

A Datalog$^{\pm}$ CQ $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ over $\mathcal{R}$ is mapped into a $\mathcal{S}$CIFF goal $G = \tau_{\mathbf{E}}(\Phi(\mathbf{X}, \mathbf{Y}))$, where $\tau_{\mathbf{E}}(\Phi(\mathbf{X}, \mathbf{Y}))$ is a conjunction of $\mathcal{S}$CIFF atoms. Notice that in the $\mathcal{S}$CIFF framework we have therefore a goal with existential variables only, and among them, we are interested in computed answer substitutions for the original (tuple of) variables $\mathbf{X}$ (and therefore $\mathbf{Y}$ variables can be made anonymous).

A Datalog$^{\pm}$ BCQ $q = \Phi(\mathbf{Y})$ is mapped similarly: $G = \tau_{\mathbf{E}}(\Phi(\mathbf{Y}))$.

Recall that in Datalog$^{\pm}$ the set of answers to a CQ q on D given T, denoted $ans(q, D, T)$, is the set of all tuples $\mathbf{t}$ such that $\mathbf{t} \in q(B)$ for all $B \in mods(D, T)$. With abuse of notation, we will write $q(\mathbf{t})$ to mean answer $\mathbf{t}$ for q on D given T.

We can hence state the following theorems for (model-theoretic) completeness of query answering.

Theorem 1 (Completeness of query answering). *For each answer $q(\mathbf{t})$ of a CQ $q(\mathbf{X}) = \exists \mathbf{Y} \Phi(\mathbf{X}, \mathbf{Y})$ on D given T, in the corresponding $\mathcal{S}$CIFF program $\langle \emptyset, \tau(\mathcal{IC}) \rangle$ there exists an answer substitution θ and an abductive explanation $\mathbf{EXP} \cup \mathbf{HAP}'$ for goal $G = \tau_{\mathbf{E}}(\Phi(\mathbf{X}, \_))$ such that:*

$$\langle \emptyset, \tau(\mathcal{IC}) \rangle \models^{g}_{\mathbf{HAP}} G\theta$$

where $\mathbf{HAP} = \tau_{\mathbf{H}}(D)$, $\mathcal{IC} = \tau(T)$, and $G\theta = \tau_{\mathbf{E}}(\Phi(\mathbf{t}, \_))$.

Corollary 1 (Completeness of boolean query answering). *If the answer to a BCQ $q = \exists \mathbf{Y} \Phi(\mathbf{Y})$ over D given T is Yes, denoted $D \cup T \models q$, then in the corresponding $\mathcal{S}$CIFF program there exists an abductive explanation* $\mathbf{EXP} \cup \mathbf{HAP}'$ *such that:*

$$\langle \emptyset, \tau(\mathcal{IC}) \rangle \models^{g}_{\mathbf{HAP}} G\theta$$

where $\mathbf{HAP} = \tau_{\mathbf{H}}(D)$, $\mathcal{IC} = \tau(T)$, *and* $G = \tau_{\mathbf{E}}(\Phi(\_))$.

The $\mathcal{S}$CIFF proof procedure has been proved sound w.r.t. $\mathcal{S}$CIFF declarative semantics in [4], therefore for each abductive explanation **EXP** for a given goal G in a $\mathcal{S}$CIFF program, there exists a $\mathcal{S}$CIFF-based computation producing a set of abducibles (positive expectations to our purposes) $\delta \subseteq \mathbf{EXP}$, and a computed answer substitution for goal G possibly more general than θ.

Example 2 (Real estate information extraction system in ALP). Let us conclude this section by re-considering the Datalog$^{\pm}$ ontology for the real estate information extraction system of Example 1. TGDs F_1-F_8 are one-to-one mapped into the following $\mathcal{S}$CIFF ICs:

$IC_1 : \mathbf{H}(ann(X, label)), \mathbf{H}(ann(X, price)), \mathbf{H}(visible(X)) \rightarrow \mathbf{E}(priceElem(X))$
$IC_2 : \mathbf{H}(ann(X, label)), \mathbf{H}(ann(X, priceRange)), \mathbf{H}(visible(X))$
$\quad \rightarrow \mathbf{E}(priceElem(X))$
$IC_3 : \mathbf{H}(priceElem(E)), \mathbf{H}(group(E, X)) \rightarrow \mathbf{E}(forSale(X))$
$IC_4 : \mathbf{H}(forSale(X)) \rightarrow (\exists P)\ \mathbf{E}(price(X, P))$
$IC_5 : \mathbf{H}(hasCode(X, C)), \mathbf{H}(codeLoc(C, L)) \rightarrow \mathbf{E}(loc(X, L))$
$IC_6 : \mathbf{H}(hasCode(X, C)) \rightarrow (\exists L)\ \mathbf{E}(codeLoc(C, L)), \mathbf{E}(loc(X, L))$
$IC_7 : \mathbf{H}(loc(X, L1)), \mathbf{H}(loc(X, L2)) \rightarrow L1 = L2$
$IC_8 : \mathbf{H}(loc(X, L)) \rightarrow \mathbf{E}(advertised(X))$

The database is then simply mapped into the following event set **HAP**:

$\{\mathbf{H}(codeLoc(ox1, central)), \mathbf{H}(codeLoc(ox1, south)),$
$\mathbf{H}(codeLoc(ox2, summertown)), \mathbf{H}(hasCode(prop1, ox2)), \mathbf{H}(ann(e1, price)),$
$\mathbf{H}(ann(e1, label)), \mathbf{H}(visible(e1)), \mathbf{H}(group(e1, prop1))\}$

The $\mathcal{S}$CIFF proof procedure applies ICs in a forward manner, and it infers the following set of abducibles from the program above:

$$\mathbf{EXP} = \{\mathbf{E}(priceElem(e1)), \mathbf{E}(forSale(prop1)), \exists P\ \mathbf{E}(price(prop1, P)),$$
$$\mathbf{E}(loc(prop1, summertown)), \mathbf{E}(advertised(prop1))\}$$

plus the corresponding **H** atoms, that are not reported for the sake of brevity.

Each of the (ground) atomic queries of Example 1 is entailed in the $\mathcal{S}$CIFF program above, since there exist sets **EXP** and $\mathbf{HAP}'$ such that:

$$\mathbf{HAP}' \cup \mathbf{EXP} \models \mathbf{E}(priceElem(e1)), \mathbf{E}(forSale(prop1)), \mathbf{E}(advertised(prop1))$$

The query $\exists L\ \mathbf{E}(loc(prop1, L))$ is entailed as well, considering the unification $L = summertown$ since:

$$\mathbf{HAP}' \cup \mathbf{EXP} \models \mathbf{E}(loc(prop1, summertown)).$$

It is worth noting that the $\mathcal{S}$CIFF framework is much more expressive than the restricted version used in this paper; in fact, in the mapping we used an empty KB, but in general the Knowledge Base can be a logic program, that can includes expectations, abducible literals, as well as CLP constraints. Beside the forward propagation of Integrity Constraints, $\mathcal{S}$CIFF supports also backward reasoning.

5 Related Work

Various approaches has been followed to reason upon ontologies.

Usually, DL reasoners implement a tableau algorithm using a procedural language. Since some tableau expansion rules are non-deterministic, the developers have to implement a search strategy from scratch.

Pellet [32] is a free open-source Java-based reasoner for SROIQ with simple datatypes (i.e., for OWL 1.1). It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering). It supports the OWL-API, the DIG interface, and the Jena interface and comes with numerous other features.

Pellet can compute the set of all the explanations for given queries by exploiting the tableau algorithm. An explanation is roughly a subset of the knowledge base (KB) that is sufficient for entailing the query. It applies Reiter's *hitting set algorithm* [29] to find all the explanations. This is a black box method: Pellet repeatedly removes an axiom from the KB and then computes again a new explanation exploiting the tableau algorithm on the new KB, recording all the different explanations so found.

Differently from Pellet, reasoners written in Prolog can exploit Prolog's backtracking facilities for performing the search. This has been observed in various works. In [8, 28] the authors proposed a tableau reasoner in Prolog for First Order Logic (FOL) based on free-variable semantic tableaux. However, the reasoner is not tailored to DLs.

Hustadt, Motik and Sattler [22] presented the KAON2 algorithm that exploits basic superposition, a refutational theorem proving method for FOL with equality, and a new inference rule, called decomposition, to reduce a $\mathcal{SHIQ}$ KB into a disjunctive Datalog program, while DLog [25, 33] is an ABox reasoning algorithm for the $\mathcal{SHIQ}$ language that allows to store the content of the ABox externally in a database and to answer instance check and instance retrieval queries by transforming the KB into a Prolog program.

Meissner presented the implementation of a reasoner for the DL $\mathcal{ALCN}$ written in Prolog [26] which was then extended and reimplemented in the Oz language [27]. Starting from [26], Herchenröder [20] implemented heuristic search techniques in order to reduce the inference time for the DL $\mathcal{ALC}$. Faizi [15] added to [20] the possibility of returning information about the steps executed during the inference process for queries but still handled only $\mathcal{ALC}$.

A different approach is the one by Ricca et al. [30] that presented OntoDLV, a system for reasoning on a logic-based ontology representation language called

OntoDLP. This is an extension of (disjunctive) ASP and can interoperate with OWL. OntoDLV rewrites the OWL KB into the OntoDLP language, can retrieve information directly from external OWL Ontologies and answers queries by using ASP.

TRILL [34, 35] adopts a Prolog-based implementation for the tableau expansion rules for $\mathcal{ALC}$ description logics. Differently from previous reasoners, TRILL is also able to return explanations for the given queries. Moreover, TRILL differs in particular from DLog for the possibility of answering general queries instead of instance check and instance retrieval only.

As reported in Section 2, reasoning upon Datalog$^{\pm}$ ontologies is achieved, instead, via the *chase* bottom-up procedure, which is exploited for deriving atoms entailed by a database and a Datalog$^{\pm}$ theory.

In this work, instead, we apply an abductive logic programming proof-procedure to reason upon ontologic data. It is worth to notice that in a previous work [1] the $\mathcal{S}$CIFF proof-procedure was interfaced with Pellet to perform ontological reasoning; in the current work, instead, $\mathcal{S}$CIFF is directly used to perform the reasoning by mapping atoms in the ontology to $\mathcal{S}$CIFF concepts (like events and expectations).

6 Conclusions and Future Work

In this paper, we addressed representation and reasoning for Datalog$^{\pm}$ ontologies in an Abductive Logic Programming framework, with existential (and universal) variables, and Constraint Logic Programming constraints in rule heads. The underlying proof procedure, named $\mathcal{S}$CIFF, is inspired by the IFF proof procedure, and had been implemented in Constraint Handling Rules [16]. The $\mathcal{S}$CIFF system has already been used for modeling and implementing several knowledge representation frameworks, also providing an effective reasoning system.

Here we have considered Datalog$^{\pm}$ ontologies, and shown how the $\mathcal{S}$CIFF language can be a useful knowledge representation and reasoning framework for them. In fact, the underlying abductive proof procedure can be directly exploited as an ontological reasoner for query answering and consistency check. To the best of our knowledge, this is the first application of ALP to model and reason upon ontologies.

Moreover, the considered $\mathcal{S}$CIFF language smoothly supports the integration of rules, expressed in a Logic Programming language, with ontologies expressed in Datalog$^{\pm}$, since a logic program can be added as (non-empty) KB to the set of ICs, therefore considering deductive rules besides the forward ICs themselves. Moreover, through dynamic acquisition of events in its **HAP** set, $\mathcal{S}$CIFF might also supports inline incrementality of the extensional part of the knowledge base (namely, the $\mathcal{A}$Box).

Many issues have not been addressed in this paper, and they will be subject of future work. First of all, we have not focused here on complexity results. Future work will be devoted to identify syntactic conditions guaranteeing tractable ontologies in $\mathcal{S}$CIFF, in the style of what has been done for Datalog$^{\pm}$.

A second issue for future work concerns experimentation and comparison with other approaches, even not Logic Programming (LP, for short) based, on real-size ontologies.

Finally, $\mathcal{S}$CIFF language is richer than the subset here used to represent Datalog$^{\pm}$ ontologies. It can support, in fact, negative expectations in rule heads, with universally quantified variables too, which basically represent the fact that something ought not to happen, and the proof procedure can identify violations to them.

Therefore, the richness of the language, and the potential of its abductive proof procedure pave the way to add further features to Datalog$^{\pm}$ ontologies.

References

1. Alberti, M., Cattafi, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Montali, M., Torroni, P.: A computational logic application framework for service discovery and contracting. International Journal of Web Services Research 8(3), 1–25 (2011)
2. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Montali, M.: An abductive framework for a-priori verification of web services. In: Maher, M. (ed.) Proceedings of the Eighth Symposium on Principles and Practice of Declarative Programming. pp. 39–50. ACM Press, New York, USA (Jul 2006)
3. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Security protocols verification in Abductive Logic Programming: a case study. In: Dikenelli, O., Gleizes, M.P., Ricci, A. (eds.) Proceedings of ESAW'05, Lecture Notes in Artificial Intelligence, vol. 3963, pp. 106–124. Springer-Verlag (2006)
4. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. ACM Transactions on Computational Logic 9(4) (2008)
5. Alberti, M., Gavanelli, M., Lamma, E.: The CHR-based implementation of the SCIFF abductive system. Fundamenta Informaticae 124(4), 365–381 (2013)
6. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Sartor, G., Torroni, P.: Mapping deontic operators to abductive expectations. Computational and Mathematical Organization Theory 12(2–3), 205 – 225 (Oct 2006)
7. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of agent interactions using social integrity constraints. Electronic Notes in Theoretical Computer Science 85(2) (2003)
8. Beckert, B., Posegga, J.: leanTAP: lean tableau-based deduction. J. Autom. Reasoning 15(3), 339–358 (1995)
9. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: International Conference on Principles of Knowledge Representation and Reasoning. pp. 70–80. AAAI Press (2008)
10. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Symposium on Principles of Database Systems. pp. 77–86. ACM (2009)
11. Calì, A., Gottlob, G., Lukasiewicz, T.: Tractable query answering over ontologies with Datalog$^{\pm}$. In: International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 477. CEUR-WS.org (2009)
12. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog$^{\pm}$: A family of logical knowledge representation and query languages for new applications. In:

IEEE Symposium on Logic in Computer Science. pp. 228–242. IEEE Computer Society (2010)
13. Calì, A., Gottlob, G., Pieris, A.: Tractable query answering over conceptual schemata. In: International Conference on Conceptual Modeling. LNCS, vol. 5829, pp. 175–190. Springer (2009)
14. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
15. Faizi, I.: A Description Logic Prover in Prolog. Bachelor's thesis, Informatics Mathematical Modelling, Technical University of Denmark (2011)
16. Frühwirth, T.: Theory and practice of constraint handling rules. Journal of Logic Programming 37(1-3), 95–138 (Oct 1998)
17. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming 33(2), 151–165 (Nov 1997)
18. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive query answering in probabilistic Datalog+/- ontologies. In: International Conference on Web Reasoning and Rule Systems. LNCS, vol. 6902, pp. 77–92. Springer (2011)
19. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The racerpro knowledge representation and reasoning system. Semantic Web 3(3), 267–277 (2012)
20. Herchenröder, T.: Lightweight Semantic Web Oriented Reasoning in Prolog: Tableaux Inference for Description Logics. Master's thesis, School of Informatics, University of Edinburgh (2006)
21. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRCPress (2009)
22. Hustadt, U., Motik, B., Sattler, U.: Deciding expressive description logics in the framework of resolution. Inf. Comput. 206(5), 579–601 (2008)
23. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming 19-20, 503–582 (1994)
24. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive Logic Programming. Journal of Logic and Computation 2(6), 719–770 (1993)
25. Lukácsy, G., Szeredi, P.: Efficient description logic reasoning in Prolog: The DLog system. TPLP 9(3), 343–414 (2009)
26. Meissner, A.: An automated deduction system for description logic with ALCN language. Studia z Automatyki i Informatyki 28-29, 91–110 (2004)
27. Meissner, A.: A simple distributed reasoning system for the connection calculus. Vietnam Journal of Computer Science 1(4), 231–239 (2014), http://dx.doi.org/10.1007/s40595-014-0023-8
28. Posegga, J., Schmitt, P.: Implementing semantic tableaux. In: DAgostino, M., Gabbay, D., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 581–629. Springer Netherlands (1999), http://dx.doi.org/10.1007/978-94-017-1754-0_10
29. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. 32(1), 57–95 (1987)
30. Ricca, F., Gallucci, L., Schindlauer, R., Dell'Armi, T., Grasso, G., Leone, N.: OntoDLV: An ASP-based system for enterprise ontologies. J. Log. Comput. 19(4), 643–670 (2009)
31. Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient owl reasoner. In: OWLED (2008)
32. Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2), 51–53 (2007)

33. Straccia, U., Lopes, N., Lukacsy, G., Polleres, A.: A general framework for representing and reasoning with annotated semantic web data. In: Fox, M., Poole, D. (eds.) Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press (2010), http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1590
34. Zese, R., Bellodi, E., Lamma, E., Riguzzi, F.: A description logics tableau reasoner in Prolog. In: Cantone, D., Asmundo, M.N. (eds.) CILC. CEUR Workshop Proceedings, vol. 1068, pp. 33–47. CEUR-WS.org (2013)
35. Zese, R., Bellodi, E., Lamma, E., Riguzzi, F., Aguiari, F.: Semantics and inference for probabilistic description logics. In: Bobillo, F., Carvalho, R.N., da Costa, P.C.G., d'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K.J., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) Uncertainty Reasoning for the Semantic Web III - ISWC International Workshops, URSW 2011-2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8816, pp. 79–99. Springer (2014)

Towards Fuzzy Granulation in OWL Ontologies

Francesca A. Lisi and Corrado Mencar

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy
{francesca.lisi,corrado.mencar}@uniba.it

Abstract. The integration of fuzzy sets in ontologies for the Semantic Web can be achieved in different ways. In most cases, fuzzy sets are defined by hand or with some heuristic procedure that does not take into account the distribution of available data. In this paper, we describe a method for introducing a granular view of data within an OWL ontology.

1 Introduction

Endowing OWL ontologies with capabilities of representing and processing imprecise knowledge is a highly desirable feature since the Semantic Web is full of imprecise and uncertain information coming from perceptual data (*i.e.*, data coming from subjective judgments), incomplete data, data with errors, etc. [16]. Moreover, even in the case that precise information is available, imprecise knowledge could be advantageous: tolerance to imprecision may lead to concrete benefits such as compact knowledge representation, and efficient and robust reasoning [20]. Additionally, humans continually acquire, manipulate and communicate imprecise knowledge: therefore any ontology capable of expressing imprecise knowledge, when a precise alternative leads to a complex representation, could be more *interpretable* by human users, i.e. easier to read and understand [1].

A number of mathematical tools are available to deal with imprecision and uncertainty in knowledge representation. The choice of the right tool depends on the type of imprecision. In particular, imprecision due to the lack of boundaries in concepts (such as coldness in the domain of indoor temperatures, interestingness of movies, etc.) are well modeled through fuzzy sets. In essence, fuzzy sets define collections of objects whose membership can be partial. Differently to probability measures, the degree of membership does not measure how likely an object is referred by a concept, but rather it quantifies how much the concept is applicable to the object. Fuzziness pervades human reasoning and allows it to intelligently act in complex environments: since fuzzy sets make possible a computational representation of concepts with no sharp boundaries, they enable machines to carry out human-centered information processing and reasoning [5].

The integration of fuzzy sets in ontologies for the Semantic Web can be achieved in different ways (see [19] for an updated overview). However, in most cases, fuzzy sets are defined by hand or with some heuristic procedure that does not take into account the distribution of available data. In this paper, we propose the adoption of a fuzzy clustering procedure to automatically acquire fuzzy sets

from data. Also, we exploit the resulting clusters, together with fuzzy quantifiers, to develop a granular view of the individuals in an OWL ontology.

The paper is structured as follows. Section 2 presents some preliminary information on Description Logics[1] (2.1), Fuzzy Set Theory (2.2), and Fuzzy DLs (2.3). Section 3 describes the proposed granulation method on OWL schemas at increasing levels of complexity. Finally, Section 4 draws some conclusive remarks along with future research directions.

2 Preliminaries

2.1 Description Logics

Description Logics (DLs) are a family of decidable First Order Logic (FOL) fragments that allow for the specification of structured knowledge in terms of classes (*concepts*), instances (*individuals*), and binary relations between instances (*roles*) [2]. Complex concepts (denoted with C) can be defined from atomic concepts (A) and roles (R) by means of the constructors available for the DL in hand. The members of the DL family differ from each other as for the set of constructors, thus for the complexity of concept expressions they can generate. For the sake of illustrative purposes, we present here a salient representative of the DL family, namely $\mathcal{ALC}$ [15], which is often considered to illustrate some new notions related to DLs. A DL *Knowledge Base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a pair where $\mathcal{T}$ is the so-called *Terminological Box* (TBox) and $\mathcal{A}$ is the so-called *Assertional Box* (ABox). The TBox is a finite set of *General Concept Inclusion* (GCI) axioms which represent is-a relations between concepts, whereas the ABox is a finite set of *assertions* (or *facts*) that represent instance-of relations between individuals (resp. couples of individuals) and concepts (resp. roles). Thus, when a DL-based ontology language is adopted, an ontology is nothing else than a TBox (*i.e.*, the intensional level of knowledge), and a populated ontology corresponds to a whole KB (*i.e.*, encompassing also an ABox, that is, the extensional level of knowledge).

The semantics of DLs can be defined directly with set-theoretic formalizations or through a mapping to FOL (as shown in [8]). Specifically, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a DL KB consists of a domain $\Delta^{\mathcal{I}}$ and a mapping function $\cdot^{\mathcal{I}}$. For instance, $\mathcal{I}$ maps a concept C into a set of individuals $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, *i.e.* $\mathcal{I}$ maps C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \{0, 1\}$ (either an individual belongs to the extension of C or does not belong to it). Under the *Unique Names Assumption* (UNA) [13], individuals are mapped to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. However UNA does not hold by default in DLs. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K}$ iff it satisfies all axioms and assertions in $\mathcal{T}$ and $\mathcal{A}$. In DLs a KB represents many different interpretations, *i.e.* all its models. This is coherent with the Open World Assumption (OWA) that holds in FOL semantics. A DL KB is *satisfiable* if it has at least one model. We also write $C \sqsubseteq_{\mathcal{K}} D$ if in any

[1] We recap that DLs are the logical foundation of the standard for web ontology languages belonging to the OWL 2 family [12].

Table 1. Syntax and semantics of constructs for $\mathcal{ALC}(\mathbf{D})$.

bottom (resp. top) concept	$\bot$ (resp. $\top$)	$\emptyset$ (resp. $\Delta^{\mathcal{I}}$)
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
abstract role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
concrete role	T	$T^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}}$
individual	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concrete value	v	$v^{\mathcal{I}} \in \Delta^{\mathbf{D}}$
concept intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
concept union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
universal abstract role restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential abstract role restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal concrete role restriction	$\forall T.\mathbf{d}$	$\{x \in \Delta^{\mathcal{I}} \mid \forall z\ (x,z) \in T^{\mathcal{I}} \rightarrow z \in \mathbf{d}^{\mathbf{D}}\}$
existential concrete role restriction	$\exists T.\mathbf{d}$	$\{x \in \Delta^{\mathcal{I}} \mid \exists z\ (x,z) \in T^{\mathcal{I}} \wedge z \in \mathbf{d}^{\mathbf{D}}\}$
general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept assertion	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
abstract role assertion	$(a,b) : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
concrete role assertion	$(a,v) : T$	$(a^{\mathcal{I}}, v^{\mathcal{I}}) \in T^{\mathcal{I}}$

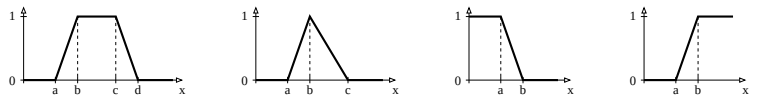
model $\mathcal{I}$ of $\mathcal{K}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (concept C is subsumed by concept D). Moreover we write $C \sqsubset_{\mathcal{K}} D$ if $C \sqsubseteq_{\mathcal{K}} D$ and $D \not\sqsubseteq_{\mathcal{K}} C$. The *consistency check*, which tries to prove the satisfiability of a DL KB $\mathcal{K}$, is the main reasoning task in DLs. It is performed by applying decision procedures mostly based on tableau calculus. All other reasoning tasks can be reformulated as consistency checks.

