Implementing the Fatio Protocol for Multi-Agent Argumentation in LogiKEy

Luca Pasetto[1,*], Christoph Benzmüller[2,3]

[1] *University of Luxembourg*
[2] *Otto-Friedrich-Universität Bamberg*
[3] *Freie Universität Berlin*

Abstract
Fatio is a dialogue protocol that has been proposed to extend the language for agent communications FIPA ACL with locutions for dialectical argumentation. Its syntax is composed of five agent locutions, and its axiomatic semantics is defined in terms of the mental states (beliefs and desires) of the participating agents, that are influenced by what is uttered. LogiKEy is a framework and methodology for the design and engineering of ethico-legal reasoners which is based on semantical embeddings of logics and logic combinations in expressive classical higher-order logic (HOL). In this paper, we explore a modelling and present a first implementation of the axiomatic semantics of Fatio in the Isabelle/HOL proof assistant system following the LogiKEy methodology.

Keywords
Proof assistants, Isabelle/HOL, Knowledge representation and reasoning, Automated theorem proving, Semantical embedding, Higher-order logic, Dialogue, Argumentation, Multi-Agent systems

1. Introduction

Recent advancements in dialogue systems based on language models are pushing the boundaries of what is possible in AI-human interaction. The rapid evolution of these technologies can set up the stage for more intuitive and helpful AI systems in everyday life, but at the same time presents a variety of risks. This motivates research on formal dialogue systems, based on logical languages, for which it is possible to formally (and eventually also mechanically) prove relevant properties. In this paper, we explore an *Isabelle/HOL* modelling of *Fatio*, a formal dialogue system based on argumentation between multiple agents by following the *LogiKEy* [1] framework and methodology. For an overview of argumentation-based dialogue, the interested reader can refer to [2].

LogiKEy [1] is a framework and methodology that can be used for the design, engineering and experimentation of reasoning systems, with a focus on ethico-legal applications [3] and normative reasoning [4]. The formal framework of LogiKEy is based on *shallow semantical embeddings* (SSE) [5] of combinations of various logics in classical higher-order logic (HOL). This meta-logical approach allows the use of off-the-shelf theorem provers and model finders for HOL, and therefore aids designers of ethical intelligent agents by allowing them to employ existing technologies instead of having to face the cumbersome task of developing completely new tools. Additionally, continuous improvements to the theorem provers enhance the reasoning capabilities within LogiKEy without needing extra adjustments. The methodology is represented in Figure 1: the relevant object logics are encoded in HOL and can in turn be used to express domain theories, with the purpose of successively experimenting with concrete applications and examples.

The rest of the paper is structured as follows: Section 2 describes the Fatio protocol as in [6], Section 3 shows its LogiKEy implementation in the Isabelle/HOL proof assistant system, and Section 4 discusses the relevant results and the remaining challenges.

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France
[*] Corresponding author.
luca.pasetto@uni.lu (L. Pasetto); christoph.benzmueller@uni-bamberg.de (C. Benzmüller)
0000-0003-1036-1718 (L. Pasetto); 0000-0002-3392-3093 (C. Benzmüller)

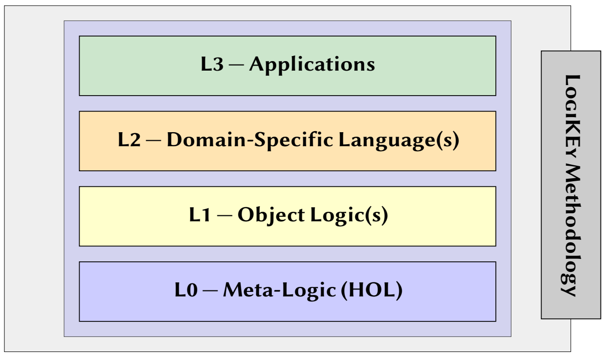


Figure 1: LogiKEy development methodology

2. The Fatio Protocol

This section summarizes the Fatio protocol for dialectical argumentation between agents that has been introduced in [6]. The original paper motivates the protocol as an extension of the language FIPA ACL [7] for agent interactions with the goal of addressing the criticism that it lacks a proper theory of argumentation. The authors therefore provide a set of locutions for the requesting and providing of reasons. The intuition is that agents that are proponents or opponents of a claim aim to establish the truth through various locutions or dialectical moves. The protocol is defined with a syntax, an axiomatic semantics, and an operational semantics.