In many applications, it is important to equip DLs with expressive means that allow to describe "concrete qualities" of real-world objects such as the length of a car. The standard approach is to augment DLs with a so-called *concrete domain* (or *datatype theory*) $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$, which consists of a datatype domain $\Delta^{\mathbf{D}}$ (*e.g.*, the set of real numbers in double precision) and a mapping $\cdot^{\mathbf{D}}$ that assigns to each data value an element of $\Delta^{\mathbf{D}}$, and to every n-ary datatype predicate $\mathbf{d}$ an n-ary (typically, $n = 1$) relation over $\Delta^{\mathbf{D}}$ [3]. In DLs extended with concrete domains, each role is therefore either *abstract* (denoted with R) or *concrete* (denoted with T). The set of constructors for $\mathcal{ALC}(\mathbf{D})$ is reported in Table 1.

2.2 Fuzzy Set Theory

A *crisp set* A over a Universe of Discourse X is characterised by a function $A\colon X \rightarrow \{0,1\}$, that is, for any $x \in X$ either $x \in A$ (*i.e.*, $A(x) = 1$) or $x \notin A$ (*i.e.*, $A(x) = 0$). A *fuzzy set* F over X is characterised by a membership function $F\colon X \rightarrow [0,1]$. For a fuzzy set F, unlike crisp sets, $x \in X$ belongs to F to a degree $F(x)$ in $[0,1]$.

The trapezoidal, the triangular, the left-shoulder, and the right-shoulder functions are frequently used as *membership functions* of fuzzy sets (see Fig. 1 for a graphical representation). In particular, the *trapezoidal function* is defined



(a) (b) (c) (d)

Fig. 1. Four notable membership functions of fuzzy sets: (a) Trapezoidal, (b) triangular, (c) left-shoulder, and (d) right-shoulder.

as follows: let $a < b \leq c < d$ be rational numbers then

$$trz(a,b,c,d)(x) = \begin{cases} 0 & \text{if } x < a \\ (x-a)/(b-a) & \text{if } x \in [a,b) \\ 1 & \text{if } x \in [b,c] \\ (d-x)/(d-c) & \text{if } x \in (c,d] \\ 0 & \text{if } x > d\ . \end{cases} \tag{1}$$

Note that triangular, left-shoulder and right-shoulder fuzzy sets are special cases of trapezoidal fuzzy sets.[2]

Fuzzy sets can be used to represent *information granules*, *i.e.* collections of objects kept together due to their similarity, proximity, etc. [4]. Information granules promote abstraction as far as they can be labeled by symbolic terms. Information granules represented by fuzzy sets are good candidates to represent perceptual information, thus they could be conveniently labeled by linguistic terms coming from natural language. The granularity level (quantifiable as the area of the membership function, for fuzzy sets) assesses the specificity of an information granule: the most specific information granule is a set with a single element (precise information); on the other extreme, an information granule covering the whole universe of discourse is the least specific.

Fuzzy clustering Although fuzzy sets have a greater expressive power than crisp sets, their usefulness depends critically on the capability to construct appropriate membership functions for various given concepts in different contexts. The problem of constructing meaningful membership functions is not a trivial one (see, *e.g.*, [10, Chapter 10]). One easy method is to define a uniform *Strong Fuzzy Partition* (SFP) usually with 5 ± 2 fuzzy sets. A SFP is a collection of fuzzy sets (usually with triangular or trapezoidal fuzzy sets) such that, for each element of the Universe of Discourse, the sum of memberships of all fuzzy sets is always 1. SFPs with trapezoidal fuzzy sets greatly enhance the efficiency of calculations because they guarantee that each element has non-zero membership degree for at most two fuzzy sets. A uniform SFP is based on fuzzy sets with the same granularity. It is very simple to define a uniform SFP, but this approach does not take into account the distribution of available data; in fact, coarse

[2] By convention, whenever the denominator of one of the fractions in (1) is 0, the membership degree is 1.

grained fuzzy sets are more useful to cover regions of the Universe of Discourse where data are more sparse; on the other hand, data crammed in small areas are better represented by more specific (fine grained) fuzzy sets.

The derivation of a SFP with variable granularity, adapted to available data, can be achieved through fuzzy clustering. A widespread algorithm for fuzzy clustering is *Fuzzy C-Means* (FCM) [6], an extension of the well-known K-Means that accommodates partial memberships of data to clusters. FCM, applied to one-dimensional, numerical data, can be used to derive a set of c clusters characterized by prototypes $\pi_1, \pi_2, \ldots, \pi_c$, with $\pi_j \in \mathbb{R}$ and $\pi_j < \pi_{j+1}$. These prototypes, along with the range of data, provide enough information to define a SFP with two trapezoidal fuzzy sets and $c-2$ triangular fuzzy sets according to the following rules:

$$F_j = \begin{cases} trz\,(m, m, \pi_1, \pi_2) & \text{if } j = 1 \\ trz\,(\pi_{j-1}, \pi_j, \pi_j, \pi_{j+1}) & \text{if } 1 < j < c \\ trz\,(\pi_{c-1}, \pi_c, M, M) & \text{if } j = c. \end{cases} \tag{2}$$

where $[m, M]$ is the range of data. In Fig. 2 an example of SFP, consisting of five fuzzy sets with variable granularity, is depicted.

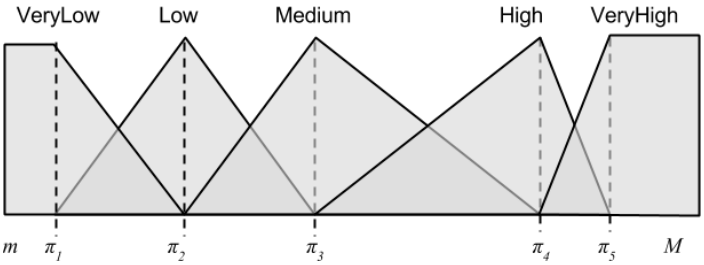


Fig. 2. Example of SFP consisting of $c = 5$ fuzzy sets with variable granularity.

Fuzzy quantifiers Fuzzy sets, like crisp sets, can be quantified in terms of their *cardinality*. Several definitions of cardinality of fuzzy sets are possible [9], although in this paper we consider only relative scalar cardinalities like the *relative σ-count*, defined for a finite Universe of Discourse X as:

$$\sigma(F) = \frac{\sum_{x \in X} F(x)}{|X|} \tag{3}$$

A relative scalar cardinality yields a value within $[0, 1]$ (being $|\emptyset| = 0$ and $|X| = 1$). On this interval, a number of fuzzy sets can be defined to represent granular concepts about cardinalities, such as `Many` (see Fig. 3 for some

notable examples). These concepts are called *fuzzy quantifiers*. Note that the usual existential quantifier ($\exists$) and universal quantifier ($\forall$) can be represented as special cases of fuzzy quantifiers: $Q_{\exists}(x) = 1$ iff $x > 0$, 0 otherwise; $Q_{\forall}(x) = 1$ iff $x = 1$, 0 otherwise.

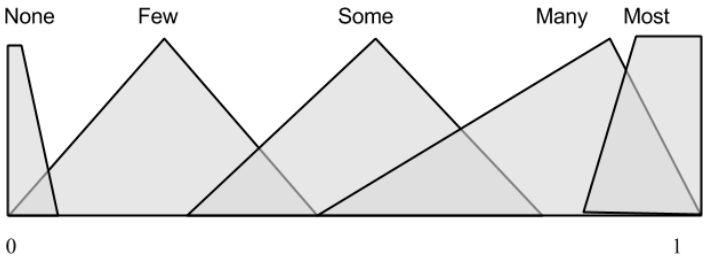


Fig. 3. Some notable fuzzy quantifiers: `None`, `Few`, `Some`, `Many`, and `Most`.

Given a fuzzy quantifier Q and a fuzzy set F, the membership degree $Q(\sigma(F))$ can be intended as the degree of truth of a *fuzzy proposition* of the form "Q`x are` F" (*e.g.*"`Many` x are `Tall`").

2.3 Fuzzy Description Logics

In fuzzy DLs, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty (crisp) set $\Delta^{\mathcal{I}}$ (the *domain*) and of a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that, *e.g.*, maps a concept C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$ and, thus, an individual belongs to the extension of C to some degree in $[0, 1]$, *i.e.* $C^{\mathcal{I}}$ is a fuzzy set. The definition of $\cdot^{\mathcal{I}}$ for $\mathcal{ALC}(\mathbf{D})$ with fuzzy concrete domains is reported in [18]. In particular, $\cdot^{\mathbf{D}}$ maps each concrete role into a function from $\Delta^{\mathbf{D}}$ to $[0, 1]$. Typical examples of datatype predicates are

$$\mathbf{d} := ls(a, b) \mid rs(a, b) \mid tri(a, b, c) \mid trz(a, b, c, d) \mid \geq_v \mid \leq_v \mid =_v , \qquad (4)$$

where *e.g.* $\geq_v$ corresponds to the crisp set of data values that are greater or equal than the value v.

Axioms in a fuzzy $\mathcal{ALC}(\mathbf{D})$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ are graded, *e.g.* a GCI is of the form $\langle C_1 \sqsubseteq C_2, \alpha \rangle$ (*i.e.* C_1 is a sub-concept of C_2 to degree at least α). We may omit the truth degree α of an axiom; in this case $\alpha = 1$ is assumed. An interpretation $\mathcal{I}$ *satisfies* an axiom $\langle \tau, \alpha \rangle$ if $(\tau)^{\mathcal{I}} \geq \alpha$. $\mathcal{I}$ is a *model* of $\mathcal{K}$ iff $\mathcal{I}$ satisfies each axiom in $\mathcal{K}$. We say that $\mathcal{K}$ *entails* an axiom $\langle \tau, \alpha \rangle$, denoted $\mathcal{K} \models \langle \tau, \alpha \rangle$, if any model of $\mathcal{K}$ satisfies $\langle \tau, \alpha \rangle$. Further details of the reasoning procedures for fuzzy DLs can be found in [17].

Fuzzy quantifiers have been also studied in fuzzy DLs. In particular, Sanchez and Tettamanzi [14] define an extension of fuzzy $\mathcal{ALC}(\mathbf{D})$ involving fuzzy quantifiers of the absolute and relative kind, and using qualifiers. They also provide algorithms for performing two important reasoning tasks with their DL: Reasoning about instances, and calculating the fuzzy satisfiability of a fuzzy concept.

3 Fuzzy Granulation of OWL Schemas

In this Section we show our proposal of introducing a granular view within an ontology. We shall proceed incrementally starting from the simplest case. For the sake of simplicity, we shall use the OWL terminology henceforth instead of the DL terminology (We remind the reader that class stands for concept, and property stands for role).

3.1 Case 1

Let C be a class and T a functional datatype property connecting instances of C to values in a numerical range **d**. See Fig. 4 for a graphical representation of this costruct.

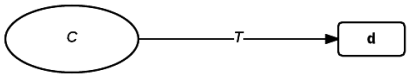


Fig. 4. Graphical representation of a functional datatype property T with domain C and range over a numerical datatype **d**.

This schema can be directly translated into a table (see Table 2) with two columns and as many rows as the number of individuals of C for which T holds.

Table 2. Tabular representation of the OWL schema depicted in Fig. 4.

C	T
a_1	v_1
a_2	v_2
$\cdots$	$\cdots$
a_n	v_n

The dataset in Table 2 can be easily granulated in a number of fuzzy sets $F_1, F_2, \ldots, F_c$ by applying, *e.g.*, the fuzzy clustering method mentioned in Section 2.2. In essence, the granulation process puts individuals in the same information granule if their respective values are similar. The use of fuzzy sets to define granules ensures a gradual membership degree of individuals to such granules, where the maximal membership is assigned to individuals detected as "prototypes" of each granule. Each fuzzy set represents a fuzzy concept, and can be tagged by a linguistic term, *e.g.* `Low`.

Table 3. Granulated individuals obtained from Table 2.

C	F_1	F_2	$\cdots$	F_c
a_1	μ_{11}	μ_{12}	$\cdots$	μ_{1c}
a_2	μ_{21}	μ_{22}	$\cdots$	μ_{2c}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
a_n	μ_{n1}	μ_{n2}	$\cdots$	μ_{nc}

The result of granulation can be represented in a new table (see Table 3), where each individual a_i is associated to a row of membership values μ_{ij}, being

$$\mu_{ij} = F_j(v_i) \tag{5}$$

For each granule F_j, the relative cardinality $\sigma(F_j)$ can be computed by means of the formula in Eq. (3). Given a fuzzy quantifier Q_k, the membership degree

$$q_{jk} = Q_k(\sigma(F_j)) \tag{6}$$

identifies the degree of truth of the fuzzy proposition "$Q_k x$ are F_j". In this way, a new table can be constructed from a collection $Q_1, Q_2, \ldots, Q_m$ of fuzzy quantifiers, as shown in Table 4. If $cm \ll n$, a sensible reduction of data can be achieved to represent the original property through a granulated view. (To further reduce data, a threshold τ can be set, so that all q_{jk} less than τ are set to zero.)

Table 4. Quantified cardinalities for the granules reported in Table 3.

	Q_1	Q_2	$\cdots$	Q_m
F_1	q_{11}	q_{12}	$\cdots$	q_{1m}
F_2	q_{21}	q_{22}	$\cdots$	q_{2m}
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
F_c	q_{c1}	q_{c2}	$\cdots$	q_{cm}

The new granulated view can be integrated in the ontology as follows. The fuzzy sets F_j are the starting point for the definition of new subclasses of C defined as $D_j \equiv C \sqcap \exists T.F_j$. Also, a new class `Granule` is defined, with individuals $g_1, g_2, \ldots, g_c$, where each individual g_j is an information granule corresponding to F_j. Each individual in D_j is then mapped to g_j by means of an object property `mapsTo`. Also, the cardinality of information granules is modeled by means of a datatype property `hasCardinality` with domain in `Granule` and range in the datatype domain `xsd:double`. Moreover, for each fuzzy quantifier Q_k, a new class is introduced, which models one of the fuzzy sets over the cardinalities of information granules. The connection between the class `Granule` and each class Q_k is established through `hasCardinality`, once fuzzified, with degrees identified as in Table 4. Note that the fuzzy proposition "$Q_k x$ are F_j" is then represented as the fuzzy assertion $g_j : \exists \texttt{hasCardinality}.Q_k$.

Example 1. In the tourism domain, we might consider an OWL ontology which encompasses the datatype property `hasPrice` with the class `Hotel` as domain and range in the datatype domain `xsd:double`. Let us suppose that the room price for Hotel Verdi (instance `verdi` of `Hotel`) is 105, *i.e.* the KB contains the assertion `(verdi, 105):hasPrice`. By applying fuzzy clustering to `hasPrice`, we might obtain three fuzzy sets (Low, `Medium`, `High`) from which the following classes are derived

$\texttt{LowPriceHotel} \equiv \texttt{Hotel} \sqcap \exists \texttt{hasPrice.Low}$
$\texttt{MidPriceHotel} \equiv \texttt{Hotel} \sqcap \exists \texttt{hasPrice.Medium}$
$\texttt{HighPriceHotel} \equiv \texttt{Hotel} \sqcap \exists \texttt{hasPrice.High}.$

With respect to these classes `verdi` shows different degrees of membership, *e.g.* `verdi` is a low-price hotel at degree 0.8 and a mid-price hotel at degree 0.2 (see Fig. 5 for a graphical representation). Subsequently, we might be interested in obtaining aggregated information about hotels. Here the class `Granule` comes into play. Quantified cardinalities allow us, for instance, to represent the fact that "Many hotels are low-price" as the fuzzy assertion $\texttt{lph} : \exists \texttt{hasCardinality.Many}$ with truth degree 0.7, where `lph` is an instance of `Granule` (*i.e.*, it is an information granule) which corresponds to `LowPriceHotel`, and `Many` is one of the fuzzy sets obtained from `hasCardinality`. Note that `verdi`, being an instance of `LowPriceHotel`, maps to `lph`, *i.e.* $(\texttt{verdi}, \texttt{lph}) : \texttt{mapsTo}$ holds to some degree.

3.2 Case 2

A natural extension of the proposed granulation method follows when the class C is specialized in subclasses, as in Fig. 6. In this case, there are as many tables with the same structure of Table 2 as the number of subclasses.

Analogously, for each subclass $SubC_j$ a structure of fuzzy information granules $F_{j1}, F_{j2}, \ldots, F_{jc}$ is produced and quantified according to the usual fuzzy quantifiers $Q_1, Q_2, \ldots, Q_m$. (The quantifiers do not depend on the subclass as their definition is fixed for all information granules.)

Example 2. Following Example 1, one may think of having a subsumption hierarchy with the class `Accommodation` as the root and `Hotel` and `B&B` as subclasses (see Fig. 7). Hotels are granulated in three fuzzy subclasses (`LowPriceHotel`, `MidPriceHotel` and `HighPriceHotel`) while B&Bs are granulated in two fuzzy subclasses (`CheapB&B` and `ExpensiveB&B`). These fuzzy classes are related to the classes representing fuzzy quantifiers via `Granule` analogously to Example 1.

3.3 Case 3

A case of particular interest is given by OWL schemas representing ternary relations. A *ternary relation* is a subset of the Cartesian product involving three domains $C \times D \times N$ (for our purposes, we will assume N a numerical domain).

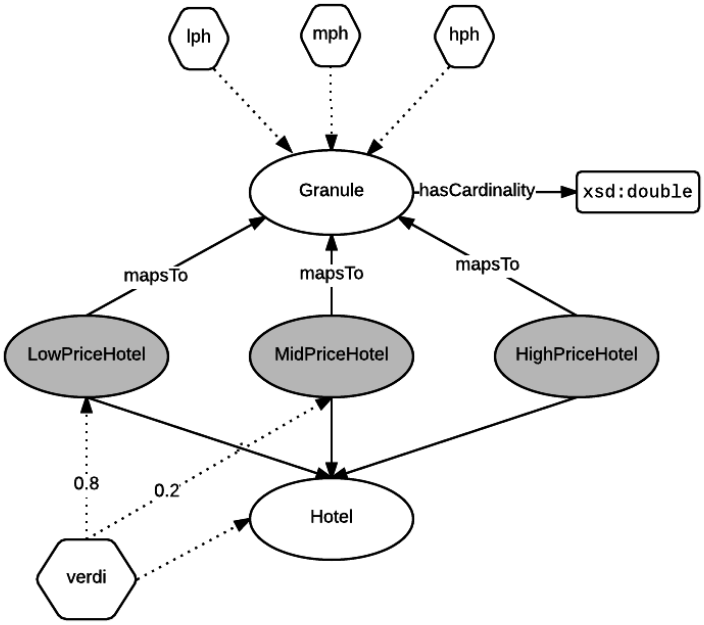


Fig. 5. Graphical representation of the output of the fuzzy granulation process on the OWL schema described in Fig. 4 and instantiated with concepts reported in Example 1. Fuzzy classes are depicted in gray.

Because of DL restrictions, however, ternary relations are not directly representable in OWL, yet they can be indirectly represented through an auxiliary class E, two object properties R_1 and R_2, and one datatype property T, as depicted in Fig. 8.

The structure in Fig. 8 corresponds to a tabular representation with three columns, and as many rows as the number of elements of the relation, as in table 5. By removing one of the two columns in Table 5, the resulting table is in accordance with Table 2, which was the starting point of the granulation process. In particular, as in the previous cases, a number of fuzzy sets $F_1, F_2, \ldots, F_c$ can be derived starting from the dataset represented in Table 5, where one column has been dropped. (We henceforth assume to drop column C.)

In order to connect information granules with classes, we proceed as follows. Given an individual $a \in C$, a subset of Table 5 can be obtained, as in Table 6.

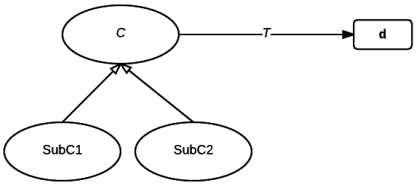


Fig. 6. Variant of the OWL schema shown in Fig. 4 for the case of C having subclasses.

Table 5. Tabular representation of the OWL schema depicted in Fig. 8.

C	D	T
a_1	b_1	v_1
$\cdots$	$\cdots$	$\cdots$
a_i	b_j	v_k
$\cdots$	$\cdots$	$\cdots$
a_n	b_m	v_l

More precisely, for each information granule F_j, it is possible to compute the relative cardinality

$$\sigma_{j_a} = \frac{\sum_{i=1}^{n_a} F_j\left(v_{a_i}\right)}{n_a} \tag{7}$$

Such cardinality can be quantified according to the fuzzy quantifiers $Q_1, \ldots, Q_m$. The result is a table similar to Table 4, but now related to the individual a.

Table 6. A slice of Table 5 obtained by fixing an individual a in C.

C	D	T
a	b_{a_1}	v_{a_1}
a	b_{a_2}	v_{a_2}
$\cdots$	$\cdots$	$\cdots$
a	$b_{a_{n_a}}$	$v_{a_{n_a}}$

Information granules, connected with the individuals in C, are arranged in the ontology in a way that merges the modeling of ternary relations as in Fig. 8 with the granular model illustrated in case 1. The new classes, representing information granules, are defined as $E'_j \equiv E \sqcap \exists R_2.D \sqcap \exists T.F_j$. They are connected to E in order to express a granular view of the relation between C and D. Finally, a natural extension of this case allows the specialization of the class D in subclasses (as in case 2).

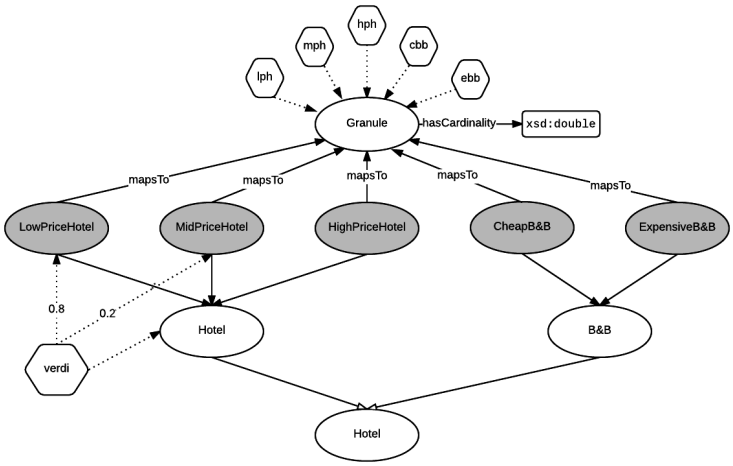


Fig. 7. Graphical representation of the output of the fuzzy granulation process on the OWL schema reported in Fig. 6 and instantiated with the concepts used in Example 2.

Example 3. With reference to the touristic domain, we might also consider the distances between hotels and attractions (see Fig. 9). This is clearly a case of ternary relation which requires to be modeled through an auxiliary class `Distance` which is connected to the classes `Hotel` and `Attraction` by means of the object properties `hasDistance` and `isDistanceFor`, respectively, and plays the role of domain for a datatype property `hasValue` with range `xsd:double`. The knowledge that "Hotel Verdi has a distance of 100 meters from the British Museum" can be therefore modeled as follows:

$(\texttt{verdi}, \texttt{d1}) : \texttt{hasDistance}$
$(\texttt{d1}, \texttt{british\_museum}) : \texttt{isDistanceFor}$
$(\texttt{d1}, 100) : \texttt{hasValue}$

After fuzzy granulation, the imprecise sentence "Hotel Verdi has a low distance from many attractions" can be considered as a consequence of the previous and the following axioms and assertions

$\texttt{LowDistance} \equiv \texttt{Distance} \sqcap \exists \texttt{isDistanceFor}.\texttt{Attraction} \sqcap \exists \texttt{hasValue}.\texttt{Low}$
$\texttt{d1} : \texttt{LowDistance}$ (to some degree)
$(\texttt{d1}, \texttt{ld}) : \texttt{mapsTo}$
$(\texttt{ld}, 0.5) : \texttt{hasCardinality}$
$\texttt{ld} : \exists \texttt{hasCardinality}.\texttt{Many}$ (to some degree)

where `Many` is defined as mentioned in Example 1.

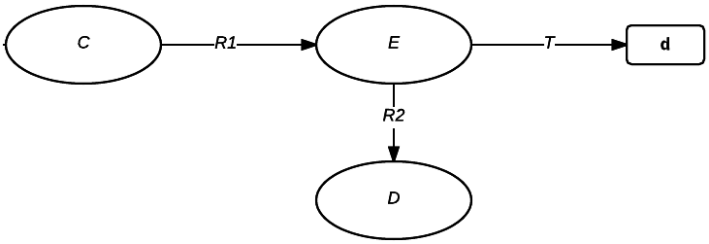


Fig. 8. Graphical representation of the OWL schema modeling a ternary relation.