2.1. Syntax

The general syntax for utterances is $\texttt{illocution}(P_i, \varphi)$ or $\texttt{illocution}(P_i, P_j, \varphi)$, where $\texttt{illocution}$ is an illocutionary act, P_i is an identifier for the agent making the utterance (the speaker), P_j (with $P_j \neq P_i$) denotes an agent at whom the utterance is directed, and φ is the content of the utterance. Utterances are intended to be made to the entire group involved in the dialogue. For the content of the utterance, any agreed formal language can be used, but the authors assume that the content layer is represented in a propositional language, with lower-case Greek letters as propositions. The set of well-formed content formulae, closed under the usual connectives, is denoted as $\mathcal{C}$. Since the protocol is used to exchange justifications for claims, some utterances also contain content comprising arguments (e.g., premises and inference-rules), which are represented by upper-case Greek letters. The set of well-formed argument formulae, closed under the usual connectives, is denoted as $\mathcal{A}$, with $\mathcal{C} \subset \mathcal{A}$. If $\varphi \in \mathcal{C}$ is a proposition in the content language and $\Phi \in \mathcal{A}$ is a justification, $\Phi \vdash^{+} \varphi$ is written to indicate that Φ is an argument in support of φ, and $\Phi \vdash^{-} \varphi$ to indicate that Φ is an argument against φ. In order to keep track of commitments to provide justifications for utterances, the concept of a dialectical obligation is introduced and is used to refer to previous commitments in the dialogue. The syntax is made up of five locutions or dialectical moves [6]:

- **F1 – $\texttt{assert}(P_i, \varphi)$**: A speaker P_i asserts a statement φ. By doing so, P_i creates a dialectical obligation to provide justification for φ, if subsequently required by another participant.
- **F2 – $\texttt{question}(P_j, P_i, \varphi)$**: A speaker P_j questions a prior utterance of $\texttt{assert}(P_i, \varphi)$ and seeks a justification for φ. The speaker P_j of the question creates no dialectical obligations.
- **F3 – $\texttt{challenge}(P_j, P_i, \varphi)$**: A speaker P_j challenges a prior utterance of $\texttt{assert}(P_i, \varphi)$ and seeks a justification for φ. Unlike a question, P_j also creates a dialectical obligation on himself to provide a justification for not asserting φ, such as an argument against φ, if questioned or challenged. Therefore, $\texttt{challenge}(P_j, P_i, \varphi)$ is considered stronger than $\texttt{question}(P_j, P_i, \varphi)$.
- **F4 – $\texttt{justify}(P_i, \Phi \vdash^{+} \varphi)$**: A speaker P_i, who had previously uttered $\texttt{assert}(P_i, \varphi)$ and was then questioned or challenged, is able to provide a justification Φ for the initial statement φ.

- **F5 – `retract`(P_i, φ)**: A speaker P_i, who had previously made an assertion `assert`(P_i, φ) or justification `justify`$(P_i, \Phi \vdash^+ \varphi)$, can withdraw this statement with the utterance of `retract`(P_i, φ) or `retract`$(P_i, \Phi \vdash^+ \varphi)$, respectively. This removes the earlier dialectical obligation on P_i to justify φ or Φ, if questioned or challenged.

The locutions are also subject to combination rules, introduced in [8], that constrain when an utterance can be made within the dialogue, ensuring structured interaction based on assertions, questions, challenges, justifications, and retractions. For instance, they ensure that the utterance `assert`(P_i, φ) may be made at any time, and that immediately following an utterance of `question`(P_j, P_i, φ) or `challenge`(P_j, P_i, φ), the speaker P_i of `assert`(P_i, φ) must reply with `justify`$(P_i, \Phi \vdash^+ \varphi)$, for some $\Phi \in \mathcal{A}$.

2.2. Axiomatic Semantics

The axiomatic semantics gives a set of axioms in terms of pre-conditions and post-conditions for each locution of Fatio, and it involves beliefs and desires of the participating agents. Mental states such as beliefs and desires are used in order to be consistent with the axiomatic semantics of FIPA ACL. The concept of a dialectical obligation is also clarified at this point, as the authors introduce, for each participant P_i, one *dialectical obligations store* $DOS(P_i)$ to record dialectical obligations. The contents of $DOS(P_i)$ are triples (P_i, X, Y), where P_i is a participant, $X \in \mathcal{C}$ or $X \in \mathcal{A}$, and $Y \in \{+, -\}$. The intuition is that $(P_i, \varphi, +) \in DOS(P_i)$ indicates that participant P_i has a dialectical obligation to provide a justification in support of φ. Two classes of modal operators are used in the semantics: one for beliefs B_i and one for desires D_i, where i is an agent identifier. Also, an element from FIPA's action language is used: `Done` [*illocution*(P_i, φ), *pre-con*] indicates that *illocution*(P_i, φ) has been uttered by participant P_i with content φ, and with pre-conditions *pre-con* true. For the purposes of the current implementation, we do not consider *pre-con* and use `Done` [*illocution*(P_i, φ)]. The pre-conditions and post-conditions are then [6]:

- `assert`(P_i, φ)
  - **Pre-conditions:** A speaker P_i does not have yet the dialectical obligation to provide justification for φ, and P_i desires that each participant $P_j (j \neq i)$ believes that P_i believes the proposition $\varphi \in C$. Formally, $((P_i, \varphi, +) \notin DOS(P_i)) \ \wedge \ (\forall j \neq i)(D_i B_j B_i \varphi)$.
  - **Post-conditions:** Each participant $P_k (k \neq i)$, believes that participant P_i desires that each participant $P_j (j \neq i)$ believe that P_i believes φ. Moreover, the dialectical obligation $(P_i, \varphi, +)$ is added to $DOS(P_i)$.
    Formally, $(P_i, \varphi, +) \in DOS(P_i) \ \wedge \ (\forall k \neq i)(\forall j \neq i)(B_k D_i B_j B_i \varphi)$.
- `question`(P_j, P_i, φ)
  - **Pre-conditions:** Some participant $P_i (i \neq j)$ has a dialectical obligation to support φ and participant P_j desires that each other participant $P_k (k \neq j)$, believes that P_j desires that P_i utter a `justify` locution for φ. Formally, $\exists i (i \neq j)$ $((P_i, \varphi, +) \in DOS(P_i) \ \wedge \ (\forall k \neq j) D_j B_k D_j \ (\exists \Delta \in \mathcal{A})$ `Done` [`justify`$(P_i, \Delta \vdash^+ \varphi)$]) .
  - **Post-conditions:** Participant P_i must utter a `justify` locution. There is no change on the dialectical obligations. Formally, $(\exists \Delta \in \mathcal{A})$ `Done` [`justify`$(P_i, \Delta \vdash^+ \varphi)$] .
- `challenge`(P_j, P_i, φ)
  - **Pre-conditions:** Some participant $P_i (i \neq j)$ has a dialectical obligation to support φ. Participant P_j desires that each other participant $P_k (k \neq j)$, believe both that P_j desires that P_i utter a `justify`$(P_i, \Delta \vdash^+ \varphi)$ locution for some $\Delta \in A$ and that P_j does not believe φ. Formally, $\exists i (i \neq j) \ ((P_i, \varphi, +) \in DOS(P_i) \wedge (\forall k \neq j)(D_j B_k \neg B_j \varphi)$
    $\wedge \ (\forall k \neq j) D_j B_k D_j \ (\exists \Delta \in \mathcal{A})$ `Done` [`justify`$(P_i, \Delta \vdash^+ \varphi)$]).
  - **Post-conditions:** Participant P_i must utter a `justify` locution. Moreover, the dialectical obligation $(P_j, \varphi, -)$ is added to $DOS(P_j)$. Formally, $((P_j, \varphi, -) \in DOS(P_j)$ $\wedge \ (\exists \Delta \in \mathcal{A})$ `Done` [`justify` $(P_i, \Delta \vdash^+ \varphi)$]).