4 Conclusions

This paper presents a starting point for introducing a granular view of data within an OWL ontology. According to the ideas presented in the paper, a number of individuals belonging to the ontology can be replaced by information granules, represented as fuzzy sets. In particular, the connection between the existing individuals (not involved in granulation) and the granulated view is possible by exploiting the peculiar representation of ternary relations in OWL.

This work is in a preliminary stage. We are currently evaluating the possibility of representing the output of our fuzzy granulation method by using OWL 2, *i.e.* within the current Semantic Web languages as suggested by Bobillo and Straccia in their proposal of Fuzzy OWL 2 [7]. In a certain sense, we are pursuing an alternative direction in comparison with the work of Sanchez and Tettamanzi [14] which, if implemented, could lead to the extension of current Semantic Web languages. However, it should be noted that there are some non-negligible restrictions to make their approach work in current fuzzy DLs. Notably, the approach considers a finite number of individuals, which causes a mismatch with the usual semantics for DLs (*i.e.*, OWA with infinite interpretations).

Future research is aimed at integrating our fuzzy granulation approach within inductive learning algorithms, such as Foil-$\mathcal{DL}$ [11], with the aim of verifying the benefits of information granulation in terms of efficiency and effectiveness of the learning process, as well as in terms of interpretability of the learning results.

Acknowledgements This work was partially funded by the *Università degli Studi di Bari "Aldo Moro"* under the IDEA Giovani Ricercatori 2011 grant "Dealing with Vague Knowledge in Ontology Refinement".

References

1. Alonso, J.M., Castiello, C., Mencar, C.: Interpretability of Fuzzy Systems: Current Research Trends and Prospects. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer

Handbook of Computational Intelligence. Springer Berlin / Heidelberg (2015)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications (2nd ed.). Cambridge University Press (2007)
3. Baader, F., Hanschke, P.: A scheme for integrating concrete domains into concept languages. In: Mylopoulos, J., Reiter, R. (eds.) Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991. pp. 452–457. Morgan Kaufmann (1991)
4. Bargiela, A., Pedrycz, W.: Granular computing: an introduction. Springer Science & Business Media (2003)
5. Bargiela, A., Pedrycz, W.: Human-centric information processing through granular modelling, vol. 182. Springer Science & Business Media (2009)
6. Bezdek, J.C.: Fuzzy clustering. In: Ruspini, E.H., Bonissone, P.P., Pedrycz, W. (eds.) Handbook of Fuzzy Computation, p. 2. Institute of Physics Pub. (1998)
7. Bobillo, F., Straccia, U.: Representing fuzzy ontologies in OWL 2. In: FUZZ-IEEE 2010, IEEE International Conference on Fuzzy Systems, Barcelona, Spain, 18-23 July, 2010, Proceedings. pp. 1–6. IEEE (2010)
8. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artificial Intelligence 82(1–2), 353–367 (1996)
9. Dubois, D., Prade, H.: Fuzzy cardinality and the modeling of imprecise quantification. Fuzzy sets and Systems 16(3), 199–230 (1985)
10. Klir, G.J., Yuan, B.: Fuzzy sets and fuzzy logic: theory and applications. Prentice-Hall, Inc. (1995)
11. Lisi, F.A., Straccia, U.: Dealing with incompleteness and vagueness in inductive logic programming. In: Cantone, D., Nicolosi Asmundo, M. (eds.) Proceedings of the 28th Italian Conference on Computational Logic, Catania, Italy, September 25-27, 2013. CEUR Workshop Proceedings, vol. 1068, pp. 179–193. CEUR-WS.org (2013), http://ceur-ws.org/Vol-1068/paper-l12.pdf
12. OWL 2 Web Ontology Language Document Overview: http://www.w3.org/TR/2009/REC-owl2-overview-20091027/. W3C (2009)
13. Reiter, R.: Equality and domain closure in first order databases. Journal of ACM 27, 235–249 (1980)
14. Sanchez, D., Tettamanzi, A.G.: Fuzzy quantification in fuzzy description logics. In: Sanchez, E. (ed.) Fuzzy Logic and the Semantic Web, Capturing Intelligence, vol. 1, pp. 135 – 159. Elsevier (2006)
15. Schmidt-Schauss, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
16. Stoilos, G., Simou, N., Stamou, G., Kollias, S.: Uncertainty and the Semantic Web. IEEE Intelligent Systems 21 (2006)
17. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
18. Straccia, U.: Description logics with fuzzy concrete domains. In: UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005. pp. 559–567. AUAI Press (2005)
19. Straccia, U.: Foundations of Fuzzy Logic and Semantic Web Languages. CRC Studies in Informatics Series, Chapman & Hall (2013)
20. Zadeh, L.A.: Is there a need for fuzzy logic? Information sciences 178(13), 2751–2779 (2008)

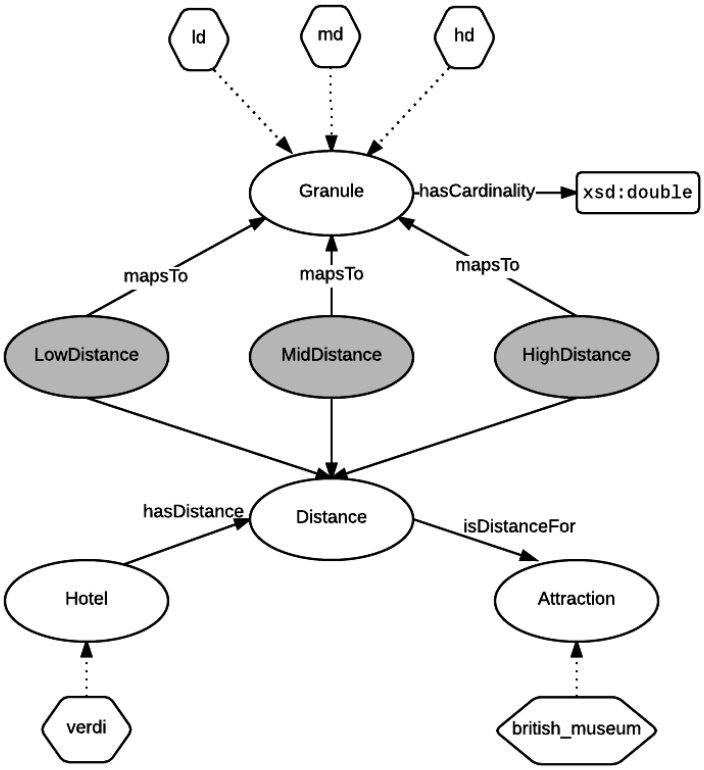


Fig. 9. Graphical representation of the output of the fuzzy granulation process on the OWL schema reported in Fig. 8 and instantiated with the concepts used in Example 3.

Preferential Description Logics meet Sports Entertainment: Cardinality Restrictions and Perfect Extensions for a Better Royal Rumble Match

Gian Luca Pozzato

Dipartimento di Informatica - Università di Torino - Italy
gianluca.pozzato@unito.it

Abstract. In this work we include cardinality restrictions and degrees of expectedness of inclusions in preferential Description Logics. We enrich the language of the nonmonotonic Description Logic *DL-Lite*$_c\mathbf{T}$, obtained by adding a typicality operator $\mathbf{T}$ to standard *DL-Lite*$_{core}$, by allowing inclusions of the form $\mathbf{T}(C) \sqsubseteq_d D$, where d is a degree of expectedness. We then propose a syntactic notion of extension of an ABox, in order to assume typicality assertions about individuals satisfying cardinality restrictions on concepts. Moreover, we define an order relation among such extended ABoxes, that allows to define a notion of *perfect extension* as the minimal one with respect to such an order relation. We apply this machinery to a problem coming from sports entertainment, namely the problem of maximizing the approval rating by the people attending to the Royal Rumble match, an annual wrestling event involving thirty athletes.

1 Introduction

The term *sports entertainment* has been coined by the World Wrestling Federation (now World Wrestling Entertainment, WWE, http://www.wwe.com) to describe professional wrestling, a combat sport combining athletics with theatrical performance. As a difference with typical athletics and games, which are conducted for competition, the main objective of sports entertainment, and especially of professional wrestling, is to entertain an audience. The owner of WWE, Vincent Kennedy McMahon, often mentions that his company has to "Give the People What They Want". Wrestling matches are driven by storylines provided by a creative team, and their outcomes are generally predetermined: duration, sequence of athletic moves, external interferences and, obviously, winners of the contests. Each athlete plays a specific role and follows a script, whereas injuries (and deaths) are only due to accidents, for instance because of a wrongly executed maneuver.

One of the most attractive events in professional wrestling is the WWE Royal Rumble match: thirty athletes are involved in this competition, and the winner receives a title shot in the main annual event of the company. The objective of each participant is to eliminate all the other competitors by tossing them over the top rope of the ring; an athlete is eliminated if both his feet touch the floor outside the ring. The match starts with the two participants who have drawn entry numbers one and two, with the remaining competitors entering the ring at regular timed intervals, usually 90 seconds, according to their entrance number assigned by means of a lottery. On the contrary, the assignment of entrance numbers to the participants, the sequence of eliminations (who eliminates

who), the last man being eliminated, as well as the winner himself are determined by the choices of the creative team and scheduled in all the details.

In the last two years, the Royal Rumble match has been marked by an extremely negative audience reaction: the trivial sequence of eliminations, as well as the fact that both the winners have been predicted before the match by professional wrestling web sites, lead the people in the arena to "boo" every single action of the show.

In this work we move a first step in order to tackle the problem of defining the script of the perfect Royal Rumble match. The idea it to support (not to replace) the creative team in the activities of selecting 1. the entrance number of the participants 2. the group of finalists, i.e. the last two or three athletes remaining in the ring after all other competitors have been eliminated 3. the winner of the match. To this aim, we exploit preferential Description Logics recently introduced in [13, 17, 19, 14].

Nonmonotonic extensions of Description Logics (DLs) have been actively investigated since the early 90s [5, 3, 6, 12, 13, 17, 10]. A simple but powerful nonmonotonic extension of DLs is proposed in [13, 17, 16, 14]: in this approach "typical" or "normal" properties can be directly specified by means of a "typicality" operator $\mathbf{T}$ enriching the underlying DL; the typicality operator $\mathbf{T}$ is essentially characterized by the core properties of nonmonotonic reasoning axiomatized by either *preferential logic* [20] or *rational logic* [21]. In these logics one can express defeasible inclusions such as "normally, a top player returning from an injury wins the Royal Rumble match":

$$\mathbf{T}(\mathit{Returning} \sqcap \mathit{Top}) \sqsubseteq \mathit{Winner}$$

As a difference with standard DLs, in these extensions one can consistently express exceptions and reason about defeasible inheritance as well. For instance, a knowledge base can consistently express that "normally, a face wrestler is supported by the crowd", whereas "typically, a face wrestler who is supposed to win the Royal Rumble match is not supported by the crowd" as follows:

$$\mathit{PredictedFace} \sqsubseteq \mathit{Face}$$
$$\mathbf{T}(\mathit{Face}) \sqsubseteq \mathit{Supported}$$
$$\mathbf{T}(\mathit{PredictedFace}) \sqsubseteq \neg \mathit{Supported}$$

The approach based on the typicality operator has been first introduced for the basic DL $\mathcal{ALC}$ [13]. In [14], the authors have extended this approach also to the logic *DL-Lite*$_{core}$ of the *DL-Lite* family. This logic is specifically tailored for effective query answering over DL knowledge bases containing a large amount of data, however, thanks to its computational complexity, it is considered a *lightweight* Description Logic: indeed, the problem of subsumption and the satisfiability of a knowledge base in *DL-Lite*$_{core}$ are NLOGSPACE in the size of the TBox [8, 9]. In this work, we restrict our concerns to the logic *DL-Lite*$_c\mathbf{T}$, whose expressive power is sufficient for the application to sports entertainment presented in Section 4.

The logic *DL-Lite*$_c\mathbf{T}$ results to be too weak in several application domains. Indeed, although the operator $\mathbf{T}$ is nonmonotonic ($\mathbf{T}(C) \sqsubseteq E$ does not imply $\mathbf{T}(C \sqcap D) \sqsubseteq E$), the logic *DL-Lite*$_c\mathbf{T}$ is monotonic, in the sense that if the fact F follows from a given knowledge base KB, then F also follows from any KB' $\supseteq$ KB. As a consequence, unless a KB contains explicit assumptions about typicality of individuals, there is no way of

inferring defeasible properties about them: in the above example, if KB contains the fact that Daniel is a face wrestler, i.e. $Face(daniel)$ belongs to KB, it is not possible to infer that he is supported by the crowd ($Supported(daniel)$). This would be possible only if the KB contained the stronger information that Daniel is a *typical* face wrestler, namely that $\mathbf{T}(Face)(daniel)$ belongs to KB.

In order to overwhelm this limit and perform useful inferences, in [17, 14] the authors have introduced a nonmonotonic extension of the logic *DL-Lite*$_c\mathbf{T}$ based on a minimal model semantics. Intuitively, the idea is to restrict our consideration to models that maximize typical instances of a concept when consistent with the knowledge base. The resulting logic, called *DL-Lite*$_c\mathbf{T}_{min}$, supports typicality assumptions, so that if one knows that Daniel is a face wrestler, one can nonmonotonically assume that he is also a *typical* face wrestler and therefore that he is supported by the crowd.

From a semantic point of view, the logic *DL-Lite*$_c\mathbf{T}_{min}$ is based on a preference relation among *DL-Lite*$_c\mathbf{T}$ models and a subsequent notion of *minimal entailment* restricted to models that are minimal with respect to such preference relation.

In several applications the assumptions of typicality in *DL-Lite*$_c\mathbf{T}_{min}$ seem to be too strong, for instance when the need arises of bounding the cardinality of the extension of a given concept, that is to say the number of domain elements being members of such a concept, as introduced in [4]. As an example, consider the following KB:

$\mathbf{T}(Face) \sqsubseteq Winner$
$\mathbf{T}(Returning) \sqsubseteq Winner$
$\mathbf{T}(Predicted) \sqsubseteq Winner$

If the assertional part of the KB contains the facts that:

$Face(daniel)$,
$Returning(dave)$,
$Predicted(roman)$

whose meaning is that Daniel is a face athlete, Dave is returning from an injury, and that Roman has been predicted to win the Royal Rumble match, respectively, then in *DL-Lite*$_c\mathbf{T}_{min}$ we conclude that

$\mathbf{T}(Face)(daniel)$
$\mathbf{T}(Returning)(daniel)$
$\mathbf{T}(Predicted)(roman)$

and then that Dave, Daniel and Roman are all winners. This happens in *DL-Lite*$_c\mathbf{T}_{min}$ because it is consistent to make the three assumptions above, that hold in all minimal models, however one should be interested in three distinct, but related aspects that cannot be captured by *DL-Lite*$_c\mathbf{T}_{min}$ as it is:

– first, one would like to restrict his attention to models/situations satisfying cardinality restrictions (in the example, there is only one winner, therefore the three assumptions above must be mutually exclusive);

– second, one could need to express different *degrees of expectedness* of typicality inclusions: for instance, normally a top face wrestler wins the Royal Rumble match, however this is in general more surprising with respect to the fact that, typically, a returning top wrestler wins. In other words, both the two inclusions represent typical properties, but the latter one seems to be more predictable;
– third, making all the consistent assumptions about prototypical properties should be in contrast with the need of taking into account a reasonable but "surprising enough" (or not obvious) scenario: in sports entertainment, a quite unpredictable script should help to obtain a positive reaction from the crowd.

In this work, we propose a new extension of the standard Description Logic *DL-Lite*$_{core}$ for reasoning about typicality called *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, whose aim is to restrict reasoning in *DL-Lite*$_c\mathbf{T}$ to "non trivial" scenarios respecting restrictions on the cardinality of concepts, in order to match the needs of proposing memorable scripts for events in sports entertainment. The original contribution of this work can be summarized as follows:

– we introduce a new Description Logic of typicality, called *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, allowing to express a degree of expectedness of typicality assumptions, that is to say TBoxes are extended by (i) inclusions of the form $\mathbf{T}(C) \sqsubseteq_d D$ where d is a positive integer, such that an inclusion with degree d is more "trivial" (or "obvious") with respect to another one with degree $d' \leq d$, as well as by (ii) restrictions on the cardinality of concepts;
– we introduce a notion of extension of an ABox for the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, corresponding to a set of typicality assumptions that can be performed in *DL-Lite*$_c\mathbf{T}_{min}$ for individual constants, then we introduce an order relation among extensions whose basic idea is to prefer extensions representing more surprising scenarios;
– we define notions of entailment in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, relying on existing reasoners for *DL-Lite*$_c\mathbf{T}$, but allowing to restrict our concern to "non trivial" scenarios, corresponding to minimal extensions with respect to the order relation among extensions of the previous point.

The plan of the paper is as follows. In Section 2 we briefly recall preferential DLs *DL-Lite*$_c\mathbf{T}$ and *DL-Lite*$_c\mathbf{T}_{min}$. In Section 3 we introduce the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, allowing to express degrees of expectedness of typicality inclusions as well as to deal with cardinality restrictions: we introduce notions of eligible and perfect extensions of an ABox for the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, allowing to describe a plausible, but unexpected scenario. In Section 4 we apply *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ in the context of sports entertainment to find a script for a better Royal Rumble match. Issues that will be object of future research are described in the concluding Section 5.

2 Preferential Description Logics *DL-Lite*$_c\mathbf{T}$ and *DL-Lite*$_c\mathbf{T}_{min}$

The logic *DL-Lite*$_c\mathbf{T}$ is obtained by adding to standard *DL-Lite*$_{core}$ the typicality operator $\mathbf{T}$ [14]. The intuitive idea is that $\mathbf{T}(C)$ selects the *typical* instances of a concept C. We can therefore distinguish between the properties that hold for all instances of concept C ($C \sqsubseteq D$), and those that only hold for the normal or typical instances of C ($\mathbf{T}(C) \sqsubseteq D$).

The language of *DL-Lite*$_c\mathbf{T}$ is defined as follows.

Definition 1. *We consider an alphabet of concept names $\mathcal{C}$, of role names $\mathcal{R}$, and of individual constants $\mathcal{O}$. Given $A \in \mathcal{C}$ and $S \in \mathcal{R}$, we define*

$$R := S \mid S^-$$
$$C_L := A \mid \exists R.\top \mid \mathbf{T}(A)$$
$$C_R := A \mid \neg A \mid \exists R.\top \mid \neg\exists R.\top$$

A DL-Lite$_c\mathbf{T}$ *KB is a pair (TBox, ABox). TBox contains a finite set of concept inclusions of the form $C_L \sqsubseteq C_R$. ABox contains assertions of the form $C(a)$ and $R(a, b)$, where C is a concept C_L or C_R, $R \in \mathcal{R}$, and $a, b \in \mathcal{O}$.*

In order to provide a semantics to the operator $\mathbf{T}$, the definition of a model $\mathcal{M} = \langle \Delta, I\rangle$ used in "standard" terminological logic *DL-Lite*$_{core}$, where Δ is the domain and I is a function mapping each concept C to its *extension* $C^I \subseteq \Delta$, is extended by a global preference relation among individuals of Δ: in this respect, $x < y$ means that x is "more normal" than y, and that the typical members of a concept C are the minimal elements of C with respect to this relation. In this framework, an element $x \in \Delta$ is a *typical instance* of some concept C if $x \in C^I$ and there is no C-element in Δ *more typical* than x. The typicality preference relation is partial. The basic idea is that the operator $\mathbf{T}$ is characterized by a set of postulates that are essentially a reformulation of the Kraus, Lehmann and Magidor's axioms of *preferential logic* **P** [20]. Intuitively, the assertion $\mathbf{T}(C) \sqsubseteq D$ corresponds to the conditional assertion $C \mathrel{|\!\sim} D$ of **P**. $\mathbf{T}$ has therefore all the "core" properties of nonmonotonic reasoning.

Definition 2 (Well-foundedness). *Given an irreflexive and transitive relation $<$ over Δ and $S \subseteq \Delta$, we define $Min_<(S) = \{x : x \in S$ and $\nexists y \in S$ s.t. $y < x\}$. We say that $<$ is well-founded if and only if, for all $S \subseteq \Delta$, for all $x \in S$, either $x \in Min_<(S)$ or $\exists y \in Min_<(S)$ such that $y < x$.*

Definition 3 (Multilinearity). *Given a preference relation $<$ over a domain Δ, we say that $<$ is multilinear if, for all $u, v, z \in \Delta$, if $u < z$ and $v < z$, then either $u = v$ or $u < v$ or $v < u$.*

Definition 4. *A model of* DL-Lite$_c\mathbf{T}$ *is any structure $\langle \Delta, <, I\rangle$, where: Δ is the domain; I is the extension function that maps each extended concept C to $C^I \subseteq \Delta$, and each role R to a $R^I \subseteq \Delta \times \Delta$; $<$ is an irreflexive, transitive, well-founded (Definition 2) and multilinear (Definition 3) relation over Δ. I is defined for atomic concepts $A \in \mathcal{C}$ end extended to complex concepts in the usual way (as for* DL-Lite$_{core}$*): $(\neg A)^I = \Delta \backslash A^I, (\exists S.\top)^I = \{x \in \Delta \mid \exists y \in \Delta$ and $(x, y) \in S^I\}, (\exists S^-.\top)^I = \{x \in \Delta \mid \exists y \in \Delta$ and $(y, x) \in S^I\}$; in addition, $(\mathbf{T}(C))^I = Min_<(C^I)$.*

Given a model $\mathcal{M}$ of Definition 4, I can be extended so that it assigns to each individual a of $\mathcal{O}$ a distinct element a^I of the domain Δ (unique name assumption). We say that $\mathcal{M}$ satisfies an inclusion $C \sqsubseteq D$ if $C^I \subseteq D^I$, and that $\mathcal{M}$ satisfies $C(a)$ if $a^I \in C^I$, $S(a, b)$ if $(a^I, b^I) \in S^I$, and $S^-(a, b)$ if $(b^I, a^I) \in S^I$. Moreover, $\mathcal{M}$ satisfies TBox

if it satisfies all its inclusions, and $\mathcal{M}$ satisfies ABox if it satisfies all its formulas. $\mathcal{M}$ satisfies a KB (TBox,ABox), if it satisfies both TBox and ABox.

We can also define a notion of entailment in *DL-Lite*$_c\mathbf{T}$. Given a query F (either an inclusion $C \sqsubseteq D$ or an assertion of the form $C(a)$ or an assertion of the form $R(a,b)$), we say that F is entailed from a KB in *DL-Lite*$_c\mathbf{T}$ if F holds in all *DL-Lite*$_c\mathbf{T}$ models satisfying KB, and we write KB $\models_{\textit{DL-Lite}_c\mathbf{T}} F$.

The semantics of the typicality operator can be specified by modal logic. The interpretation of $\mathbf{T}$ can be split into two parts: for any x of the domain Δ, $x \in (\mathbf{T}(C))^I$ just in case (i) $x \in C^I$, and (ii) there is no $y \in C^I$ such that $y < x$. Condition (ii) can be represented by means of an additional modality $\Box$, whose semantics is given by the preference relation $<$ interpreted as an accessibility relation. The interpretation of $\Box$ in $\mathcal{M}$ is as follows: $(\Box C)^I = \{x \in \Delta \mid \text{for every } y \in \Delta, \text{if } y < x \text{ then } y \in C^I\}$. We immediately get that $x \in (\mathbf{T}(C))^I$ if and only if $x \in (C \sqcap \Box\neg C)^I$.

Even if the typicality operator $\mathbf{T}$ itself is nonmonotonic (i.e. $\mathbf{T}(C) \sqsubseteq E$ does not imply $\mathbf{T}(C \sqcap D) \sqsubseteq E$), what is inferred from a KB can still be inferred from any KB' with KB $\subseteq$ KB'. In order to perform nonmonotonic inferences, in [17] the authors have strengthened the above semantics by restricting entailment to a class of minimal (or preferred) models. Intuitively, the idea is to restrict entailment to models that *minimize the untypical instances of a concept*. The resulting logic is called *DL-Lite*$_c\mathbf{T}_{min}$.

Given a KB, we consider a finite set $\mathcal{L}_\mathbf{T}$ of concepts: these are the concepts whose untypical instances we want to minimize. We assume that the set $\mathcal{L}_\mathbf{T}$ contains at least all concepts C such that $\mathbf{T}(C)$ occurs in the KB or in the query F. As we have already said, $x \in C^I$ is typical for C if $x \in (\Box\neg C)^I$. Minimizing the untypical instances of C therefore means to minimize the objects falsifying $\Box\neg C$ for $C \in \mathcal{L}_\mathbf{T}$. Hence, for a given model $\mathcal{M} = \langle \Delta, <, I\rangle$, we can define:

$$\mathcal{M}^{\Box^-}_{\mathcal{L}_\mathbf{T}} = \{(x, \neg\Box\neg C) \mid x \notin (\Box\neg C)^I, \text{ with } x \in \Delta, C \in \mathcal{L}_\mathbf{T}\}.$$

Definition 5 (Preferred and minimal models). *Given two models $\mathcal{M} = \langle \Delta, <, I\rangle$ and $\mathcal{M}' = \langle \Delta', <', I'\rangle$ of a knowledge base* KB*, we say that $\mathcal{M}$ is preferred to $\mathcal{M}'$ w.r.t. $\mathcal{L}_\mathbf{T}$, and we write $\mathcal{M} <_{\mathcal{L}_\mathbf{T}} \mathcal{M}'$, if (i) $\Delta = \Delta'$, (ii) $\mathcal{M}^{\Box^-}_{\mathcal{L}_\mathbf{T}} \subset \mathcal{M}'^{\Box^-}_{\mathcal{L}_\mathbf{T}}$, (iii) $a^I = a^{I'}$ for all $a \in \mathcal{O}$. $\mathcal{M}$ is a* minimal model *for* KB *(w.r.t. $\mathcal{L}_\mathbf{T}$) if it is a model of* KB *and there is no other model $\mathcal{M}'$ of* KB *such that $\mathcal{M}' <_{\mathcal{L}_\mathbf{T}} \mathcal{M}$.*

Definition 6 (Minimal Entailment in *DL-Lite*$_c\mathbf{T}_{min}$). *A query F is minimally entailed in* DL-Lite$_c\mathbf{T}_{min}$ *by* KB *with respect to $\mathcal{L}_\mathbf{T}$ if F is satisfied in all models of* KB *that are minimal with respect to $\mathcal{L}_\mathbf{T}$. We write* KB $\models_{\text{DL-Lite}_c\mathbf{T}_{min}} F$.

3 The Logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$: between *DL-Lite*$_c\mathbf{T}$ and *DL-Lite*$_c\mathbf{T}_{min}$

In this section we define an alternative semantics that allows us to express a degree of expectedness for the typicality inclusions and to limit the number of typicality assumptions in the ABox in order to obtain less predictable scenarios. The basic idea is similar to the one proposed in [13], where a completion of an $\mathcal{ALC} + \mathbf{T}$ ABox is proposed in order to assume that every individual constant of the ABox is a typical element of

the most specific concept he belongs to, if this is consistent with the knowledge base. Here we propose a similar, algorithmic construction in order to compute only *some* assumptions of typicality of domain elements/individual constants, in order to describe an alternative, surprising but not counterintuitive scenario, satisfying suitable constraints about the cardinality of the extensions of concepts. To this aim, we further extend a TBox with *cardinality restrictions* as defined in [4], that is to say axioms of the form either $(\geq n\ C)$ or $(\leq n\ C)$ or $(= n\ C)$, where n is a positive integer.

First of all, let us define the language $\mathcal{L}$ of the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$:

Definition 7. *We consider an alphabet of concept names $\mathcal{C}$, of role names $\mathcal{R}$, and of individual constants $\mathcal{O}$. Given $A \in \mathcal{C}$ and $S \in \mathcal{R}$, we define*

$R := S \mid S^-$
$C_R := A \mid \neg A \mid \exists R.\top \mid \neg\exists R.\top$

A DL-Lite$_c\mathbf{T}^{\mathsf{exp}}$ *KB is a pair (TBox, ABox). TBox contains axioms of the form:*

- $C_R \sqsubseteq C_R$*;*
- $\mathbf{T}(A) \sqsubseteq_d C_R$*, where $A \in \mathcal{C}$ and $d \in \mathbb{N}^+$ is called the* degree of expectedness*;*
- $(\geq n\ C_R)$*, where $n \in \mathbb{N}^+$;*
- $(\leq n\ C_R)$*, where $n \in \mathbb{N}^+$;*
- $(= n\ C_R)$*, where $n \in \mathbb{N}^+$.*

ABox contains assertions of the form $C(a)$ and $R(a, b)$, where C is a concept of C_R, $R \in \mathcal{R}$, and $a, b \in \mathcal{O}$.

3.1 Extensions of the ABox and order among extensions

Given an inclusion $\mathbf{T}(C) \sqsubseteq_d D$, the more the degree of expectedness is high, the more the inclusion is, in some sense, "obvious", not surprising. Given another inclusion $\mathbf{T}(C') \sqsubseteq_{d'} D'$, with $d' < d$, we assume that this inclusion is less "obvious", more surprising with respect to the other one. As an example, let KB contain $\mathbf{T}(Student) \sqsubseteq_4 SocialNetworkUser$ and $\mathbf{T}(Student) \sqsubseteq_2 PartyParticipant$, representing that typical students make use of social networks, and that normally they go to parties; however, the second inclusion is less obvious with respect to the first one.

Given a KB, we define a finite set $\mathbb{C}$ of concepts for the evaluation of typical properties. We assume that, for all $\mathbf{T}(C) \sqsubseteq_d D \in$ KB, then $C \in \mathbb{C}$.

Given an individual a explicitly named in the ABox, we define the set of "plausible" typicality assumptions $\mathbf{T}(C)(a)$ that can be minimally entailed from KB in the logic *DL-Lite*$_c\mathbf{T}_{min}$, with $C \in \mathbb{C}$. We then consider an ordered set of pairs (a, C) of all possible assumptions $\mathbf{T}(C)(a)$, for all concepts $C \in \mathbb{C}$ and all individual constants a occurring in ABox. This is formally stated in the next definition:

Definition 8 (Assumptions in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$). *Given a KB=(TBox,ABox) and the set of concepts $\mathbb{C}$, we define, for each individual name a occurring in ABox:*

$$\mathbb{C}_a = \{C \in \mathbb{C} \mid KB \models_{\text{DL-Lite}_c\mathbf{T}_{min}} \mathbf{T}(C)(a)\}$$

We also define $\mathbb{C}_{ABox} = \{(a, C) \mid C \in \mathbb{C}_a$ *and* a *occurs in ABox*$\}$ *and we impose an order on the elements of* $\mathbb{C}_{ABox}$*:*

$$\mathbb{C}_{ABox} =< (a_1, C_1), (a_2, C_2), \ldots, (a_n, C_n) > .$$

Furthermore, we define the ordered multiset*:*

$$d_{ABox} =< d_1, d_2, \ldots, d_n >$$

respecting the order imposed on $\mathbb{C}_{ABox}$*, where* $d_i = avg(\{d \in \mathbb{N}^+ \mid \mathbf{T}(C_i) \sqsubseteq_d D \in$ *TBox*$\})$.

Intuitively, the ordered multiset d_{ABox} contains tuples of the form $< d_1, d_2, \ldots, d_n >$, where d_i is the degree of expectedness of the assumption $\mathbf{T}(C)(a)$, such that $(a, C) \in \mathbb{C}_{\text{ABox}}$ at position i. d_i corresponds to the average of all the degrees d of typicality inclusions $\mathbf{T}(C) \sqsubseteq_d D$ in the TBox.

In order to define alternative scenarios, where not all plausible assumptions are taken into account, we consider different extensions of the ABox and we introduce an order among them, allowing to range from unpredictable to trivial ones. Starting from tuples $< d_1, d_2, \ldots, d_n >$ in d_{ABox}, the first step is to build all alternative tuples where 0 is used in place of some d_i to represent that the corresponding typicality assertion $\mathbf{T}(C)(a)$ is no longer assumed (Definition 9). Furthermore, we define the *extension* of the ABox corresponding to a string so obtained (Definition 10). To give an intuitive idea, before introducing the formal definitions, let us consider the following example:

Example 1. Given a KB, let the only typicality inclusions in TBox be $\mathbf{T}(C) \sqsubseteq_1 D$ and $\mathbf{T}(E) \sqsubseteq_2 F$. Let a and b be the only individual constants occurring in the ABox. Suppose also that (i) KB $\models_{\textit{DL-Lite}_c^{\mathbf{T}_{min}}} \mathbf{T}(C)(a)$, (ii) KB $\models_{\textit{DL-Lite}_c^{\mathbf{T}_{min}}} \mathbf{T}(C)(b)$, and (iii) KB $\models_{\textit{DL-Lite}_c^{\mathbf{T}_{min}}} \mathbf{T}(E)(b)$. We have that:

$\mathbb{C}_{\text{ABox}} = \{(a, C), (b, C), (b, E)\}$

$d_{\text{ABox}} =< 1, 1, 2 >$

Other possible tuples are: $< 0, 0, 2 >$, corresponding to extending the ABox with the only assumption $\mathbf{T}(E)(b)$; $< 0, 1, 0 >$, corresponding to extending the ABox with the only assumption $\mathbf{T}(C)(b)$; $< 1, 0, 0 >$, corresponding to extending the ABox with $\mathbf{T}(C)(a)$; $< 0, 1, 2 >$, corresponding to extending the ABox with the assumptions $\mathbf{T}(C)(b)$ and $\mathbf{T}(E)(b)$; $< 1, 0, 2 >$, corresponding to extending the ABox with $\mathbf{T}(C)(a)$ and $\mathbf{T}(E)(b)$; $< 1, 1, 0 >$, corresponding to extending the ABox with $\mathbf{T}(C)(a)$ and $\mathbf{T}(C)(b)$; $< 0, 0, 0 >$, corresponding to not extending the ABox (the set of typicality assumptions is empty).

Let us now introduce formal definitions for the above mentioned notions of string of plausible assumptions and of extension of an ABox corresponding to a string.

Definition 9 (Strings of plausible assumptions $\mathcal{S}$). *Given a KB=(TBox,ABox) and the set* $\mathbb{C}_{ABox}$*, let* $d_{ABox} =< d_1, d_2, \ldots, d_n >$ *be the ordered multiset of Definition 8. We define the set* $\mathcal{S}$ *of all the* strings of plausible assumptions *with respect to KB as*

$$\mathcal{S} = \{< s_1, s_2, \ldots, s_n > \mid \forall i = 1, 2, \ldots, n \textit{ either } s_i = d_i \textit{ or } s_i = 0\}$$

Definition 10 (Extension of the ABox). *Let KB=(TBox,ABox) and let* $\mathbb{C}_{ABox} =< (a_1, C_1), (a_2, C_2), \ldots, (a_n, C_n) >$ *as in Definition 8. Given a string of plausible assumptions* $< s_1, s_2, \ldots, s_n > \in \mathcal{S}$ *of Definition 9, we define the extension* $\widehat{ABox}$ *of the ABox corresponding to the string as*

$$\widehat{ABox} = \{\mathbf{T}(C_i)(a_i) \mid (a_i, C_i) \in \mathbb{C}_{ABox} \text{ and } s_i \neq 0\}$$

It is easy to observe that, in *DL-Lite*$_c\mathbf{T}_{min}$, the set of typicality assumptions that can be inferred from a KB corresponds to the extension of the ABox corresponding to the string d_{ABox}, that is to say no element is set to 0: all the typicality assertions of individuals occurring in the ABox, that are consistent with the KB, are assumed. This corresponds to the "most obvious" situation. On the contrary, in *DL-Lite*$_c\mathbf{T}$, no typicality assumptions can be derived from a KB, and this corresponds to extending the ABox by the assertions corresponding to the string $< 0, 0, \ldots, 0 >$, i.e. by the empty set. This corresponds to the most surprising situation. Between them, all the other strings of $\mathcal{S}$ (Definition 9), corresponding to alternative extensions of the ABox, that we propose to order as follows:

Definition 11 (Order between extensions). *Given a KB=(TBox,ABox) and the set* $\mathcal{S}$ *of strings of plausible assumptions (Definition 9), let* $s =< s_1, s_2, \ldots, s_n >$ *and* $r =< r_1, r_2, \ldots, r_n >$*, with* $s, r \in \mathcal{S}$*. Furthermore, let* $\widehat{ABox_s}$ *and* $\widehat{ABox_r}$ *be the extensions of the ABox corresponding to* s *and* r *(Definition 10), respectively. We say that* $s \leq r$ *if there exists a bijection* δ *between* s *and* r *such that, for each* $(s_i, r_j) \in \delta$*, it holds that* $s_i \leq r_j$*, and there is at least one* $(s_i, r_j) \in \delta$ *such that* $s_i < r_j$*. We say that* $\widehat{ABox_s}$ *is* more surprising *(or* less trivial*) than* $\widehat{ABox_r}$ *if* $s \leq r$*.*

Intuitively, a string s whose elements are "lower" than the ones of another string r corresponds to a less trivial ABox. For instance, recalling Example 1, let us consider the strings $s =< 1, 1, 0 >$ and $r =< 1, 0, 2 >$, we have that $s \leq r$, because there exists a bijection $\{(1, 1), (0, 0), (1, 2)\}$ whose pairs (s_i, r_i) are such that $s_i \leq r_i$. The assumptions $\mathbf{T}(C)(a)$ and $\mathbf{T}(C)(b)$ corresponding to s are then considered less trivial than $\mathbf{T}(C)(a)$ and $\mathbf{T}(E)(b)$ corresponding to r. It is worth noticing that the order of Definition 11 is partial: as an example, the strings $< 1, 1 >$ and $< 0, 2 >$ are not comparable, in the sense that neither $< 1, 1 > \leq < 0, 2 >$ nor $< 0, 2 > \leq < 1, 1 >$. In order to choose between two incomparable situations, we introduce the following notion of weak order. Intuitively, the idea is as follows: given two incomparable extensions $\widehat{\text{ABox}_s}$ and $\widehat{\text{ABox}_r}$, we assume that $\widehat{\text{ABox}_s}$ is weakly less trivial than $\widehat{\text{ABox}_r}$ if $\widehat{\text{ABox}_r}$ is strictly included in another extension $\widehat{\text{ABox}_u}$ more trivial than $\widehat{\text{ABox}_s}$.

Definition 12 (Weak preference). *Given a KB=(TBox,ABox), let* $\widehat{ABox_s}$ *and* $\widehat{ABox_r}$ *be two extensions of the ABox such that neither* $\widehat{ABox_s}$ *is more surprising than* $\widehat{ABox_r}$ *nor* $\widehat{ABox_r}$ *is more surprising than* $\widehat{ABox_s}$*. We say that* $\widehat{ABox_s}$ *is* (weakly) more surprising *(or* (weakly) less trivial*) than* $\widehat{ABox_r}$ *if there exists an extension* $\widehat{ABox_u}$ *of ABox such that (i)* $\widehat{ABox_s}$ *is more surprising than* $\widehat{ABox_u}$ *(Definition 11) and (ii)* $\widehat{ABox_r} \subset \widehat{ABox_u}$*.*

As an example, let

$\widehat{\text{ABox}_s} = \{\mathbf{T}(C)(a)\}$,
$\widehat{\text{ABox}_r} = \{\mathbf{T}(D)(b)\}$,
$\widehat{\text{ABox}_u} = \{\mathbf{T}(D)(b), \mathbf{T}(E)(b)\}$

be three extensions of the ABox of a given KB=(TBox,ABox), corresponding to $s =< 1, 0, 0 >$, $r =< 0, 1, 0 >$, and $u =< 0, 1, 2 >$, respectively. We have that $s =< 1, 0, 0 >$ and $r =< 0, 1, 0 >$ are not comparable with respect to the relation $\leq$. However, we have that (i) $s \leq u$ and that (ii) $\widehat{\text{ABox}_r} \subset \widehat{\text{ABox}_u}$, therefore we conclude that $\widehat{\text{ABox}_s}$ is *(weakly) more surprising* (or *(weakly) less trivial*) than $\widehat{\text{ABox}_r}$.

3.2 Cardinality restrictions on concepts and perfect extensions

In general, it could be useful to restrict logical entailment to models in which the cardinality of the extensions of some concepts is bounded. More expressive DLs allow to specify (un)qualified number restrictions, in order to specify the number of possible elements filling a given role R. As an example, number restrictions allow to express that a student attends to 3 courses. Number restrictions are therefore "localized to the fillers of one particular role" [4], for instance we can have $Student \sqsubseteq \geq 3 Attends.Course$ as a restriction on the number of role fillers of the role $Attends$. However one could need to express global restrictions on the number of domain elements belonging to a given concept, for instance to express that in the whole domain there are exactly 3 courses. In DLs not allowing cardinality restrictions one can only express that every student must attend to three courses, but not that all must attend to the same ones.

In the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, cardinality restrictions on concepts are added to the TBox as in Definition 7. They are expressions of the form either $(\geq n\ C)$ or $(\leq n\ C)$ or $(= n\ C)$, where n is a positive integer and C is an extended concept.

Definition 13. *Given a* DL-Lite$_c\mathbf{T}$ *model* $\mathcal{M} = \langle \Delta, <, I \rangle$*, where* I *is extended so that it assigns to each individual* a *of* $\mathcal{O}$ *a distinct element* a^I *of the domain* Δ *(unique name assumption), we say that* $\mathcal{M}$ *satisfies:*

- *(elements of a TBox)*
  - *an inclusion* $C \sqsubseteq D$ *if* $C^I \subseteq D^I$*;*
  - *a typicality inclusion* $\mathbf{T}(C) \sqsubseteq_d D$ *if* $Min_<(C^I) \subseteq D^I$*;*
  - *a cardinality restriction of the form* $(\geq n\ C)$ *if* $\sharp C^I \geq n$
  - *a cardinality restriction of the form* $(\leq n\ C)$ *if* $\sharp C^I \leq n$
  - *a cardinality restriction of the form* $(= n\ C)$ *if* $\sharp C^I = n$
- *(elements of an ABox)*
  - *an assertion of the form* $C(a)$ *if* $a^I \in C^I$
  - *an assertion of the form* $R(a, b)$ *if* $(a^I, b^I) \in R^I$.

Given a KB=($\mathcal{T} \cup \mathcal{C}$*,ABox), where* $\mathcal{T}$ *is a set of inclusions and* $\mathcal{C}$ *is a set of axioms of cardinality restrictions, we say that a model* $\mathcal{M}$ *satisfies KB if it satisfies all the inclusions in* $\mathcal{T}$*, all the axioms of cardinality restrictions in* $\mathcal{C}$ *and all the assertions in ABox.*

Given a KB=(TBox,ABox), we say that an extension of ABox is an *eligible extension* if it admits a *DL-Lite*$_c\mathbf{T}$ model as in Definition 13:

Definition 14 (Eligible extension $\widehat{\text{ABox}}$). *Given a* DL-Lite$_c\mathbf{T}$ *KB=(TBox,ABox) and an extension $\widehat{ABox}$ of ABox as in Definition 10, we say that $\widehat{ABox}$ is* eligible *if there exists a* DL-Lite$_c\mathbf{T}$ *model $\mathcal{M}$ that satisfies KB'=(TBox, ABox $\cup$ $\widehat{ABox}$).*

Definition 15 (Minimal (perfect) extensions). *Given a KB=(TBox,ABox) and the set $\mathcal{S}$ of strings of plausible assumptions (Definition 9), we say that an eligible extension $\widehat{ABox_s}$ is minimal if there is no other eligible extension $\widehat{ABox_r}$ which is* (weakly) more surprising *(or* (weakly) less trivial*) than it.*

Given the above definitions, we can define a notion of entailment in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$. Intuitively, given a query F, we check whether F follows in the monotonic logic *DL-Lite*$_c\mathbf{T}$ from a given KB, whose ABox is augmented with extensions that are minimal (perfect) as in Definition 15. We can reason either in a skeptical way, by allowing that F is entailed if it follows in *all* KBs, obtained by considering each minimal extension of the ABox, or in a credulous way, by assuming that F is entailed if there exists at least one extension of the ABox allowing such inference. This is stated in a rigorous manner by the following definition:

Definition 16 (Entailment in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$). *Given a KB=(TBox,ABox) and given $\mathbb{C}$ a set of concepts, let $\mathcal{E}$ the set of all extensions of ABox that are minimal as in Definition 15. Given a query F, we say that (i) F is skeptically entailed from KB in* DL-Lite$_c\mathbf{T}^{exp}$*, written KB $\models^{sk}_{\text{DL-Lite}_c\mathbf{T}^{exp}} F$, if (TBox, ABox $\cup$ $\widehat{ABox}$) $\models_{\text{DL-Lite}_c\mathbf{T}} F$ for all $\widehat{ABox} \in \mathcal{E}$; (ii) F is credulously entailed from KB in* DL-Lite$_c\mathbf{T}^{exp}$*, written KB $\models^{cr}_{\text{DL-Lite}_c\mathbf{T}^{exp}} F$, if there exists $\widehat{ABox} \in \mathcal{E}$ such that (TBox, ABox $\cup$ $\widehat{ABox}$) $\models_{\text{DL-Lite}_c\mathbf{T}} F$.*

Let us conclude this section with an example of how the proposed approach works.

Example 2. Let us recall and simplify the example of the Introduction. Consider a KB=(TBox,ABox) where TBox is as follows:

$\mathbf{T}(Face) \sqsubseteq_1 Winner$
$\mathbf{T}(Predicted) \sqsubseteq_2 Winner$
$\mathbf{T}(Returning) \sqsubseteq_3 Winner$

expressing that, normally, a returning athlete wins the Royal Rumble match, and this is more predictable with respect to the fact that an athlete whose victory has been predicted, typically wins the match. Furthermore, normally a face wrestler wins the Royal Rumble match, but this inclusion is the most unexpected among the ones belonging to the KB. ABox contains the following facts about Dean, Roman, and Dave:

$Face(dean)$
$Face(roman)$
$Predicted(roman)$
$Returning(dave)$

Moreover, the TBox is enriched by the cardinality restriction $(= 1\ Winner)$, i.e. we restrict our concern to models in which there is only one winner.

Let $\mathbb{C} = \{Face, Predicted, Returning\}$. By Definition 8 above, we have that:

$\mathbb{C}_{dean} = \{Face\}$
$\mathbb{C}_{roman} = \{Face, Predicted\}$
$\mathbb{C}_{dave} = \{Returning\}$

and, obviously, $\mathbb{C}_{\text{ABox}} = \mathbb{C}_{dean} \cup \mathbb{C}_{roman} \cup \mathbb{C}_{dave}$. Concerning the degrees of expectedness, we have:

$$d_{\text{ABox}} =< 1, 1, 2, 3 >$$

The leftmost 1 is due to the fact that $\mathbf{T}(Face) \sqsubseteq_1 Winner$ belongs to the TBox, and we have that in *DL-Lite*$_c\mathbf{T}_{min}$ one can assume that Dean is a $\mathbf{T}(Face)$. Similarly, the other 1 is due to the fact the in *DL-Lite*$_c\mathbf{T}_{min}$ we can assume $\mathbf{T}(Face)(roman)$. Similarly, we have 2 in the multiset d_{ABox}, by the presence of $\mathbf{T}(Predicted) \sqsubseteq_2 Winner$ in the TBox and the fact that $\mathbf{T}(Predicted)(roman)$ is minimally entailed from the KB. Last, 3 is justified by the presence of $\mathbf{T}(Returning) \sqsubseteq_3 Winner$ and the fact that we can assume $\mathbf{T}(Returning)(dave)$.

As mentioned above, in *DL-Lite*$_c\mathbf{T}_{min}$ the minimal model semantics forces all the consistent typicality assumptions, namely we are considering an ABox extended with the following facts:

$\mathbf{T}(Face)(dean)$
$\mathbf{T}(Face)(roman)$
$\mathbf{T}(Predicted)(roman)$
$\mathbf{T}(Returning)(dave)$

corresponding (in the sense of Definition 10) to the multiset $< 1, 1, 2, 3 >$. However, from the resulting KB, in *DL-Lite*$_c\mathbf{T}$ we obtain that Dean, Roman and Dave are all winners, against the fact that we want to have only one winner: the extension corresponding to $< 1, 1, 2, 3 >$ is indeed not eligible in the sense of Definition 14.