- `justify`$(P_i, \Phi \vdash^+ \varphi)$
  - **Pre-conditions:** A speaker P_i has a dialectical obligation to support φ. Another speaker $P_j (j \neq i)$ has uttered a `question`(P_j, P_i, φ) or a `challenge`(P_j, P_i, φ) locution. Furthermore, P_i desires that each participant $P_k (k \neq i)$ believes that P_i believes that $\Phi \in A$ is an argument for φ. Formally, $((P_i, \varphi, +) \in DOS(P_i)) \wedge (\texttt{Done}\,[\texttt{question}\,(P_j, P_i, \varphi)] \vee \texttt{Done}\,[\texttt{challenge}\,(P_j, P_i, \varphi)) \wedge (\exists \Phi \in \mathcal{A})\,(\forall k \neq i)\, D_i B_k B_i\,(\Phi \vdash^+ \varphi)$.
  - **Post-conditions:** Each participant $P_k (k \neq i)$ believes that P_i desires that each participant $P_j (j \neq i)$ believes that P_i believes that $\Phi \in A$ is an argument for φ. Moreover, the dialectical obligation $(P_i, \Phi, +)$ is added to $DOS(P_i)$. Formally, $((P_i, \Phi, +) \in DOS(P_i)) \wedge (\forall k \neq i)\,(\forall j \neq i)\, B_k D_i B_j B_i\,(\Phi \vdash^+ \varphi)$.
- `retract`(P_i, φ)
  - **Pre-conditions:** For the proposition $\varphi \in C$, it is required that $(P_i, \varphi, +) \in DOS(P_i)$. Participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer believes φ. Additionally, P_i desires that each participant $P_j (j \neq i)$ understands that P_i no longer maintains a disbelief in φ. Formally, $((P_i, \varphi, +) \in DOS(P_i) \wedge (\forall j \neq i)\, D_i B_j \neg B_i \varphi) \vee ((P_i, \varphi, -) \in DOS(P_i) \wedge (\forall j \neq i)\, D_i B_j \neg\neg B_i \varphi)$.
  - **Post-conditions:** Depending on the case outlined in the pre-conditions, either each participant $P_k (k \neq i)$ believes that participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer believes φ, or each participant $P_k (k \neq i)$ believes that participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer disbelieves φ. Moreover, either $(P_i, \varphi, +)$ or $(P_i, \varphi, -)$ is removed from $DOS(P_i)$. Formally, $((P_i, \varphi, +) \notin DOS(P_i) \wedge (\forall k \neq i)\,(\forall j \neq i)\, B_k D_i B_j \neg B_i \varphi) \vee ((P_i, \varphi, -) \notin DOS(P_i) \wedge (\forall k \neq i)\,(\forall j \neq i)\, B_k D_i B_j \neg\neg B_i \varphi)$.

2.3. Operational Semantics

The operational semantics is given in terms of internal decision mechanisms for agents that in the protocol are not specified and are left to be implemented, for example through reasoning methods based on formal argumentation. For instance, a decision mechanism is used by an agent to decide whether to assert a given statement at a given point of the dialogue, and another decision mechanism is used by an agent with a dialectical obligation to justify a statement or argument to identify the best possible justification at a given point of the dialogue. We do not go into more details of the operational semantics here, as for now we only focused on the encoding of the axiomatic semantics.

3. Fatio in LogiKEy

The possibility of using the existing LogiKEy infrastructure has allowed for rapid prototyping of the Fatio protocol in the Isabelle/HOL proof assistant system [9]. Figure 2 shows the layers of the LogiKEy methodology applied to Fatio.