In order to find only one winner and to obtain a non-trivial outcome of the match, let us consider the set $\mathcal{S}$ of all plausible strings of typicality assumptions (Definition 9):

$$\mathcal{S} = \{< 1, 1, 2, 3 >, < 0, 1, 2, 3 >, < 1, 0, 2, 3 >, < 1, 1, 0, 3 >, < 1, 1, 2, 0 >, < 0, 0, 2, 3 >, < 1, 0, 0, 3 >, < 0, 1, 0, 3 >, < 1, 0, 2, 0 >, < 0, 1, 2, 0 >, < 1, 1, 0, 0 >, < 0, 0, 0, 3 >, < 0, 0, 2, 0 >, < 1, 0, 0, 0 >, < 0, 1, 0, 0 >, < 0, 0, 0, 0 >\}$$

The only eligible extensions of ABox, corresponding to the above strings are:

$\widehat{\text{ABox}_1} = \{\mathbf{T}(Returning)(dave)\}$, corresponding to $< 0, 0, 0, 3 >$
$\widehat{\text{ABox}_2} = \{\mathbf{T}(Predicted)(roman)\}$, corresponding to $< 0, 0, 2, 0 >$
$\widehat{\text{ABox}_3} = \{\mathbf{T}(Face)(dean)\}$, corresponding to $< 1, 0, 0, 0 >$
$\widehat{\text{ABox}_4} = \{\mathbf{T}(Face)(roman)\}$, corresponding to $< 0, 1, 0, 0 >$
$\widehat{\text{ABox}_5} = \{\mathbf{T}(Face)(roman), \mathbf{T}(Predicted)(roman)\}$, corresponding to $< 0, 1, 2, 0 >$

We aim at choosing the less trivial scenario. To this aim, we observe that $\widehat{\text{ABox}_3}$ and $\widehat{\text{ABox}_4}$ are less trivial than $\widehat{\text{ABox}_5}$, because $< 1, 0, 0, 0 > \leq < 0, 1, 2, 0 >$ and $< 0, 1, 0, 0 > \leq < 0, 1, 2, 0 >$. Furthermore, $\widehat{\text{ABox}_3}$ and $\widehat{\text{ABox}_4}$ are less trivial than $\widehat{\text{ABox}_1}$ (again, $< 1, 0, 0, 0 > \leq < 0, 0, 0, 3 >$ and $< 0, 1, 0, 0 > \leq < 0, 0, 0, 3 >$).

$\mathbf{T}(Predicted) \sqsubseteq_4 Winner$	$\mathbf{T}(MidCarder) \sqsubseteq_4 FastExit$	$Heel(wyatt)$
$\mathbf{T}(Face) \sqsubseteq_1 Winner$	$\mathbf{T}(MidHeel) \sqsubseteq_1 \neg EarlyEntrance$	$Returning(bryan)$
$\mathbf{T}(Returning) \sqsubseteq_4 Winner$	$\mathbf{T}(MidFace) \sqsubseteq_4 EarlyEntrance$	$Face(bryan)$
$\mathbf{T}(BigMan) \sqsubseteq_4 Final$	$\mathbf{T}(Heel) \sqsubseteq_2 EarlyEntrance$	$Heel(kane)$
$\mathbf{T}(Face) \sqsubseteq_3 Final$	$\mathbf{T}(Face) \sqsubseteq_2 EarlyEntrance$	$BigMan(kane)$
$\mathbf{T}(Heel) \sqsubseteq_2 Final$	$\mathbf{T}(BodyBuilder) \sqsubseteq_3 FastExit$	$Predicted(reigns)$
$MidFace \sqsubseteq Face$	$(= \ 2 \ EarlyEntrance)$	$Face(reigns)$
$MidHeel \sqsubseteq Heel$	$(\geq \ 2 \ Final)$	$BigMan(bigshow)$
$MidHeel \sqsubseteq MidCarder$	$(\leq \ 3 \ Final)$	$Face(ryback)$
$MidFace \sqsubseteq MidCarder$	$(= \ 1 \ Winner)$	$BodyBuilder(ryback)$
		$Face(ziggler)$

Fig. 1. A portion of the KB in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ adopted for the application to sports entertainment.

Moreover, $\widehat{\text{ABox}_3}$ and $\widehat{\text{ABox}_4}$ are less trivial than $\widehat{\text{ABox}_2}$ (again, $< 1,0,0,0 > \leq < 0,0,0,2 >$ and $< 0,1,0,0 > \leq < 0,0,0,2 >$). The strings $< 1,0,0,0 >$ and $< 0,1,0,0 >$ are not comparable, however $\widehat{\text{ABox}_3}$ is weakly less trivial than $\widehat{\text{ABox}_4}$, since $\widehat{\text{ABox}_4} \subset \widehat{\text{ABox}_5}$ and $< 1,0,0,0 > \leq < 0,1,2,0 >$. This allows to conclude that $\widehat{\text{ABox}_3}$ is minimal (the perfect extension) and to suggest that Dean has to be chosen as the winner of the Royal Rumble match.

4 *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ meets Sports Entertainment: a Better Royal Rumble match

In Figure 1 we present a small portion of the simple ontology with exceptions we have considered in order to suggest an alternative, possibly better script of the Royal Rumble 2015. We have considered athletes involved in the Royal Rumble 2015, that took place on January 25, 2015 at the Wells Fargo Center in Philadelphia, Pennsylvania: Roman Reigns, widely predicted, won the contest. One minimal/perfect extension suggests the following alternative script: the match starts with a returning, face athlete, Daniel Bryan, and another face superstar, Ryback. Dolph Ziggler is the winner, with Bray Wyatt being the last athlete eliminated. We have then asked over 30 wrestling experts and fans about the result, and all of them found it very interesting and significantly better than the original one. Obviously, this is only a preliminary feedback, we aim at taking care of a more precise evaluation of the quality (in terms of non triviality) of the scenario proposed by adopting the machinery described in this work.

5 Conclusions and Future Issues

We have moved a first step in the direction of an alternative semantics for preferential Description Logics and its application in the context of sports entertainment. We have introduced the Description Logic of typicality *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$, an extension of *DL-Lite*$_{core}$ with a typicality operator $\mathbf{T}$ allowing to:

– express typicality inclusions of the form $\mathbf{T}(A) \sqsubseteq_d B$, where d is a positive integer representing a degree of expectedness;
– reason in presence of restrictions on the cardinality of concepts;
– perform plausible inferences in presence of alternative, non-trivial scenarios.

We are currently working on studying the complexity of standard reasoning tasks in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$. We have chosen the logic *DL-Lite*$_c\mathbf{T}_{min}$ as the base logic of our approach also thanks to its computational properties: in [14] the authors have shown that minimal entailment in *DL-Lite*$_c\mathbf{T}_{min}$ is in Π_2^p. We strongly conjecture that adding cardinality restrictions and the machinery for finding the perfect extension of an ABox is absorbed by the complexity of reasoning in *DL-Lite*$_c\mathbf{T}_{min}$, and is therefore inexpensive. We also intend to develop and implement proof methods for reasoning in the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ of optimal complexity.

One limit of the proposed approach is that the computation of the extension of the ABox only applies to individuals explicitly named in the knowledge base. This is one of the limits of the completion of an $\mathcal{ALC} + \mathbf{T}$ ABox in [13] as well. We aim at extending our work in order to also consider the individuals introduced by the existential restrictions (e.g. $(\exists HasSon.\top)(bob)$, the son of Bob). To this aim, we can define the assumptions in *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ on domain elements rather than on individual constants. In our approach, we first consider all possible typicality assumptions that are minimally entailed in *DL-Lite*$_c\mathbf{T}_{min}$ from a KB *without* cardinality restrictions, and then we restrict our concern to models satisfying cardinality restrictions and the ABox extended with such assumptions. We aim at studying an alternative approach in which cardinality restrictions are directly expressed in the initial KB, and the notion of preference among extensions of the ABox is replaced by a preference relation of expectedness among models, thus allowing to consider domain elements not explicitly named in the ABox.

The above mentioned alternative approach suggests the opportunity of studying extensions of Description Logics of typicality with restrictions on the cardinality of concepts. This task is of its own interest. As far as we know, no other non-monotonic extension of DLs (DLS+default rules [3], DLs+circumscription [5], DLs+ Lifschitz's nonmonotonic logic MKNF [12, 22], DLs+rational closure [11, 18, 19]) has been extended to reason in presence of cardinality constraints.

In this work we have tried to tackle a problem coming from sports entertainment, however the logic *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ of typicality with degree of expectedness can find several alternative applications. As an example, it could be applied in healthcare and medical diagnosis, where ontologies with exceptions should be useful for reasoning about defeasible inheritance (e.g. normally, the heart is positioned in the left-hand side of the chest, however people with situs inversus have the heart positioned in the right-hand side). Sometimes, the "obvious" diagnosis given a set of symptoms is not the right one: the semantics of *DL-Lite*$_c\mathbf{T}^{\mathsf{exp}}$ could be used in order to formulate a "mystery" diagnosis, alternative to the standard one.

As a further direction, we aim at extending our approach to other Description Logics. On the one hand, we want to take into account other lightweight DLs, for instance the logics of the $\mathcal{EL}$ family, allowing for conjunction ($\sqcap$) and (qualified) existential restriction ($\exists R.C$). Despite their relatively low expressivity, they are relevant for several applications, in particular in the bio-medical domain; for instance, small extensions

of $\mathcal{EL}$ can be used to formalize medical terminologies, such as the GALEN Medical Knowledge Base, the Systemized Nomenclature of Medicine, and the Gene Ontology used in bioinformatics. In [1, 2, 7] it is shown that reasoning in $\mathcal{EL}$ and several of its extensions remains tractable (i.e., polynomial-time decidable) in the presence of the TBox, and even of general concept inclusions (GCIs). An extension $\mathcal{EL}^{\bot}\mathbf{T}_{min}$ with the typicality operator has been introduced in [14]. On the other hand, we want to consider more expressive Description Logics, in particular the logics underlying the standard language for ontology engineering OWL. In this direction, it seems to be promising an alternative approach to non-monotonic semantics for the typicality operator based on a notion of rational closure in DLs [18]. As a difference with the semantics adopted in this paper, the semantics in [15, 18] exploits an alternative notion of preference among models, based on the idea of minimizing the rank of objects in the domain (that is, their level of "untypicality"), rather than minimizing the $\neg\Box\neg C$-elements in the models. This alternative way of minimizing typicality has the nice property that corresponds to a simple reformulation of rational closure for the Description Logic $\mathcal{ALC}$ [18]. A similar correspondence has also been proved for the more expressive $\mathcal{SHIQ}$ in [19].

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03). pp. 325–330 (2003)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L., Saffiotti, A. (eds.) Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005). pp. 364–369. Professional Book Center, Edinburgh, Scotland, UK (August 2005)
3. Baader, F., Hollunder, B.: Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic. Journal of Automated Reasoning (JAR) 15(1), 41–68 (1995)
4. Baader, F., Buchheit, M., Hollunder, B.: Cardinality restrictions on concepts. Artificial Intelligence 88(1-2), 195–213 (1996)
5. Bonatti, P.A., Lutz, C., Wolter, F.: DLs with Circumscription. In: KR. pp. 400–410 (2006)
6. Bonatti, P.A., Faella, M., Sauro, L.: Defeasible inclusions in low-complexity dls: Preliminary notes. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 696–701 (2009)
7. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, gci axioms, andwhat else? In: de Mántaras, R.L., Saitta, L. (eds.) Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004). pp. 298–302. IOS Press (2004)
8. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: DL-Lite: Tractable Description Logics for Ontologies. In: Veloso, M., Kambhampati, S. (eds.) Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference. pp. 602–607. AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA (July 2005)
9. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. Journal of Automated Reasoning (JAR) 39(3), 385–429 (2007)
10. Casini, G., Straccia, U.: Rational closure for defeasible description logics. In: Janhunen, T., Niemelä, I. (eds.) Logics in Artificial Intelligence - Proceedings of the12th European Conference (JELIA 2010). Lecture Notes in Computer Science (LNCS), vol. 6341, pp. 77–90. Springer (2010)

11. Casini, G., Straccia, U.: Defeasible Inheritance-Based Description Logics. Journal of Artificial Intelligence Research (JAIR) 48, 415–473 (2013)
12. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logics (ToCL) 3(2), 177–225 (2002)
13. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: ALC+T: a preferential extension of description logics. Fundamenta Informaticae 96, 341–372 (2009)
14. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Reasoning about typicality in low complexity DLs: the logics $\mathcal{EL}^{\bot}\mathbf{T}_{min}$ and *DL-Lite*$_c\mathbf{T}_{min}$. In: Walsh, T. (ed.) Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 894–899. IOS Press, Barcelona, Spain (2011)
15. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A minimal model semantics for nonmonotonic reasoning. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) Logics in Artificial Intelligence - Proceedings of the 13th European Conference (JELIA 2012). Lecture Notes in Artificial Intelligence (LNAI), vol. 7519, pp. 228–241. Springer-Verlag, Toulouse, France (September 2012)
16. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Preferential vs Rational Description Logics: which one for Reasoning About Typicality? . In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010). FAIA (Frontiers in Artificial Intelligence and Applications), vol. 215, pp. 1069 – 1070. IOS Press, Lisbon, Portugal (August 2010)
17. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: A NonMonotonic Description Logic for Reasoning About Typicality. Artificial Intelligence 195, 165 – 202 (2013)
18. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Minimal model semantics and rational closure in description logics. In: Eiter, T., Glimm, B., Kazakov, Y., Krötzsch, M. (eds.) Informal Proceedings of the 26th International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1014, pp. 168–180. CEUR-WS.org (2013)
19. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Rational closure in $\mathcal{SHIQ}$. In: DL 2014, 27th International Workshop on Description Logics. CEUR Workshop Proceedings, vol. 1193, pp. 543–555. CEUR-WS.org (2014)
20. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
21. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55(1), 1–60 (1992)
22. Motik, B., Rosati, R.: Reconciling Description Logics and rules. Journal of the ACM 57(5) (2010)

Games with Additional Winning Strategies*

Vadim Malvone, Aniello Murano, and Loredana Sorrentino

Università degli Studi di Napoli Federico II

Abstract. In game theory, deciding whether a designed player wins a game corresponds to check whether he has a winning strategy. There are situations in which it is important to know whether some extra winning strategy also exists. In this paper we investigate this question over two-player finite games, under the reachability objective. We provide an automata-based technique that, given such a game, it allows to decide in linear time whether the game admits more than a winning strategy. We discuss along the paper some case studies and use them to show how to apply our solution methodology.

1 Introduction

Game theory is a very powerful mathematical framework with several useful applications in different fields. In economics, it is used to deal with solution concepts such as Nash equilibrium [18]. In computer science, it is applied to solve problems in robotics, multi-agent system verification and planning [1, 15, 20–22].

In the basic setting, a game consists of two players playing in a turn-based manner, i.e, the moves of the players are interleaved. Solving a two-player game amounts to check whether one of the players has a *winning strategy*. That is, he can use a sequence of moves (a *strategy*) that makes him to satisfy the game target, no matter how his opponent plays. In several settings, however, having instead a more precise (quantitative) answer would be beneficial. For example, in planning a rescue, it would be useful to know whether a robot team has more than a winning strategy from a critical stage, just to have a backup plan in case the scenario changes during the rescue. Such a redundancy allows to strengthen the ability of winning the game and therefore its safety.

In this paper, we address the quantitative question of checking whether a player has more than a strategy to win a two-player finite game G. We investigate this problem under the *reachability* target and show an automata-based solution to solve it in linear time. Precisely, we build an automaton that accepts only trees that are witnesses of more than one winning strategy for the designed player over the game G. Hence, we reduce the addressed quantitative question to the emptiness of this automaton. To give an evidence of our approach, we report on some cooperative and adversarial game examples.

Related works. Counting strategies has been deeply exploited in the formal verification setting [3,5,7–9,13] by means of infinite duration games. The automata construction we use here takes inspiration from the ones used in [2, 9, 19].

* This paper is partially supported by the FP7 EU project 600958-SHERPA.

2 Case Studies

In this section we consider two different case studies of two-player games. In the first case the players behave adversarial. In the second one, they are cooperative.

Cop and Robber Game. Assume we have a maze where a cop aims to catch a robber, while the latter, playing adversarial, aims for the opposite. For simplicity, we assume the maze to be a grid divided in rooms, each of them named by its coordinates in the plane (see Figure 1). Each room can have one or more doors that allow the robber and the cop to move from one room to another. Each door has associated a direction along with it can be crossed. Both the cop and the robber can enter every room. The cop, being in a room, can physically block only one of its doors. The robber can move in another room if there is a non-blocked door he can take, placed between the two rooms, with the right direction. The robber wins the game if he can reach one of the safe places (EXIT) situated in the four corners of the maze. Otherwise, the robber is blocked in a room or he can never reach a safe place, and thus the cop wins the game. We assume that both the cop and the robber are initially siting in the middle of the maze, that is in the room $(1, 1)$. Starting from the maze depicted in Figure 1, one can see that the robber has only one strategy to win the game. Consider now two orthogonal variations of the maze. For the first one, consider flipping the direction of the door d_{12}. In this case, the robber loses the game. As second variation, consider flipping the direction of the door d_4. Then the robber wins the game and he has now two strategies to accomplish it.

Escape Game. Assume we have an arena similar to the one described in the previous example, but now with a cooperative interaction between two players, a human and a controller, aiming at the same target. Precisely, consider the arena depicted in Figure 2 representing a building where a fire is occurring. The building consists of rooms and, as before, each room has one-way doors and its position is determined by its coordinates. We assume that there is only one exit in the corner $(2, 2)$. One can think of this game as a simplified version of an automatic control station that starts working after an alarm fire occurs and all doors have been closed. Accordingly, we assume that the two players play in turn and at the starting moment all doors are closed. At each control turn, he opens one door of the room in which the human is staying. The human turn consists of taking one of the doors left open if its direction is in accordance with the move. We assume that there is no communication between the players, but the move. We start the game with the human siting in the room $(0, 0)$ and the controller moving first. It is not hard to see that the human can reach the exit trough the doors d_1, d_4, d_7, d_{10} opened by the controller. Actually, this is the only possible way the human has to reach the exit. Conversely, if we consider the scenario in which the direction of the door d_3 is flipped, then there are two strategies to let the human to reach the exit. Therefore, the latter scenario can be considered as better (i.e., more robust) than the former. Clearly, this extra information can be used to improve an exit fire plan at its designing level.

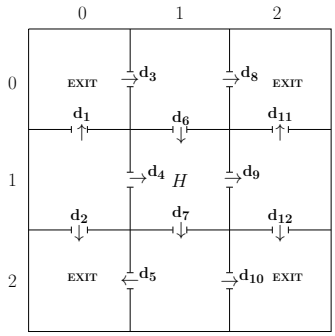


Fig. 1. Cop and Robber Game.

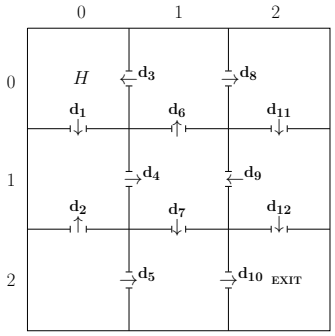


Fig. 2. Escape Game.

3 The Game Model

In this paper, we consider two-player turn-based games that are suitable to represent the case studies we have introduced in the previous section. Precisely, we consider games consisting of an arena and a target. The arena describes the configurations of the game through a set of states, being partitioned between the two players. In each state, only the player that owns it can take a move. This kind of interaction is also known as token-passing. About the target, we consider the reachability objective, that is some states are declared *target*. A winning strategy for a designed player is a path from the initial state to a target state. If such a winning strategy exists we say that the player wins the game. Clearly, the player has more than a winning strategy if there are different paths reaching a target state. The formal definition of the considered game model follows.

Definition 1. *A* turn-based two-player reachability game *(2TRG, for short), played between Player* 0 *and Player* 1*, is a tuple* $G \triangleq < \mathrm{St},\ s_I,\ \mathsf{tr},\ \mathrm{W} >$*, where* St *is a finite non-empty set of* states*, partitioned in* St_0 *and* St_1 *with* St_i *being the set of states of Player* i*,* $s_I \in \mathrm{St}$ *is a designated* initial state*,* W *is a set of target states, and* $\mathsf{tr} \subseteq \mathrm{St}_i \times \mathrm{St}_{1-i}$*, for* $i \in \{0, 1\}$ *is a* transition function *mapping a state of a player to a state belonging to the other player.*

The previous two case studies can be easily modeled using a *2TRG*. We now give some details. As set of states we use all the rooms in the maze, together with the status of their doors. For example, the state $((0,0), \{d_1^c, d_3^c\})$ is the initial state of the *Escape Game* where d_i^c means that the door d_i is closed. For an open door, instead, we will use the label o in place of c. Formally, let $D_{i,j}$ be the set of doors (up to four) belonging to the room (i, j), which can be flagged either with c (closed) or o (open), then we set $\mathrm{St} \subseteq \{((i,j), D_{i,j}) \mid 0 \leq i, j \leq 2\}$. Transitions are taken by the human/robber in order to change the room (coordinates) or by the cop/controller to change the status of its doors. These moves are taken in accordance with the shape of the maze. The partitioning of the states between the players follows immediately, as well as the definition of the target states.

4 Searching for Multiple Winning Strategies

To check whether Player 1 has a winning strategy in a 2*TRG* G one can use a classic backward algorithm. We briefly recall it. Let $succ : \mathrm{St} \rightarrow 2^{\mathrm{St}}$ be the function that for each state $s \in \mathrm{St}$ in G gives the set of its successors. The algorithm starts from a set S equal to W. Iteratively, it tries to increase S by adding all states that sutisfy one following conditions holds: (i) $s \in \mathrm{St_o}$ and $succ(s) \subseteq S$; or, (ii) $s \in \mathrm{St_1}$ and $succ(s) \cap S \neq \emptyset$. If S increases at each iteration and at a certain point we enter the initial state, then Player 1 wins the game.

In case one wants to ensure that more than a winning strategy exists, the above algorithm becomes not appropriate. We use instead a top-down automata-theoretic approach. To give an intuition of this solution, first consider that in a 2*TRG* a *witness* for a winning strategy is a tree that takes for each node corresponding to a state s in the game, one successor if s belongs to Player 1, or all successors, otherwise. Indeed, if all the leaves of this tree are target states, then surely Player 1 has a winning strategy over the game. In case we want to ensure that at least two winning strategies exist then at a certain point along the tree Player 1 must take two successors. We build a tree automaton that accepts exactly this kind of witness trees. For the lack of space, we omit the definition of tree and the related concepts. We refer for this to [19].

Definition 2. *A nondeterministic tree automaton (NTA, for short) is a tuple* $A \triangleq < Q,\ \Sigma,\ q_0,\ \delta,\ F >$*, where* Q *is a set of states,* Σ *is an alphabet,* $q_0 \in Q$ *is an initial state,* $\delta : Q \times \Sigma \rightarrow 2^{Q^*}$ *is a transition function mapping pairs of states and symbols to a set of states, and* $F \subseteq Q$ *is a set of the accepting states.*

An *NTA* A recognizes trees and works as follows. For a node tree labeled by σ and A being in a state s, it sends different copies of itself to successors in accordance with δ. For example, if $\delta(s, \sigma) = \{(s_1, s_2), (s_3, s_4)\}$ either A proceeds with (s_1, s_2) or (s_3, s_4), by associating them to two (possibly different) successors. By $L(A)$ we denote the set of trees accepted by A. It is not empty if $L(A) \neq \emptyset$.

We now give the main result of this paper, *i.e.* we show that it is possible to decide in linear time whether, in a 2*TRG*, Player 1 has more than a winning strategy. We later report on the application of this result along the case studies.

Theorem 1. *For a 2TRG game* G *it is possible to decide in linear time whether Player* 1 *has more than a strategy to win the game.*

Proof (sketch). Consider a 2*TRG* game G. We build an *NTA* A that accepts all trees that are witnesses of more than a winning strategy for Player 1 over G. We briefly describe the automaton. It uses $\mathrm{St} \times \{ok, split\}$ as set of states where *ok* and *split* are flags and the latter is used to remember that along the tree Player 1 has to ensure the existence of two winning strategies by opportunely choosing a point where to "split". We use a one-letter alphabet Σ, as this set takes no role. For the initial state we set $q_0 = (s_I, split)$. For the transitions, starting from a state $q = (s, flag)$, we distinguish between two cases: *(i)* $s \in \mathrm{St_o}$. If $flag = ok$ then $\delta(q) = succ(s) \times \{ok\}$, otherwise, let

$succ(s) = \{s_1, \ldots, s_n\}$ then $\delta(q) = \{((s_1, f_1), \ldots, (s_n, f_n))\}$ and there exists $1 \leq i \leq n$ such that $f_i = split$ and for all $j \neq i$, we have $f_j = ok$. *(ii)* $s \in \mathrm{St}_1$. If $flag = ok$ then $\delta(q) = \{((s', ok))\}$ with $s' \in succ(s)$, otherwise, we have $\delta(q) = \{((s', ok), (s'', ok)), ((s', split))\}$, with $s', s'' \in succ(s)$ and $s' \neq s''$. The set of accepting states is $\mathrm{W} \times \{ok\}$. A tree is accepted by A if at a certain point Player 1 can take two successors in G both leading to a target state.

The size of the automaton is just linear in the size of the game. Moreover, by using the fact that, from [19], checking the emptiness of an NTA can be performed in linear time, the desired complexity result follows. □

Consider the Escape Game example. By applying the above construction, the automaton A accepts an empty language. Indeed, for each input tree, A always leads to a leaf containing either a state with a non-target component (i.e., the tree is a witness of a losing strategy) or with a flag *split* (i.e., Player 1 cannot select two winning strategies). Conversely, consider the same game, but flipping the direction of the door d_3 in the maze. In this case, A accepts exactly one tree. Indeed starting from the initial state $(((0,0), \{d_1^c, d_3^c\}), split)$, A sends two copies of itself to two successors in the tree, respectively with states $(((0,0), \{d_1^o, d_3^c\}), ok)$ and $(((0,0), \{d_1^c, d_3^o\}), ok)$, corresponding to two different winning strategies for the controller.

A similar reasoning can be exploited with the Cop and Robber Game example. Indeed, the automaton accepts an empty language. Conversely, by flipping the door d_4, it accepts the tree that is witnessing of two different winning strategies each of them going through one of the two doors left unblocked by the cop.

5 Conclusion and Future Work

In this paper we have introduced a simple but effective automata-based methodology to check whether a player has more than a winning strategy in a two-player game under the reachability objective. We have showed how this methodology can be applied in practice by reporting on its use over two different game scenarios, one cooperative and one adversarial. We believe that the solution algorithm we have conceived in this paper can be used as core engine to count strategies in more involved game scenarios and in many solution concepts reasoning.

This work opens to several interesting questions and extensions. For instance, it would be worth investigating game scenarios in which one or both players have imperfect information regarding some moves of the other player. The imperfect information setting is an important field of study in game theory with several practical applications. For some related works see [6, 12, 14]. Another interesting direction would be to consider the counting of strategies in multi-agent concurrent games. This kind of games have several interesting applications in artificial intelligence [20–22]. One can also consider some kind of hybrid game, where one can opportunely combine team of players working concurrently with some others playing in a turn-based manner [10, 11, 17]. Last but not least, it would be worth investigating infinite-state games. These games arise for example in case the interaction among the players behaves in a recursive way [4, 16].

References

1. R. Alur, T. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.
2. A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic. In *LICS'09*, pages 342–351. IEEE Computer Society, 2009.
3. P. Bonatti, C. Lutz, A. Murano, and M. Vardi. The Complexity of Enriched muCalculi. *LMCS*, 4(3):1–27, 2008.
4. L. Bozzelli, A. Murano, and A. Peron. Pushdown Module Checking. *FMSD*, 36(1):65–95, 2010.
5. D. Calvanese, G. D. Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *IJCAI'99*, volume 99, pages 84–89, 1999.
6. K. Chatterjee, L. Doyen, T. A. Henzinger, and J. Raskin. Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3(4):1–23, 2007.
7. M. Faella, M. Napoli, and M. Parente. Graded Alternating-Time Temporal Logic. *FI*, 105(1-2):189–210, 2010.
8. A. Ferrante and A. Murano. Enriched Mu-Calculi Module Checking. In *FOSSACS'09*, LNCS 5504, pages 183–197. Springer, 2007.
9. A. Ferrante, A. Murano, and M. Parente. Enriched Mu-Calculi Module Checking. *LMCS*, 4(3):1–21, 2008.
10. W. Jamroga and A. Murano. On Module Checking and Strategies. In *AAMAS'14*, pages 701–708. IFAAMAS, 2014.
11. W. Jamroga and A. Murano. Module checking of strategic ability. In *AAMAS'15*, pages 227–235. IFAAMAS, 2015.
12. J.H. Reif. The complexity of two-player games of incomplete information. *Journal of computer and system sciences*, 29(2):274–301, 1984.
13. O. Kupferman, U. Sattler, and M. Vardi. The Complexity of the Graded muCalculus. In *CADE'02*, LNCS 2392, pages 423–437. Springer, 2002.
14. O. Kupferman and M. Vardi. Module Checking Revisited. In *CAV'97*, LNCS 1254, pages 36–47. Springer, 1997.
15. O. Kupferman, M. Vardi, and P. Wolper. Module Checking. *IC*, 164(2):322–344, 2001.
16. A. Murano and G. Perelli. Pushdown multi-agent system verification. In *IJCAI'15*, 2015.
17. A. Murano and L. Sorrentino. A game-based model for human-robots interaction. In *WOA'15*, CEUR Workshop Proceedings. CEUR-WS.org. To appear, 2015.
18. R. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.
19. W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 133–192. Elsevier and MIT Press, 1990.
20. M. Wooldridge. Intelligent Agents. In G. Weiss, editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press: Cambridge, Mass, 1999.
21. M. Wooldridge. *Reasoning about Rational Agents*. MIT Press : Cambridge, Mass, 2000.
22. M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

An Authority Degree-Based Evaluation Strategy for Abstract Argumentation Frameworks

Andrea Pazienza[1], Floriana Esposito[1,2], and Stefano Ferilli[1,2]

[1] Dipartimento di Informatica – Università di Bari
{*name.surname*}@uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni – Università di Bari

Abstract. Abstract argumentation allows to determine in an easy, formal way which claims survive in a conflicting dispute. It works by considering claims as abstract entities, and expressing attack relationships among them. However, this level of expressiveness prevents abstract argumentation systems from being directly applied to reasoning processes where the context is relevant. An outstanding example is when a claim is supported by appealing to authority, so that the audience assigns reliability to the claim's justification based on the authority's renowned experience in the domain. To handle this, we propose to enrich the classical representation used in abstract argumentation by associating arguments with weights that express their degree of authority. The weights' values define their strength in the given domain, which in turn should affect the evaluation of their degree of justification. This paper defines a strategy to combine these weights in order to determine which arguments withstand in a dispute concerning a given domain. Such a strategy was implemented in the ARCA system, that allows to comfortably set up argumentation problems and solve them using both traditional extension-based semantics and the proposed evaluation approach. ARCA is used to illustrate the proposed strategy by means of sample use cases.

1 Introduction

Argumentation is a major component of our everyday lives, in that we are continuously faced with conflicting information and associated inconsistencies. Roughly, an argument is a bunch of information (i.e., a set of *assumptions*) from which conclusions can be drawn, based on a number of reasoning steps. The assumptions used are called the *premises* of the argument, while its *conclusion* (chosen from many possible ones) is called the *claim* of the argument. The support of an argument provides the reason (or, equivalently, a *justification*) for the claim of the argument. This structure simplifies understanding of the opinions of other people and helps in the identification of fallacies in their reasoning. People usually argue in turns, by providing arguments and counterarguments to initial arguments. The winner of the argumentation is the arguer with the last unchallenged argument.

Many strategies can be found in the literature for the identification of the successful arguments in an argumentation dispute context. Some such strategies

are based on the so-called Abstract Argumentation Framework, that will be presented in the next sections. This model of argumentation takes a set of abstract arguments, i.e., arguments whose internal structure or specific interpretation is ignored. The abstract nature of the arguments, and the relationship with non-monotonic reasoning formalisms, yield a very simple and quite general model that allows to easily understand which sets of arguments are mutually compatible. Unfortunately, abstract system representations are not always suitable to depict real situations. This is because abstract systems lack of elements which can empower the representation setting so that conflicts can be automatically identified or the strength of a conflict can be determined. For example, the abstract argumentation framework does not allow to consider the weight of each argument based on the authority of the person who claims it, which may be relevant to the proper evaluation process of judging an argument.

Sometimes it may be appropriate to cite an authority to support a position. This argumentative schema is known as argument from authority, or "*argumentum ad verecundiam*" [17]. Of course, this type of argument can result in a fallacy, especially if the authority is not really such. For instance, an appeal to authority can be inappropriate if the person is not qualified to have an expert opinion on the argument. However, in general an ad verecundiam inductive argument (i.e., an argument whose conclusion is claimed to follow not with certainty but with probability) is not necessarily a fallacy, especially when the relevance of the referred authority is supported by a renowned and proved experience in the argued domain.

This work proposes a novel approach to handle these situations, that extends the abstract argumentation setting by allowing the association of arguments to weights expressing their reliability. Such weights are assigned on the basis of an *authority degree* which takes into account the reliability of the authority who states the argument in the argued domain. The objective is to overcome the low level of expressiveness that characterizes the standard abstract argumentation framework, and to make it able to handle different degrees of reliability on the arguments.

This paper is organized as follows. The next section recalls useful background information, including related works. Then, Section 3 introduces the abstract argumentation framework along with the standard evaluation strategies used in the process of justifying an argument leading to a conclusion. Section 4 describes the proposed approach and how it is embedded in the standard abstract evaluation system, and Section 5 concludes the paper and outlines future work issues.

2 Background and Related Work

As a general, informal definition, argumentation involves the identification of applicable assumptions and conclusions for a given problem under consideration. In this activity, it often faces conflicting information, which results in the need to evaluate the justification for the available conclusions. This, in turn, may involve comparing arguments, evaluating them in some aspects, and judging a

set of arguments and counterarguments to consider whether any part of them can be considered as warranted according to some standard principle. In this context, it can be also safely assumed that each argument has a proponent, who is the person putting forward the argument, and that each argument has an audience, who is the group of people reached by the argument.

Probably the foundational and most important philosophical work for the development of argumentation was made by Toulmin [15]. In particular, he put forward the widely accepted definition for the structure of an abstract argument: an *argument* has a conclusion that is inferred from available data, a *warrant* that allows one to jump to conclusions, and a possible *rebuttal*, which is another argument that disagrees with the original argument. This approach is structural and, in a sense, logical. However, it does not just provide a comprehensive account of the logic of argumentation, and furthermore, it does not address many important questions about how to automate the construction or use of layouts of arguments.

In order to handle arguments systematically, a "formalization" of argumentation is needed. Many professions implicitly or explicitly explore these issues and, in facts, put the systematic use of arguments at the heart of their work. Outstanding examples can be found in the legal, medical, and journalistic professions. The study of formal argumentation started among critical thinking and practical reasoning philosophers [14, 16]. *Critical thinking* is concerned with argument identification and its evaluation by spotting the weak or missing points in arguments. *Practical reasoning* in argumentation is a type of decision making, in which the arguments are used to determine what is the best course of action in practical situations, where the knowledge of the world is incomplete.

However, the need to go beyond the systematic handling of arguments motivates the search for techniques that are able to scale up and deal with substantial and complex problems. Classical logic is appealing as a starting point for argumentation: it provides a rich representation formalism and powerful reasoning mechanism. Unfortunately, inconsistency causes problems in reasoning with classical logic [12]. And, as previously pointed out, argumentation inherently involves conflicting (i.e., inconsistent) information. If the knowledge that is available for constructing arguments is consistent, then no conflicting arguments can be obtained, and thus no recourse to argumentation is needed.

As a partial response to the issue of inconsistency arising in argumentation, three main approaches to formalization for argumentation have been proposed in the literature, namely: abstract systems [8], defeasible systems [13], and coherence systems [9]. The first two approaches use formalisms that are much less expressive (as regards both the complexity of information that can be represented and the complexity of the inferences that can be drawn) than classical logic, thereby circumventing the problem of inconsistency as manifested by the "*ex falso sequitur quodlibet*" rule. The third approach adopts a simple strategy to improve the problem of inconsistency.

In particular, abstract systems build on the seminal proposal by Dung [8]. It is based on the assumption that the structure of a set of arguments and counter-

arguments can be expressed by defining a set of arguments and a binary 'attack' relationship between pairs of arguments. The attack relationship captures the situation of one argument undermining the credibility of another. This setting can be represented as a graph, with each node representing an argument and each edge representing an 'attack'. Under this representation, the set of nodes in the graph is the starting point. Given such a graph, the objective is determining which subset(s) (called *extension(s)*) of its nodes (i.e., arguments) can be accepted. Providing different strategies to answer this question corresponds to defining different argumentation semantics. In other words, the idea of a semantics is, given an argumentation framework, to specify zero or more sets of acceptable arguments. Dung also provided a number of semantics, which specify different evaluation strategies ranging from the credulous to the skeptical (see next section for more details). Also Caminada proposed new extension based semantics approaches, which produce reasonable results in situations where Dung's extensions have drawbacks or don't exist [5, 4].

The argumentation literature emphasized the importance of considering additional criteria, namely *preferences*, when evaluating arguments in a framework. Preferences are expressed between arguments and reflect their relative strengths. In [1] a Preference-based Argumentation Framework (PAF) is built to handle correctly critical attacks in the framework and to refine the evaluation of arguments.

A first introduction to weighted attack relations in an argumentation framework can be found in [11], where a natural extension of Dung's model of argument systems is investigated in order to propose attacks associated with a weight indicating the relative strength of the attack. Such a model takes the name of Weighted Argumentation Framework (WAF). This model was further explored in [10] to check how much inconsistency should be tolerated in a WAF. This approach permits a much more fine-grained level of analysis of argument systems than the unweighted case, and can provide useful solutions when conventional argument systems cannot provide any. Furthermore, in [7] weights are used for relaxing extensions in order to improve the inferential power of the argumentation framework, while in [3] the authors suggest semirings as a mean to parametrically represent WAFs.

Another early extension of Dung's proposal with weights is Value-based Argumentation Frameworks (VAFs) [2]. In the VAF approach, the strength of an argument depends on the social values that it advances, and the decision about whether the attack of one argument on another succeeds depends on the comparative strength of the values advanced by the involved arguments.

A more general approach to extending Dung's proposal is that of Bipolar Argumentation Frameworks (BAFs), which take into account two kinds of interaction between arguments: a positive interaction (by which an argument can help or support another argument) and a negative interaction (by which an argument can attack another argument) [6].

3 Abstract Argument Systems

An *abstract argument system* or *Argumentation Framework* (AF for short), as introduced by Dung, is a pair $\langle A, R\rangle$ consisting of a set A, whose elements are called *arguments*, and a binary relation $R \subseteq A \times A$ on A, called *attack relation*. Given two arguments $\alpha, \beta \in A$, the relation $\alpha R \beta$ represents an attack from α against β. In general, arguments α and β are in conflict if argument α refutes argument β or if α is attacking premises supporting β. More precisely, we talk about:

- *Rebutting*, when there is an explicit contradiction between conclusions ; or
- *Undercutting*, when argument α attacks the applicability of a rule that supports β, without necessarily denying it.

An AF has a typical representation as a directed graph where nodes are arguments and edges are drawn from attacking to attacked arguments. Representing the structure and meaning of arguments at so high a level of abstraction allows to better focus on properties that are independent from any specific context, and makes it applicable to a wide variety of domains. On the other hand, this formalism lacks of expressiveness, which prevents its direct application in any specific domain. Indeed, in order to set up an AF one first needs to build an underlying knowledge base, along with mechanisms to generate the set of arguments from it and determine in which ways these arguments attack each other. Then, once the AF has been set up, a second issue is how to determine a *justification state* for the involved arguments and, in particular, how to identify which are the justified ones. Informally, an argument is considered to be justified if it survives to attack relations. Therefore, the next step is to understand which argument is not defeated from the confrontation with the others. This process, called *argument evaluation*, aims at determining the justification state of the arguments in an abstract argumentation system.

An *argumentation semantics* is the formal way of determining which arguments or statements can be considered as justified in the argument evaluation process. Two main approaches to the definition of argumentation semantics are available in the literature: the *labelling-based* one and the *extension-based* one. In the former, the idea is to define a mapping that associates each argument to one of a set of labels corresponding to the possible states of argumentsin the given context. A sensible choice for the set of labels is:

- *in* for the accepted arguments
- *out* for the rejected arguments
- *undec* for undefined (not accepted or refused) arguments.

The labeling operation can be seen as the result of the reasoning carried out by an agent which analyzes the arguments and marks them as justified, rejected or temporarily undecided. One of the benefits derived from the use of labelling-based semantics is the possibility of defining a more refined defeat-status by introducing different levels of justification and rejection (e.g., 'very acceptable', 'quite acceptable', 'not acceptable').

In the extension-based approach the idea is to derive, from an AF, an 'extension' E, that is a subset of A representing a set of argument which are considered as acceptable. These semantics can assign each node to a single status (*unique-status*) or multiple statuses (*multiple-status*). The difference is in the management of the temporarily undecided state. A multiple-status semantics can resolve a mutual attack issue by generating two hypothetical solutions in which the conflicting arguments can be alternately assumed as acceptable.

By considering the expressiveness of the two approaches to semantics, it can be observed that any extension-based semantics can be equivalently translated in a labelling-based one by adopting a set of two labels *in* and *out* that correspond to extension membership. Vice versa, in general an arbitrary assignment of labels cannot be translated in terms of extensions. This is because labellings always include a label that corresponds to the extension membership, while other labels are derivable from extension membership and the attack relation. Consequently, equivalent extension-based definitions of labelling-based semantics are in general applicable. This is the reason why extension-based semantics are more widely exploited in the literature.

A basic requirement for any extension E is derived from its interpretation as a set of arguments which can survive together. In other words, if an argument α attacks another argument β, one reasonably does not expect to have them together in the same extension. This corresponds to the concept of *conflict-free* that is at the basis of all extension-based semantics.

Definition 3.1 (conflict-free) *Given an Argumentation Framework $AF = \langle A, R\rangle$, a set $S \subseteq A$ is* conflict-free *iff $\nexists \alpha, \beta \in S$ s.t. $\alpha R \beta$ (α attacks β).*

A second requirement corresponds to the need of a set of arguments to resist the attacks it receives from other arguments by counterattacking them. This feature is based on the notions of acceptable argument and admissible set.

Definition 3.2 (acceptability) *Given an Argumentation Framework $AF = \langle A, R\rangle$, an argument $\alpha \in A$ is* acceptable *wrt a set $S \subseteq A$ iff $\forall \beta \in A : \beta R \alpha \Rightarrow \exists \gamma \in S$ s.t. $\gamma R \beta$ (α is defended by S).*

Definition 3.3 (admissibility) *Given an Argumentation Framework $AF = \langle A, R\rangle$, a set $S \subseteq A$ is* admissible *iff S is conflict-free and $\forall \alpha \in S$ α is acceptable wrt S.*

Now suppose that the attackers of an argument α are all attacked by an extension E. Then the attack suffered by α is canceled because E is 'defending' α, and α is reinstated because it should belong to E. This property takes the name of *reinstatement* and leads to the following principle:

Definition 3.4 (reinstatement principle) *Given an Argumentation Framework $AF = \langle A, R\rangle$, a semantics satisfies the* reinstatement principle *iff for all extensions $E \subseteq A$ it holds that*

if α is acceptable w.r.t E then $\alpha \in E$.

3.1 Extension-Based Semantics

Since semantics provide the basis for evaluating the justification state of arguments, one may first require that the evaluation basis of an AF is not empty. Some (labelling- or extension-based) semantics may allow many alternative justification states for the arguments. Two main alternatives may be considered for the notion of justification state:

- *skeptical justification* requires that an argument is accepted in all semantics;
- *credulous justification* requires that an argument is accepted in at least one semantics.

Of course in a unique-status approach credulous and skeptical justifications coincide, but in multiple-status approaches typically the credulous justification includes the skeptical justification.

Let us now consider some approaches to determine argumentation semantics proposed in the literature.

Complete semantics The notion of complete extension is based on the principles of admissibility and reinstatement. It is a set which is able to defend itself and includes all arguments it defends.

Definition 3.5 (complete extension) *Given an Argumentation Framework* $AF = \langle A, R \rangle$*, a set* $S \subseteq A$ *is* complete extension *iff* S *is admissible and* $\nexists$ $\alpha \in A$ *such that:*

- α *is acceptable wrt* S
- $\alpha \notin S$

The following semantics build their own extensions referring to the complete extensions.

Ground semantics For each AF there exists only one *ground extension* which corresponds to the set of arguments that satisfies the conditions of admissibility and that is minimal with respect to the inclusion relation between the admissible sets of AF. Compared to complete extensions, the ground is the complete minimal one with respect to set inclusion.

Definition 3.6 (ground extension) *Given an Argumentation Framework* $AF = \langle A, R \rangle$*, a set* $S \subseteq A$ *is a* ground extension *iff* S *is admissible and* S *is a* $\subseteq$*-minimal subset of* A*.*

Preferred semantics A *preferred extension* S of an AF is the admissible set of AF which is maximal with respect to set inclusion. For each admissible set E of AF there exists at least one preferred extension S such that $E \subseteq S$ (it can be also the empty set). Compared to complete extensions, the preferred extension is the complete maximal one with respect to set inclusion.

Definition 3.7 (preferred extension) *Given an Argumentation Framework $AF = \langle A, R \rangle$, a set $S \subseteq A$ is a* preferred extension *iff S is admissible and S is a $\subseteq$-maximal subset of A.*

Stable semantics A *stable extension* of an AF is a complete extension which attacks all arguments that are not its members. Any stable extension is also a maximal conflict-free set of AF.

Definition 3.8 (stable extension) *Given an Argumentation Framework $AF = \langle A, R \rangle$, a set $S \subseteq A$ is a* stable extension *iff*

- *S is a complete extension*
- *$S \cup S_{defeated} = A$*

where $S_{defeated} = \{\beta \in A \mid \alpha \in E \wedge \alpha R \beta\}$

Semi-stable semantics A *semi-stable extension* [5] S of an AF is a complete extension which relies on the idea of maximizing not only the arguments belonging the extension but also those attacked by it. Any semi-stable extension S is also a set with maximal range with respect to the inclusion set.

Definition 3.9 (semi-stable extension) *Given an Argumentation Framework $AF = \langle A, R \rangle$, a set $S \subseteq A$ is a* stable extension *iff*

- *S is a complete extension*
- *$S \cup S_{defeated}$ is maximal wrt A*

where $S_{defeated} = \{\beta \in A \mid \alpha \in E \wedge \alpha R \beta\}$

Ideal semantics An extension of an AF is called *ideal* if it corresponds to the largest admissible set that is a subset of each preferred extension.

Definition 3.10 (ideal extension) *Given an Argumentation Framework $AF = \langle A, R \rangle$, a set $S \subseteq A$ is an* ideal extension *iff S is the admissible $\subseteq$-maximal subset of A such that $\forall S_{preferred} : S \subseteq S_{preferred}$.*

Eager semantics An *eager* extension of an AF corresponds to the largest admissible set that is a subset of each semi-stable extension. It relies on a concept that is similar to the ideal semantics, with the restriction that the admissible set must be in the intersection of semi-stable extensions.

Definition 3.11 (eager extension) *Given an Argumentation Framework $AF = \langle A, R \rangle$, a set $S \subseteq A$ is an* eager extension *iff S is the admissible $\subseteq$-maximal subset of A such that $\forall S_{semi-stable} : S \subseteq S_{semi-stable}$.*

An ordering relationship exists among the semantics described above [5]. The ground, preferred, ideal, eager, semi-stable and stable extensions can be all obtained starting from complete extensions. In particular, each stable extension is also semi-stable and each semi-stable extension is also a preferred one. Finally, by definition, a preferred extension is a complete extension too.

4 Authority Degree

The semantics shown above neglect information that, in some cases, may turn out to be of crucial importance to the argumentation. For instance, the Abstract Argumentation Framework does not distinguish between rebuttal and undercutting attacks, in order to provide more efficient computation of extensions. Also, in some cases, it would be advisable to evaluate the set of reliable arguments by taking into account the context in which the sentences are claimed, and specifically the trustworthiness of those who claim them. Adding quantitative information becomes of crucial importance when arguments have different levels of strength. Hence, adding a weight to arguments allows to give them the right strength, so as to represent real dialogues.

A first refinement to deal with this scenario could be to distinguish utterances made by domain experts from those made by novices or by outsiders of the domain of the argumentation. By domain we mean a context in which a person is skilled. The more confident a person within a domain, the higher his authority in that domain. For example, in a wine and food context, the opinion or contradiction of a mathematician has a minor significance compared to that of a winemaker of unquestionable professionalism. Conversely, in a mathematical context the winemaker level of reliability should be less than that of the mathematician. This degree of reliability might be captured in an Argumentation Framework by introducing an *authority degree* associated to nodes, such that two nodes reporting utterances made by experts in different domains will have different weights into attacking the same node. Thus, arguments are partitioned in domains which reflect the area of expertise of each arguer.

A second refinement might be to consider the number of attackers and defenders for a node in the graph. In abstract argumentation terms, the larger the number of attackers of a node, the more likely it is that it should be defeated, and, conversely, the larger number of defenders (i.e., of reinstatements) it has, the more likely it is that it may be admissible.

In general, our proposal is to associate a '*social*' weight to attack relations, preferring large admissible sets which attack external nodes and at the same time defend their members, rather smaller sets of isolated admissible nodes. In this setting, the domain-based authority degree works as '*normalizer*': it rebalances weights with the aim of avoiding over-defended conclusions. Another novelty of our approach is that the authority degree may differ depending on the intended domain. This is different than in PAFs [1], where preferences are taken into account at the semantics level. That is, instead of modifying the inputs of Dung's framework, PAFs extend semantics with preferences.