The layers are as follows:

- **L0** is the underlying Meta-Logic (HOL) layer;
- **L1** represents the combination of object logics: (L1.1) a deeply embedded logic to be used to express the content of the Fatio locutions and (L1.2) a shallowly embedded logic to represent the axiomatic semantics in terms of beliefs and desires;
- **L2** is about domain-specific languages: (L2.1) is the encoding of world knowledge using L1.2 and (L2.2) is the actual language of Fatio, based on agents uttering locutions; and
- **L3** is the layer for dialogue applications, that use in turn the L2 languages.

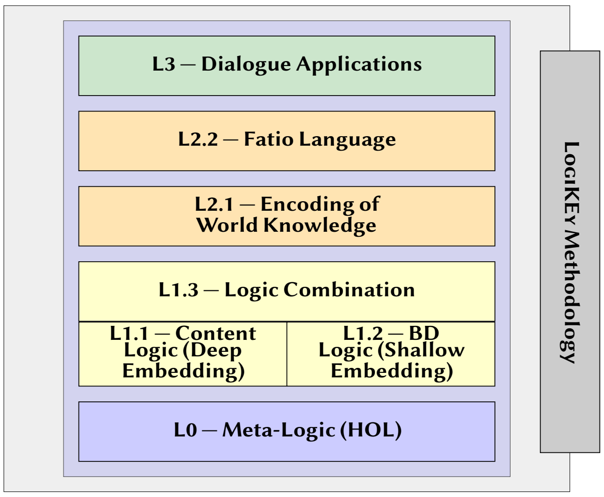

L3 — Dialogue Applications
L2.2 — Fatio Language
L2.1 — Encoding of World Knowledge
L1.3 — Logic Combination
L1.1 — Content Logic (Deep Embedding)
L1.2 — BD Logic (Shallow Embedding)
L0 — Meta-Logic (HOL)
LogiKEy Methodology

Figure 2: LogiKEy development methodology for Fatio

The architecture shows that at layer L1 we have both a shallow and a deep embedding of logics, but additionally the situation is even more complex, as the two logics are also nested within each other. We now go through the main parts of the implementation and describe choices and challenges.[1]

We begin with layer L1.2, showing the higher-order multimodal logic that we then use to represent beliefs and desires of agents. Here, σ is the type of world depended formulas, and complex formulas are built by following the semantics of the logical connectives.

```
typedecl i (*Type of possible worlds.*)
typedecl μ (*Type of individuals.*)
type_synonym σ="(i⇒bool)" (*Type of world depended formulas (truth sets).*)
type_synonym α="(i⇒i⇒bool)" (*Type of accessibility relations between worlds.*)
(*Lifted HOMML connectives: they operate on world depended formulas (truth sets).*)
definition mtop :: "σ" ("⊤") where "⊤ ≡ λw. True"
definition mbot :: "σ" ("⊥") where "⊥ ≡ λw. False"
definition mneg :: "σ⇒σ" ("¬_"[52]53)  where "¬φ ≡ λw. ¬φ(w)"
definition mand :: "σ⇒σ⇒σ" (infixr"∧"51) where "φ∧ψ ≡ λw. φ(w)∧ψ(w)"
definition mor  :: "σ⇒σ⇒σ" (infixr"∨"50) where "φ∨ψ ≡ λw. φ(w)∨ψ(w)"
definition mimp :: "σ⇒σ⇒σ" (infixr"→"49) where "φ→ψ ≡ λw. φ(w)⟶ψ(w)"
definition mequ :: "σ⇒σ⇒σ" (infixr"↔"48) where "φ↔ψ ≡ λw. φ(w)⟷ψ(w)"
definition mall :: "('a⇒σ)⇒σ" ("∀") where "∀Φ ≡ λw.∀x. Φ(x)(w)"
definition mallB:: "('a⇒σ)⇒σ" (binder"∀"[8]9) where "∀x. φ(x) ≡ ∀φ"
definition mexi :: "('a⇒σ)⇒σ" ("∃") where "∃Φ ≡ λw.∃x. Φ(x)(w)"
definition mexiB:: "('a⇒σ)⇒σ" (binder"∃"[8]9) where "∃x. φ(x) ≡ ∃φ"
definition mbox :: "α⇒σ⇒σ" ("□_ _") where "□ r φ ≡ (λw. ∀v. r w v ⟶ φ v)"
definition mdia :: "α⇒σ⇒σ" ("◇_ _") where "◇ r φ ≡ (λw. ∃v. r w v ∧ φ v)"
```