4.1 Authority Function

The setting we propose is defined through a number of functions to be used in the evaluation strategy of justification. In the following, $\langle A, R\rangle$ will indicate an argumentation framework and ϵ_α the authority degree for an argument $\alpha \in A$

in its domain Δ_α. A *domain function* allows us to focus on the most represented domains in the framework rather than on niche domains:

Definition 4.1 (domain function) $\delta\colon A \to \mathbb{R}$ *s.t.* $\forall \alpha \in A$ *:*

$$\delta(\alpha) = \frac{N_\alpha}{|A|}$$

with N_α *number of arguments having domain* Δ_α*.*

The domain function δ acts as a moderator to balance the weights of the attacking nodes within the argued context. In fact, this ensures that the arguments in support of the discussed domain have more relevance.

In the following, we propose three functions which allow to consider the strength of a group of persons involved in the same domain.

An *attacking function* returns the number of attacks launched by an argument towards other nodes.

Definition 4.2 (attacking function) $f_a\colon A \to \mathbb{N}$ *s.t.* $\forall \alpha \in A$:

$$f_a(\alpha) = |U(\alpha)|, \text{where } U(\alpha) = \{\beta \in A \mid \alpha R \beta \wedge \epsilon_\alpha \cdot \delta(\alpha) \geq \epsilon_\beta\}$$

A *defeating function* returns the number of attacks an argument suffers from other nodes.

Definition 4.3 (defeating function) $f_d : A \to \mathbb{N}$ *s.t.* $\forall \alpha \in A$:

$$f_d(\alpha) = |E(\alpha)|, \text{where } E(\alpha) = \{\beta \in A \mid \beta R \alpha \wedge \epsilon_\alpha < \epsilon_\beta \cdot \delta(\beta)\}$$

A *defending function* returns the number of attacks suffered by an argument that are not defended by other sufficiently reliable arguments.

Definition 4.4 (defending function) $f_r\colon A \to \mathbb{N}$ *s.t.* $\forall \alpha \in A$:

$$f_r(\alpha) = |D(\alpha)|, \text{where } D(\alpha) = \{\beta \in A \mid \beta R \alpha \wedge \exists \gamma \in A \text{ s.t. } \gamma R \beta \wedge \epsilon_\gamma \cdot \delta(\gamma) \geq \epsilon_\beta\}$$

In the last three functions described above, the domain function δ serves to support the attacking node in order to contextualize the weight of the attack compared to the weight of the attacked node.

Now, let call us each argument $\alpha \in A$ as *authority-node* wrt an argument $\beta \in A$, that argument which launches attacks towards other nodes $\beta \in A$ such that $\epsilon_\alpha * \delta(\alpha) \geq \epsilon_\beta$. An *authority function* measures the degree of an argument $\alpha \in A$ based on the number of attacks launched as authority-node wrt $\beta \in A$ and the number of attacks suffered by other arguments being authority-nodes against it.

Definition 4.5 (authority function) *Let* $f_{authority}\colon \mathbb{N} \to \mathbb{N}$ *be a function s.t.*

$$f_{authority}(\alpha) = f_a(\alpha) - f_d(\alpha) + f_r(\alpha)$$

The more attacks launched by $\alpha \in A$ as an authority-node wrt $\beta \in A$, the higher its $f_{authority}$; the more attacks it suffers by arguments which are authority-nodes against it, the lower its $f_{authority}$. Intuitively, if the attacks suffered by a node α are defended by other authority-nodes, then its authority degree will depend only on its successful attacks (i.e. attacks towards less reliable nodes). Indeed, the suffered attacks, which decrease the value of its authority, are balanced by the number of attacks from which it is defended by authority-nodes. Hence, the authority function is used to select the 'stronger' admissible set: namely the most reliable set will be the one with the lower number of nodes such that their authority function value is maximum. In this setting, none of the classical Dung's extensions are considered, but only the collection of admissible sets, ordered according to the value of their $f_{authority}$. Then, the smallest admissible set with highest value of the authority function is chosen as more reliable justified set. Applying a more skeptical semantics may limit the aim of this paper because we would lose the sense of the domain's weight associated to nodes. In general, an attack is considered valid if the attacker's authority (decreased by a domain-dependent factor) is higher than the attacked authority.

Fig. 1 shows an example of a graph depicting an AF. Node labels indicate the level of authority. Nodes in gray belong to domain Δ' and those in white to domain Δ''. Due to space constraints, we will determine in the following the authority degree for node α only, using the above functions.

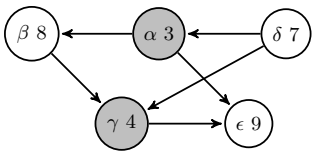


Fig. 1. AF graph example with authority degrees

Let consider the node α, it's domain value for domain Δ' is $\delta(\alpha) = \frac{2}{5} = 0.4$;
$f_a(\alpha) = 0$ because $3 \cdot 0.4 = 1.2$ s.t. $1.2 \not\geq 8$ and $1.2 \not\geq 9$;
$f_d(\alpha) = 1$ because $3 < 8 \cdot 0.6 = 4.8$;
$f_r(\alpha) = 0$ because node δ suffers no attack;
$f_{authority}(\alpha) = 0 - 1 + 0 = -1$.

The authority functions for the remaining nodes are calculated in the same way. Then, the smallest admissible set, having the maximum sum of authority functions values for its members, is chosen as the most reliable justified subset of arguments.

4.2 ARCA

The proposed strategy was implemented in the ARCA system (acronym for *Abstract Resolution of Conflicts in Argumentation*). ARCA includes a logic

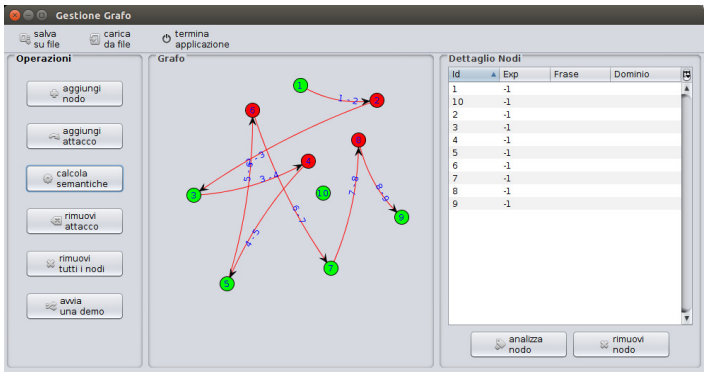


Fig. 2. ARCA

program core which can compute both the classical extension based semantics and the proposed authority degree-based evaluation of arguments. On top of it, ARCA provides a graphic tool (see Fig. 2) to enter and display arguments and attacks in an AF, and to associate to each claim a domain of origin and a degree of experience in that domain. It also allows to associate and display utterances associated to arguments and to show nodes with different colors denoting their acceptability according to the different semantics. This allows one to easily see conflicts and observe the differences between different semantic extensions.

4.3 Sample Scenario

Let us now show the various features of ARCA using a sample scenario in which some professionals olive growers are arguing about the most useful criterion of olive trees pruning. In mature trees, pruning is mainly required to renew the fruiting surface of the tree and achieve high yields, maintain vegetative growth of sprouts, maintain the skeleton structure, contain tree size, favor light penetration and air circulation inside the canopy, permit control of pests and diseases, prevent aging of the canopy, and eliminate dead wood.

Three novice croppers, Albert, John and Jack, are expressing their point of view, according their own (limited) experience:

1. Albert: "*When in doubt, less pruning is better.*"
2. John: "*Not all trees in a grove need to be pruned every year.*"
3. Jack: "*Pruning should be rapid and simple.*"

Samuel, a renowned expert olive grower, counterargues all three statements according to his large experience: "*The type of pruning must be adjusted in relation to plant age, training system, crop load, product use, environmental conditions, soil fertility, and farm structure.*" Thus, Samuel's opinion has more

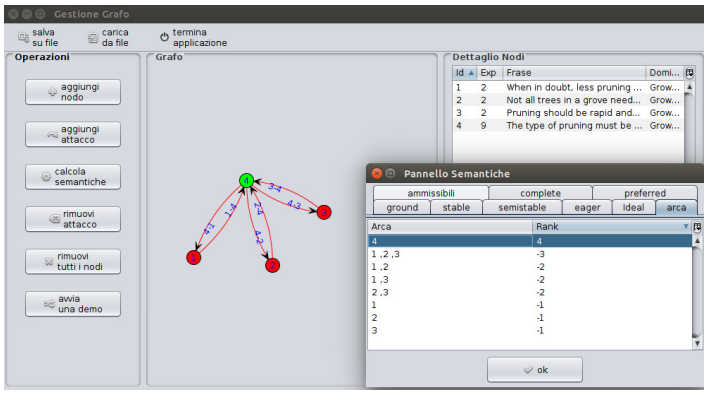


Fig. 3. Growers Argumentation

relevance than those of the three novice growers. Hence, his authority degree is such that the attacks he is suffering have no effect in the argumentation. In fact, Fig. 3 shows that, among all admissible subsets, the one with higher rank is precisely the set containing Samuel's (winning) argument.

Now, suppose Julian, an agronomist, takes part in the discussion. As a person with special knowledge in soil management and field-crop production, he explains in which direction new techniques in this domain are going and, therefore, which strategy is better to accomplish all aims: "*Current tendency is to prune olive trees as little as possible, so as to reduce costs substantially and simplify pruning management.*" This sentence attacks Samuel's claim and generates circular conflicts between Julian's claim and the novice growers' statements. Since Julian has less practical experience in pruning olive trees, his claim has less weight in the argumentation. In this situation, the evaluation of extension-based semantics and the ARCA solution are quite different. In skeptical extension-based semantics such as Semi-Stable and Stable extensions (Fig. 4 (a)), the admissible undefeated set of arguments includes both the claims of novice growers, and the agronomist's one. In the ARCA solution (see Fig. 4 (b)), the weight of the agronomist in the argumentation is such that his argument is undefeated, despite his weight is lower than Samuel's one in the argued domain.

Suppose now that Samuel counterattacks also Julian's claim with another argument: "*Pruning should be more severe on old trees and trees of low vigor than on young plants, or on trees growing in irrigated conditions and in fertile soils.* The authority weight of Samuel's last argument determines the winning arguments in the ARCA solution. Indeed, in Fig. 5 the two arguments expressed by Samuel are a subset of admissible elements which has a higher rank in the ARCA solution.

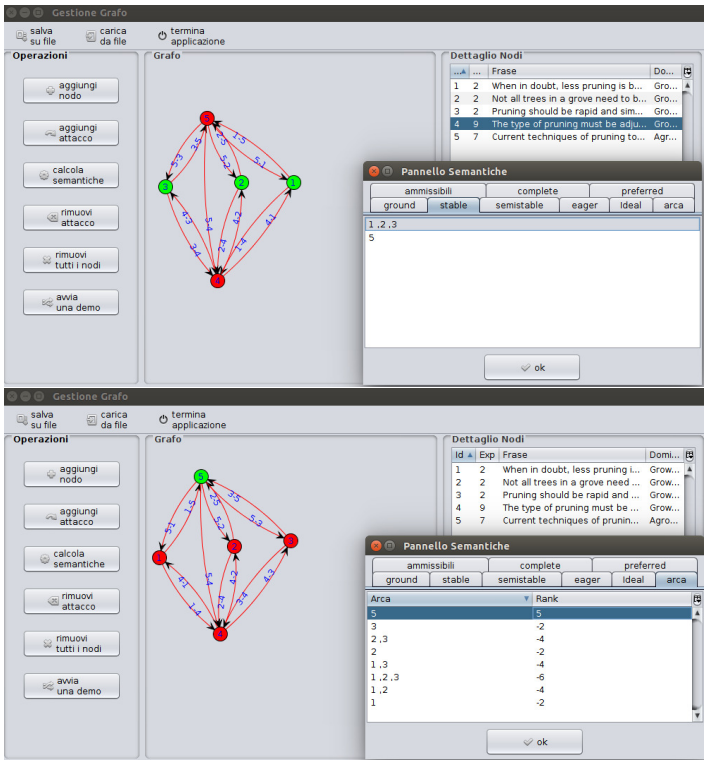


Fig. 4. Stable Extension solution (a) versus ARCA solution (b)

This sample scenario clearly shows how the weighted evaluation strategy of justified arguments may lead to more reliable and sensible results when the context domain is relevant.

5 Conclusions and Future Work

Abstract Argumentation is a formal approach to define which claims withstand in a dispute, in which the only expressed property is a binary 'attack' relation representing the rebuttal of an argument to another. The aim is determining an evaluation strategy that allows to justify conflicting arguments. While several such strategies have been defined, not always they are useful or sensible. This is due to the low level of expressiveness of abstract systems which doesn't allow to represent all relevant contextual situations. E.g., using an appeal to authority,

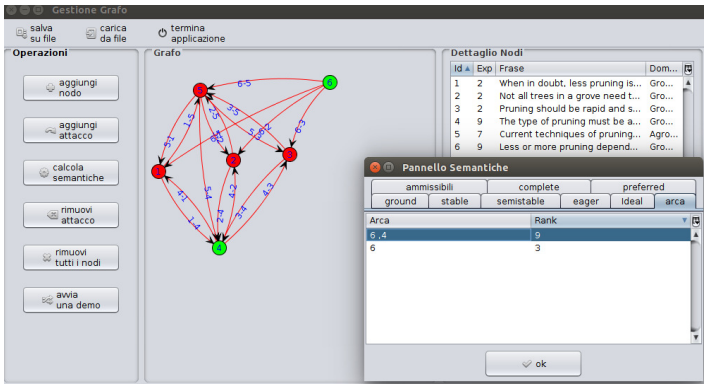


Fig. 5. Last Samuel's claim determines the winning arguments

claims may have different weights in the discourse, and justifications might be evaluated differently. In turn, the weight of a claim depends on the degree of experience (i.e., the authority) of the person expressing it in a particular domain. This work proposed a novel evaluation strategy which may take into account the authority degree of arguments in a given domain in order to understand which arguments survive in a debate. It was implemented in the ARCA system, that allows easily to set up abstract argumentation frameworks and solve justified arguments with both classical extension-based semantics and the proposed authority-weighted approach. A sample scenario is used to illustrate how ARCA works and how the proposed strategy is useful to best handle a real-world argumentation problem in which the reliability of claims is significant.

As future work, we will investigate the adoption of modal logics depending on the actor's domain membership. E.g., deontic logic in legal field or temporal logic in historical-literary field. We would also like to implement (semi-)automatic analysis of arguments expressed in natural language, so as to help the users of ARCA in setting up their argumentation frameworks. Specifically concerning the use of weights, we would like to extend the abstract model with the use of Value-based Argumentation Frameworks in which the justification of an argument depends on the social values that it advances, and the process of justification of one argument depends on the strength of the values involved in the argumentation. Also the extension to Bipolar Argumentation Frameworks will be considered, so as to distinguish the sense of 'support' to an argument from the sense of 'defense' of an argument.

Acknowledgments

This work was partially funded by the Italian PON 2007-2013 project PON02_00563_3489339 'Puglia@Service'.

References

[1] Leila Amgoud and Srdjan Vesic. Rich preference-based argumentation frameworks. *International Journal of Approximate Reasoning*, 55(2):585 – 606, 2014.

[2] T. J. M. Bench-Capon, S. Doutre, and P. E. Dunne. Value-based argumentation frameworks. In *Artificial Intelligence*, pages 444–453, 2002.

[3] Stefano Bistarelli and Francesco Santini. A common computational framework for semiring-based argumentation systems. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 131–136, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[4] M. Caminada. Comparing two unique extension semantics for formal argumentation: Ideal and eager. In *BNAIC 2007*, pages 81–87, 2007.

[5] M. Caminada, W. A. Carnielli, and P. E. Dunne. Semi-stable semantics. *J. Log. Comput.*, 22(5):1207–1254, 2012.

[6] C. Cayrol and M.C. Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In Lluís Godo, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer Berlin Heidelberg, 2005.

[7] Sylvie Coste-Marquis, Sébastien Konieczny, Pierre Marquis, and Mohand Akli Ouali. Selecting extensions in weighted argumentation frameworks. In *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012*, pages 342–349, 2012.

[8] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[9] P. E. Dunne and T.J.M. Bench-Capon. Coherence in finite argument systems. *Artificial Intelligence*, 141(1–2):187 – 203, 2002.

[10] P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Inconsistency tolerance in weighted argument systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 851–858, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[11] P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, 175(2):457 – 486, 2011.

[12] A. C. Kakas, F. Toni, and P. Mancarella. Argumentation for propositional logic and nonmonotonic reasoning. In *Proceedings of the 29th Italian Conference on Computational Logic, Torino, Italy, June 16-18, 2014.*, pages 272–286, 2014.

[13] H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. In *Handbook of philosophical logic*, pages 219–318. Springer Netherlands, 2002.

[14] H. S. Richardson. *Practical Reasoning about Final Ends.* Cambridge Studies in Philosophy. Cambridge University Press, 1997.

[15] S. Toulmin. *The Uses of Argument.* E-Books von NetLibrary. Cambridge University Press, 2003.

[16] D. Walton. *Fundamentals of Critical Argumentation.* Critical Reasoning and Argumentation. Cambridge University Press, 2005.

[17] D. Walton. *Appeal to Expert Opinion: Arguments from Authority.* Pennsylvania State University Press, 2010.

Towards Visualising Security with Arguments

Stefano Bistarelli[1,2], Fabio Rossi[1], Francesco Santini[2,⋆], and Carlo Taticchi[1]

[1] Dipartimento di Matematica e Informatica, Università di Perugia, Italy
[bista,rossi,taticchi]@dmi.unipg.it
[2] IIT-CNR, Pisa, Italy
francesco.santini@iit.cnr.it

Abstract. Abstract Argumentation has been proved as a simple yet powerful approach to manage conflicts in reasoning with the purpose to find subsets of "surviving" arguments. Our intent is to exploit such form of resolution to visually support the administration of security in complex systems. For instance, in case threat countermeasures are in conflict (also with assets) and only some of them can be selected.

1 Introduction and Motivations

An *Abstract Argumentation Framework* (AAF), or System, as introduced in a seminal paper by Dung [6], is simply a pair $\langle A, R\rangle$ consisting of a set A whose elements are called arguments and of a binary relation R on A, called "attack" relation. An abstract argument is not assumed to have any specific structure but, roughly speaking, an argument is anything that may attack or be attacked by another argument. The sets of arguments (or *extensions*) to be considered are then defined under different semantics, which are related to varying degrees of scepticism or credulousness.

In this work, our goal is to start developing a tool to visualise security threats and related countermeasures as arguments, as if security was a continuous dynamic discussion between the administrator and the surveilled system. Existing automated tools to defend a system from such security threats are one potential solution, but a completely automated approach could undervalue the strong analytic capabilities of humans, particularly in problematic situations that require vigilant human oversight.

We measure the strength of subsets of arguments and single arguments in accordance with Argumentation Theory. We print such strength degrees in different colours with the purpose to immediately catch the attention of the Security Administrator on what is going on in his system, and help him to take a decision on the set of countermeasures to be considered.

2 Preliminaries

In this section we briefly summarise the background information related to classical Abstract Argumentation Frameworks (AAFs) [6].

⋆ The author is supported by MIUR PRIN 2010XSEMLC "Security Horizons".

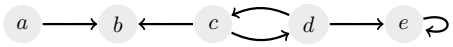


Fig. 1: An example of AAF.

Definition 1 (AAF). *An Abstract Argumentation Framework (AAF) is a pair* $F = \langle A, R \rangle$ *of a set* A *of arguments and a binary relation* $R \subseteq A \times A$*, called the attack relation.* $\forall a, b \in A$*,* aRb *(or,* $a \rightarrowtail b$*) means that* a *attacks* b*. An AAF may be represented by a directed graph whose nodes are arguments and edges represent the attack relation. A set of arguments* $S \subseteq A$ *attacks an argument* a*, i.e.,* $S \rightarrowtail a$*, if* a *is attacked by an argument of* S*, i.e.,* $\exists b \in S.b \rightarrowtail a$*. An argument* $a \in A$ *is defended (in* F*) by a set* $S \subseteq A$ *if for each* $b \in A$*, such that* $b \rightarrowtail a$*, also* $S \rightarrowtail b$ *holds.*

Argumentation semantics [6] characterise a collective "acceptability" for arguments. Respectively, *adm*, *com*, *prf*, and *stb* stand for admissible, complete, preferred, and stable semantics.

Definition 2 (Semantics [6]). *Let* $F = \langle A, R \rangle$ *be an AAF. A set* $S \subseteq A$ *is conflict-free (in* F*), denoted* $S \in cf(F)$*, iff there are no* $a, b \in S$*, such that* $a \rightarrowtail b$ *or* $b \rightarrowtail a \in R$*. For* $S \in cf(F)$*, it holds that*

- $S \in adm(F)$*, if each* $a \in S$ *is defended by* S*;*
- $S \in com(F)$*, if* $S \in adm(F)$ *and for each* $a \in A$ *defended by* S*,* $a \in S$ *holds;*
- $S \in prf(F)$*, if* $S \in adm(F)$ *and there is no* $T \in adm(F)$ *with* $S \subset T$*;*
- $S \in stb(F)$*, if for each* $a \in A \backslash S$*,* $S \rightarrowtail a$*;*

We also recall that the requirements in Def. 2 define an inclusion hierarchy on the corresponding extensions, from the most to the least stringent: $stb(F) \subseteq prf(F) \subseteq com(F) \subseteq adm(F)$. Moreover, $\sigma(F) \neq \emptyset$ always holds for each considered semantics σ (except for the stable one).

Definition 3 (Arguments acceptance-state). *Given one of the semantics* σ *in Def. 2 and a framework* F*, an argument* a *is* i) *sceptically accepted if* $\forall S \in \sigma(F), a \in S$*,* ii) *a is credulously accepted if* $\exists S \in \sigma(F), a \in S$ *and* a *is not sceptically accepted, and* iii) *a is rejected if* $\nexists S \in \sigma(F), a \in S$*.*

Consider $F = \langle A, R \rangle$ in Fig. 1, with $A = \{a, b, c, d, e\}$ and $R = \{a \rightarrowtail b, c \rightarrowtail b, c \rightarrowtail d, d \rightarrowtail c, d \rightarrowtail e, e \rightarrowtail e\}$. In F we have $adm(F) = \{\emptyset, \{a\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}\}$, $com(F) = \{\{a\}, \{a, c\}, \{a, d\}\}$, $prf(F) = \{\{a, d\}, \{a, c\}\}$, and $stb(F) = \{\{a, d\}\}$. Hence, argument a is sceptically accepted in $com(F)$, $prf(F)$ and $stb(F)$, while it is only credulously accepted in $adm(F)$.

3 A Visualisation Example

Consider a small research and development company. This company cooperates with other (often large) enterprises for the development of complex goods. Such

company possesses high-tech knowledge which has to be protected from competitors. The company needs to efficiently use its resources with the purpose to survive in a highly competitive market. In short, the company has the goal (i.e., asset) of ensuring the productivity of operations (QoS).

In this small example, the security-system administrator has identified the following threats and related security controls (in square brackets): hacker penetration (HP) [host IDS (HI), network IDS (NI)] (where IDS stands for Intrusion Detection System), employee abuse (EA) [monitoring functionality (MF), audit procedures (AP)], and compromise of communication channel (CCC) [virtual private network (VPN), encrypted line (EL)].

We would like to emphasise that abstract arguments have no internal structure, and are not "directly linked" to classical logic. For this reason, we can consider multiple sources of information but and belief, such as case law, common sense, and expert opinion. We can consider information coming from multiple network-sensors, in the form of logs, warnings, and errors. Facts and beliefs can be also taken from internal policy documents, and standard documents as well. For instance the *Standard of Good Practice for Information Security*, is a business-focused, practical and comprehensive guide to identifying and managing information security risks in organizations and their supply chains. The 2011 Standard is aligned with the requirements for an Information Security Management System (ISMS) set out in ISO/IEC 27000-series standards, and provides wider and deeper coverage of ISOIEC 27002[3] control topics, as well as cloud computing, information leakage, consumer devices and security governance.

To work on our example we use *SecArg*[4] (Security with Arguments). SecArg is based on ConArg [4, 5] (ARGumentation with CONstraints), which is an Abstract Argumentation reasoning-tool using the *Gecode* library[5], an efficient C++ environment where to develop constraint-based applications. The input (text) file passed to SecArg contains the list of arguments partitioned into *countermeasures*, *threats*, *assets*, and attacks between them: for instance, *countermeasure(HI)*, *threat(HP)*, *att(HI,HP)* (hacker penetration is prevented by a host IDS). SecArg visually represents the different nature of arguments with different colours: green for countermeasures, red for threats, and yellow for assets.

A more extended example is represented in Fig. 2. In such AAF we have that executing a host IDS and a monitoring functionality on the same machine (i.e., HI&MF) impacts on its QoS. Hence, we pose an attack between them, and we also consider not having HI (NotHI) or MF (NotMF). Moreover, we have some countermeasures in conflict, i.e., EL or VPN, and MF.

We obtain three stable extensions (we use the stable semantics because it is the most sceptical one, see Sec. 2): *i)* {AP, VPN, EL, HI, NI, NotMF, QoS}, *ii)* {AP, VPN, EL, HI, NI, HI&MF}, and *iii)* {AP, VPN, EL, NI, NotHI, NotMF, QoS}. In this case, reasoning in terms of stable or preferred semantics is the same, since they both returns the same three extensions. Reasoning on the scep-

[3] ISO, ISO, and I. E. C. Std. "ISO 27002: 2005." Information Technology-Security Techniques-Code of Practice for Information Security Management. ISO (2005).

[4] `http://www.dmi.unipg.it/secarg`

[5] `http://www.gecode.org`

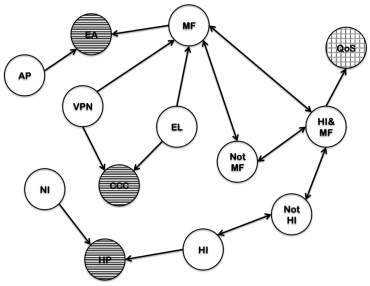


Fig. 2: The AAF with controls, threats (horizontal filling), and *QoS* asset.

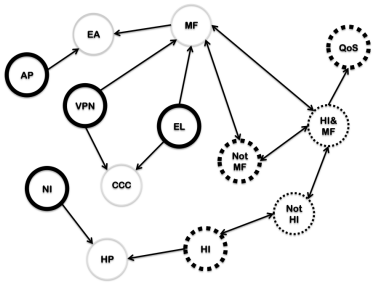


Fig. 3: Sceptically (thick), credulously (dotted) accepted, rejected (grey).

tical acceptance of arguments in such three extensions, we obtain that AP, VPN, EL, NI are sceptically accepted (i.e., "always"). This means that, for the attack-/countermeasure scenario we have depicted, having audit procedures, a virtual private network, an encrypted line, and a network IDS is always considered a valid argument. Therefore, they correspond to a strong suggestion for the security administrator. On the other hand, there are some other arguments that are rejected (see Def. 3), that is they never appear in such extensions; for instance EA, HP, MF, and CCC. All three threats are successfully "avoided", in the sense that adopted security countermeasures always prevent all of them. Moreover, also adopting the monitoring functionality countermeasure is not a good idea given this scenario, since it is rejected as well. Finally, the remaining arguments appear sometimes but not always in such three extensions (they are credulously accepted, according to Def. 3): NotHI (in 1 extension), HI&MF (1), HI (2), NotMF (2), QoS (2). The number of times they appear is visually highlighted in SecArg by filling arguments with different shades of grey, and also returning the appearance ratio, e.g,. 66.6% for QoS and 33.3% for NotHI. This can be interpreted as a strength-score for these arguments: for instance, having an host IDS beats not having it (2 to 1): hence the administrator is recommended to use it. For the sake of presentation, in Fig. 3 we use thick continuous circles for sceptically accepted arguments, thin/thick dotted circles for credulously accepted ones (respectively for lower/higher ratio of appearance, e.g., QoS is thicker than NotHI), and light-grey circles for rejected arguments.

4 Related and Future Work

Since the application of Argumentation to Cybersecurity-related issues is relatively a new field (or, at least, not deeply investigated), there is a few related work to be mentioned. A bunch of works applying Argumentation-based conflict-resolution to the specific case of firewall rules are [1–3]. In our approach, however, we would like to provide a general reasoning-tool.

In [8] the authors suggest the use of Argumentation to provide automated support for Cybersecurity decisions. Three different tasks where Argumentation can contribute are surveyed in the paper: first, the establishment of a security policy, drawing from a range of information on best practice and taking into account likely attacks and the vulnerability of the system to those attacks. Secondly, the process diagnosis to determine if an attack is underway after some apparent anomaly in system operation is detected; the final goal is to decide what action, if any, should be taken to ensure system integrity. At last, Argumentation can be used to reconfigure a security policy in the aftermath of a successful attack: this reconfiguration needs to ensure protection against future similar-attacks, without creating new vulnerabilities.

In [7] the authors propose how arguments can support the decision making process: the aim is to help the system security administrator to react (or not) to possible ongoing attacks. For instance, a decision can be taken either to disable traffic through port 80 or not to disable it.

In the next future we would like to extend SecArg from both the theoretical and practical point of view by *i)* interactively changing the AAF with a new node or attack and immediately see how much such modification impacts on the strength of arguments; *ii)* selecting a subset S of arguments and get the minimal amount of change to the AAF that transforms S into an extension satisfying a given semantics (e.g., preferred).

References

1. Applebaum, A., Levitt, K.N., Rowe, J., Parsons, S.: Arguing about firewall policy. In: Verheij, B., Szeider, S., Woltran, S. (eds.) COMMA. Frontiers in Artificial Intelligence and Applications, vol. 245, pp. 91–102. IOS Press (2012)
2. Bandara, A.K., Kakas, A.C., Lupu, E.C., Russo, A.: Using argumentation logic for firewall policy specification and analysis. In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T. (eds.) DSOM. LNCS, vol. 4269, pp. 185–196. Springer (2006)
3. Bandara, A.K., Kakas, A.C., Lupu, E.C., Russo, A.: Using argumentation logic for firewall configuration management. In: Integrated Network Management. pp. 180–187. IEEE (2009)
4. Bistarelli, S., Rossi, F., Santini, F.: Benchmarking hard problems in random abstract AFs: The stable semantics. In: Computational Models of Argument - Proceedings of COMMA. FAIA, vol. 266, pp. 153–160. IOS Press (2014)
5. Bistarelli, S., Rossi, F., Santini, F.: A first comparison of abstract argumentation reasoning-tools. In: ECAI 2014 - 21st European Conference on Artificial Intelligence. FAIA, vol. 263, pp. 969–970. IOS Press (2014)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–357 (1995)
7. Martinelli, F., Santini, F.: Debating cybersecurity or securing a debate? - (position paper). In: Foundations and Practice of Security - 7th International Symposium, FPS 2014. LNCS, vol. 8930, pp. 239–246. Springer (2014)
8. Rowe, J., Levitt, K., Parsons, S., Sklar, E., Applebaum, A., Jalal, S.: Argumentation logic to assist in security administration. In: Proceedings of the 2012 Workshop on New Security Paradigms. pp. 43–52. NSPW '12, ACM (2012)

SUNNY for Algorithm Selection: A Preliminary Study

Roberto Amadini, Fabio Biselli, Maurizio Gabbrielli, Tong Liu, and Jacopo Mauro

Department of Computer Science and Engineering
University of Bologna, Italy.

Abstract. Given a collection of algorithms, the Algorithm Selection (AS) problem consists in identifying which of them is the best one for solving a given problem. In this paper we show how we adapted the algorithm selector SUNNY, originally tailored for constraint solving, to deal with general AS problems. Preliminary investigations based on the AS Library benchmarks already show some promising results: for some scenarios SUNNY is able to outperform AS state-of-the-art approaches.

1 Introduction

Given a collection of algorithms, the *Algorithm Selection* (AS) problem basically consists in identifying which of them is the best one for solving a given problem. Initially proposed by Rice in 1976 [9], in the last decade AS has attracted some attention [7, 10]. In particular, the original notion of AS has been extended by the definition of *Algorithm Portfolio* (AP) [6]. In a nutshell, AP approaches exploit a portfolio $\{A_1, \ldots, A_m\}$ of different algorithms to get a globally better algorithm. They go beyond the original notion of AS introduced by Rice since APs perform the algorithm selection case-by-case instead of in advance. When a new, unseen problem p comes, an AP approach tries to predict which is (or which are) the best constituent algorithm(s) $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, with $1 \leq i_j \leq m$, for solving p and then runs such algorithm(s) on p. Scheduling $k > 1$ algorithms can reduce the risk of selecting only one algorithm —maybe the wrong one— and possibly enables the knowledge sharing between the scheduled algorithms. However, note that the boundary between AS and AP is fuzzy: these two related problems are often considered as equivalent. For this reason, with a little abuse of notation, in the following we will only use the AS notation for indicating both AS and AP problems.

SUNNY is an algorithm selector tailored for *Constraint Programming* (CP), where the algorithms to be selected correspond to different constraint solvers. Originally conceived for solving Constraint Satisfaction Problems (CSPs) only [1], it has been later on adapted for dealing with Constraint Optimisation Problems (COPs) [3]. SUNNY is also the algorithm that underpins `sunny-cp` [4], a constraint solver exploiting a portfolio of different constituent solvers for solving both CSPs and COPs.

In this paper we present a preliminary evaluation of SUNNY on different AS benchmarks taken from the *Algorithm Selection library* (ASlib) [5]. We show that SUNNY can be applied also outside the CP domain, reaching promising performance in different fields such as Answer-Set Programming (ASP), Quantified Boolean Formula (QBF), or the Container Pre-marshalling Problem. Conversely, for the Boolean Satisfiability (SAT) problems of ASlib there is still a performance gap with the best AS approaches.

2 SUNNY

The SUNNY [1] algorithm was originally introduced for constraint solving. Fixed a solving timeout τ and a portfolio $\mathcal{A}$ of algorithms, SUNNY exploits instances similarity to produce a sequential schedule $\sigma = [(A_1, t_1), \ldots, (A_h, t_h)]$ where algorithm $A_i \in \mathcal{A}$ has to run for t_i seconds and $\sum_{i=1}^{h} t_i = \tau$. For any input problem x, SUNNY uses a k-*Nearest Neighbours* (k-NN) algorithm to select from a training set of known instances the subset $N(x, k)$ of the k instances closer to the *feature vector* of x according to the Euclidean distance. Basically, the feature vector of x is a collection $F(x) \in \mathbb{R}^d$ of numerical attributes that characterise x (e.g., statistics over the variables or the constraints of x). Starting from the $N(x, k)$ instances SUNNY relies on three heuristics to compute the schedule σ: H_{sel}, for *selecting* the most promising algorithms $\{A_1, \ldots, A_h\} \subseteq \mathcal{A}$ to run; H_{all}, for *allocating* to each $A_i \in \mathcal{A}$ a certain runtime $t_i \in [0, \tau]$ for $i = 1, \ldots, h$; H_{sch}, for *scheduling* the sequential execution of the algorithms according to their presumed speed. The heuristics H_{sel}, H_{all}, and H_{sch} depends on the application domain. For CSPs, H_{sel} selects the smallest sub-portfolio $S \subseteq \mathcal{A}$ that solves the most instances in $N(x, k)$, by using the runtime for breaking ties. H_{all} allocates to each $A_i \in S$ a time t_i proportional to the instances that S can solve in $N(x, k)$, by using a special *backup solver* for covering the instances of $N(x, k)$ not solvable by any solver. Finally, H_{sch} sorts the solvers by increasing solving time in $N(x, k)$. For COPs the approach is similar, but different evaluation metrics are used. We conclude the section by showing an example of how SUNNY works on a given CSP; for more details about SUNNY we refer the interested reader to [1, 3].

Example 1 *Let x be a CSP, $\mathcal{A} = \{A_1, A_2, A_3, A_4\}$ a portfolio, A_3 the backup solver, $\tau = 1800$ seconds the solving timeout, $N(x, k) = \{x_1, ..., x_5\}$ the $k = 5$ neighbours of x, and the runtimes of solver A_i on problem x_j defined as in Table 1. In this case, the smallest sub-portfolios that solve the most instances (4 to be precise) in $N(x, k)$ are $\{A_1, A_2, A_3\}$, $\{A_1, A_2, A_4\}$, and $\{A_2, A_3, A_4\}$. The heuristic H_{sel} selects $S = \{A_1, A_2, A_4\}$ because these solvers are faster in solving the instances in $N(x, k)$. Since A_1 and A_4 solve 2 instances, A_2 solves 1 instance and x_1 is not solved by any solver, the time window $[0, \tau]$ is partitioned in $2 + 2 + 1 + 1 = 6$ slots: 2 assigned to A_1 and A_4, 1 slot to A_2, and 1 to the backup solver A_3. Finally, H_{sch} sorts the solvers by increasing solving time. The final schedule produced by SUNNY is therefore $\sigma = [(A_4, 600), (A_1, 600), (A_3, 300), (A_2, 300)]$.*

	x_1	x_2	x_3	x_4	x_5
A_1	τ	τ	**3**	τ	**278**
A_2	τ	**593**	τ	τ	τ
A_3	τ	τ	**36**	**1452**	τ
A_4	τ	τ	τ	**122**	**60**

Table 1. Runtimes (in seconds). τ means the solver timeout.

Scenario	m	n	d	τ	*Domain*
ASP	11	1294	138	600	Answer-Set Programming
CSP	2	2024	86	5000	Constraint Satisfaction Problem
MAXSAT	6	876	37	2100	Maximum Satisfiability Problem
PREMARSH	4	527	16	3600	Container Pre-marshalling Problem
PROTEUS	22	4021	198	3600	CSP, with possible encoding into SAT
QBF	5	1368	46	3600	Quantified Boolean Formula
SAT11-HAND	15	296	115	5000	SAT 2011 Competition – Handcrafted problems
SAT11-INDU	18	300	115	5000	SAT 2011 Competition – Industrial problems
SAT11-RAND	9	600	115	5000	SAT 2011 Competition – Random problems
SAT12-ALL	31	1614	115	1200	SAT Challenge 2012 – All problems
SAT12-HAND	31	767	115	1200	SAT Challenge 2012 – Handcrafted problems
SAT12-INDU	31	1167	115	1200	SAT Challenge 2012 – Industrial problems
SAT12-RAND	31	1362	115	1200	SAT Challenge 2012 – Random problems

Table 2. ASlib Scenarios.

3 Evaluation

To evaluate SUNNY on different scenarios we exploited the Algorithm Selection library (ASlib). ASlib provides standardised format and data for representing AS scenarios allowing the comparison of different AS approaches. Each ASlib scenario contains: an algorithm space $\mathcal{A} = \{A_1, \ldots, A_m\}$; a problem space $\mathcal{X} = \{x_1, \ldots, x_n\}$; a feature space $\mathcal{F}_d = \{F_1, \ldots, F_n\}$ where $F_j \in \mathbb{R}^d$ is the feature vector of the problem x_j; a performance space $\mathcal{P}_\tau = \{P_{1,1}, \ldots, P_{m,n}\}$ where $P_{i,j} \in \mathbb{R}$ measures the performance of algorithm A_i on problem x_j within a timeout of τ seconds. ASlib contains 13 heterogeneous scenarios[1] as summarised in Table 2. The scenarios differ in the number of algorithms m, problems n, features d, and in the time limits τ. For every scenario, the runtime is used as performance measure: if algorithm A solves problem x in $t < \tau$ seconds the runtime $\texttt{RunTime}(A, x)$ of A on x is t. Otherwise, $\texttt{RunTime}(A, x) = \tau$. Each scenario of the ASlib is evaluated with a *10-fold cross validation*: $\mathcal{X}$ is partitioned in 10 subsets $\mathcal{X}_1, \ldots, \mathcal{X}_{10}$ called folds, treating in turn a fold $\mathcal{X}_i$ as the test set and the union $\bigcup_{j \neq i} \mathcal{X}_j$ of the other folds as the training set.

Adapting SUNNY to ASlib scenarios was rather straightforward. Fixed a training set $\mathcal{X}_{tr} \subseteq \mathcal{X}$ and a corresponding feature space $\mathcal{F}_{tr}$, we normalised the feature vectors by removing all the constant features of $\mathcal{F}_{tr}$ and scaling them in the range $[-1, 1]$. Then, for each unknown problem $x \notin \mathcal{X}_{tr}$, SUNNY computes the neighbourhood $N(x, k) \subseteq \mathcal{X}_{tr}$ and the resulting schedule $\sigma = [(a_1, t_1), \ldots, (a_h, t_h)]$ exactly as explained in Section 2. Following the methodology of [4], we set $k = \sqrt{|\mathcal{X}_{tr}|}$ and the backup solver as the algorithm of $\mathcal{A}$ having the lower average $\texttt{RunTime}$ in $\mathcal{X}_{tr}$.

Table 3 shows for each scenario the *Fraction of Solved Instances* (FSI) of SUNNY. As the name underlines, the FSI of an AS approach is the ratio between the number of instances it solves and all the instances of the scenario. SUNNY is compared against the state-of-the-art AS approaches reported in [5] (viz., ISAC, SNNAP, aspeed, claspfolio, claspfolio-pre, zilla, and LLAMA) and two additional baselines: the *Single Best Solver* (SBS), i.e., the algorithm in $\mathcal{A}$ with highest FSI, and the *Virtual Best Solver* (VBS), i.e., the oracle approach that for every $x \in \mathcal{X}$ always select the algorithm $A \in \mathcal{A}$ for

[1] We considered the 1.0.1 version of ASlib. For more details, we refer the reader to [5].

Scenario	*VBS*	*SBS*	ISAC	SNNAP	aspeed	claspfolio	claspfolio-pre	zilla	LLAMA	SUNNY
ASP	0.937	0.859	0.896	0.910	0.890	**0.923**	0.923	0.915	0.920	0.913
CSP	0.875	0.858	0.859	0.858	0.862	0.872	0.872	0.872	**0.873**	0.870
MAXSAT	0.853	0.769	0.823	0.818	0.845	0.844	0.844	**0.848**	0.841	0.842
PREMARSH	1	0.812	0.843	0.753	**0.956**	0.867	0.945	0.918	0.879	0.949
PROTEUS	0.887	0.628	0.812	0.794	**0.867**	0.832	0.855	0.838	0.835	0.859
QBF	0.77	0.577	0.692	0.615	0.745	0.744	0.753	0.746	0.751	**0.754**
SAT11-HAND	0.74	0.497	0.541	0.611	**0.676**	0.649	0.672	0.655	0.669	0.622
SAT11-INDU	0.843	0.717	0.710	0.740	0.710	**0.763**	0.763	0.717	0.750	0.730
SAT11-RAND	0.82	0.603	0.773	0.743	0.777	0.805	0.807	**0.810**	0.797	0.805
SAT12-ALL	0.988	0.753	0.752	0.880	0.778	0.917	0.916	0.926	**0.929**	0.893
SAT12-HAND	0.701	0.477	0.467	0.580	0.587	0.636	0.638	0.649	**0.653**	0.608
SAT12-INDU	0.821	0.736	0.735	0.777	0.719	**0.788**	0.779	0.775	0.775	0.743
SAT12-RAND	0.764	0.731	0.740	0.730	0.724	**0.744**	0.743	0.737	0.742	0.727

Table 3. Fraction of Solved Instances.

which $\texttt{RunTime}(A, x)$ is minimal. We can see that SUNNY is the best approach for the QBF scenario, and that for all the non-SAT scenarios it is rather close to the best performance. Conversely, for the SAT benchmarks its performance is quite poor.

The FSI metric is commonly used for comparing different AS approaches due to its simplicity and significance. However, it does not take into account the time needed to solve a problem. To capture also the timing aspects of the resolution, the *Penalised Average Runtime* (PAR) measure is often used. PAR_k represents the average time taken to solve the problems by giving a penalisation of $k \times \tau$ seconds for the instances not solved within the timeout τ.

Table 4 shows the results considering the average PAR_{10} score. In this case the *SBS* is the single algorithm having the lower PAR_{10} score. Not surprisingly, PAR_{10} is strongly anti-correlated to FSI and the results of 4 somehow reflect what observed in Table 3. However, some differences arise. For instance, in addition to QBF, by considering PAR_{10} SUNNY is the best approach also for the PROTEUS scenarios. This means that in this scenario aspeed solves few instances more than SUNNY, but SUNNY is on-average faster.

Scenario	*VBS*	*SBS*	ISAC	SNNAP	aspeed	claspfolio	claspfolio-pre	zilla	LLAMA	SUNNY
ASP	400.2	880.5	653.6	571.2	711.2	**487.2**	496.0	539.5	509.1	549.0
CSP	6344.3	7201.6	7148.6	7201.6	7163.0	6511.2	6521.4	6491.5	**6466.7**	6617.8
MAXSAT	3127.2	4893.1	3763.1	3855.7	3748.4	3320.4	3629.3	**3234.7**	3367.6	3354.9
PREMARSH	227.6	7002.9	5880.8	9042.1	**1964.1**	5025.0	2395.7	3179.1	4634.2	2221.5
PROTEUS	4105.9	13443.4	6782.5	7430.3	5363.4	6075.1	5525.0	5900.4	6066.6	**5254.4**
QBF	8337.1	15330.2	11201.3	13954.0	9714.3	9333.6	9089.8	9222.5	9075.0	**9064.8**
SAT11-HAND	13360.7	25649.1	23325.1	19820.4	**16688.4**	17975.7	16897.8	17602.4	16906.5	19308.5
SAT11-INDU	8187.5	14605.9	14968.9	13426.8	15008.2	**12322.7**	12383.9	14621.2	12996.6	14014.9
SAT11-RAND	9186.4	19916.4	11575.1	12984.4	11589.0	9982.4	9936.2	**9719.6**	10341.2	9960.262
SAT12-ALL	241.3	3079.9	3101.0	1558.6	2810.2	1113.0	1163.2	1014.7	**980.3**	1429.8
SAT12-HAND	3662.2	6338.9	6466.3	5112.3	5071.9	4450.4	4459.4	4306.3	**4252.9**	4808.3
SAT12-INDU	2221.5	3266.0	3306.2	2796.8	3499.9	**2653.5**	2800.2	2838.4	2837.4	3211.891
SAT12-RAND	2872.8	3271.1	3168.8	3289.9	3382.1	**3119.6**	3161.9	3207.6	3149.6	3327.9

Table 4. Penalised Average Runtime.

4 Conclusions

In this work we presented an evaluation of SUNNY algorithm on different Algorithm Selection (AS) scenarios coming from the Algorithm Selection library (ASlib). Despite SUNNY is tailored for constraint solving, its adaptation to AS appears to be promising also in other fields such as Answer-Set Programming (ASP), Quantified Boolean Formula (QBF), or the Container Pre-marshalling Problem. Conversely, for the Boolean Satisfiability (SAT) problems there is still a performance gap with the best approaches.

We would like to remark that in this evaluation we used the default SUNNY approach without leveraging its settings to fit the different scenarios. As a future work we would like to try to improve the performance of SUNNY by using well-known techniques like pre-solving, parameters tuning, and feature selection. It would be interesting to consider also different scenarios, like optimisation and planning problems. Indeed, the ASlib currently contains a limited number of scenarios for which the only metric is the runtime. It would be nice also to perform a deeper study to better understand the SUNNY performance (and in particular why SUNNY is not so good for the SAT benchmarks).

We strongly encourage the submission of new scenarios and new algorithm selectors to the ASlib in order to foster the study and the comparison of new and better AS approaches. For instance, since SUNNY turns out to be the best approach for QBF, it would be interesting to consider a comparison with the multi-engine solver AQME [8].

We are currently implementing SUNNY as an automated algorithm selector for ASlib scenarios, with the aim of enrolling it to the next ICON Challenge on Algorithm Selection. Moreover, we are also interested in studying how SUNNY can be optimally parallelised to run its algorithms simultaneously on multiple cores. A preliminary investigation on the SUNNY parallelisation for CSPs and COPs is presented in [2].

References

1. R. Amadini, M. Gabbrielli, and J. Mauro. SUNNY: a Lazy Portfolio Approach for Constraint Solving. *TPLP*, 14(4-5):509–524, 2014.
2. R. Amadini, M. Gabbrielli, and J. Mauro. A Multicore Tool for Constraint Solving. In *IJCAI*, 2015. Pre-print available at: `http://arxiv.org/abs/1502.03986`.
3. R. Amadini, M. Gabbrielli, and J. Mauro. Portfolio approaches for constraint optimization problems. *AMAI*, pages 1–18, 2015.
4. R. Amadini, M. Gabbrielli, and J. Mauro. SUNNY-CP: a Sequential CP Portfolio Solver. In *SAC*, 2015. Available at `http://www.cs.unibo.it/~amadini/sac_2015.pdf`.
5. Algorithm Selection Library - coseal. `https://code.google.com/p/coseal/wiki/AlgorithmSelectionLibrary`.
6. C. P. Gomes and B. Selman. Algorithm portfolios. *Artif. Intell.*, 126(1-2):43–62, 2001.
7. L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60, 2014.
8. L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints*, 14(1):80–116, 2009.
9. J. R. Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65–118, 1976.
10. K. A. Smith-Miles. Towards insightful algorithm selection for optimisation using meta-learning concepts. In *IJCNN*, pages 4118–4124. IEEE, 2008.

Indice degli autori