This logic is shallowly embedded, and after defining a datatype for speakers (for simplicity, at this moment we list only three possible speakers a, b and c) we define the two modal operators B_i and D_i for each speaker i.

```
datatype Speaker = a | b | c

consts BeliefRelationOf::"Speaker⇒α" ("@ᴮ_")
consts DesireRelationOf::"Speaker⇒α" ("@ᴰ_")

definition BeliefOfSpeaker::"Speaker⇒σ⇒σ" ("B_ _") where
  "Bₓ φ ≡ (□ (@ᴮ x) φ)"
definition DesireOfSpeaker::"Speaker⇒σ⇒σ" ("D_ _") where
  "Dₓ φ ≡ (□ (@ᴰ x) φ)"
```

[1]The code of the Isabelle/HOL implementation that we describe here is available at https://github.com/cbenzmueller/LogiKEy/tree/master/Fatio/v1

At layer L1.1 (on the left below) we have a deeply embedded propositional logic, with formulas of type `Formula`. This logic is then used inside the Fatio language of type `FatioL` to represent the content of the Fatio locutions: we show the syntax of the Fatio language at layer L2.2 (on the right below), with the five above mentioned kinds of locutions and content of type `Formula`.

```
datatype AtomExpression = p | q | r | n

datatype Formula =   —‹Formulas›
  Atom AtomExpression ("_ᵃ")
  | Bot ("⊥")
  | Neg Formula ("¬")
  | And Formula Formula (infixr "∧" 97)
  | Or Formula Formula (infixr "∨" 93)
  | Imp Formula Formula (infixr "→" 95)
```

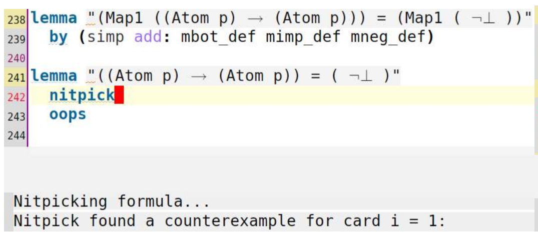
```
datatype Sign = Plus ("+") | Minus ("-")

datatype FatioL =   —‹Fatio Language›
  Assert Speaker Formula ("assert[_,_]")
  | Question Speaker Speaker Formula ("question[_,_,_]")
  | Justify Speaker Formula Sign Formula ("justify[_,_⊢_]")
  | Challenge Speaker Speaker Formula ("challenge[_,_,_]")
  | Retract Speaker Formula Sign ("retract[_,_,_]")
```

In the axiomatic semantics, the beliefs and desires of the agents are about formulas that have been uttered during the dialogue using the content language, that is, the deeply embedded one. However, the axiomatic semantics is expressed in the shallowly embedded higher-order multimodal logic. Therefore, at layer L1.3 it is necessary to have a mapping between the deeply embedded content formulas (L1.1) and the shallowly embedded higher-order multimodal logic formulas (L1.2). This is built starting from the atoms of the languages with `mkHOMMLatom` (on the left below) and arriving to the complex formulas with the function `Map1` specified on the relative connectives (on the right below). We also map to the higher-order multimodal logic the special formulas that appear in the axiomatic semantics for `Done` $[illocution(P_i, \varphi)]$ (with `MapDone`) and for $\Phi \vdash^+ \varphi$ in `justify`$(P_i, \Phi \vdash^+ \varphi)$ (with `MapEntailment`).

```
(* mappings to shallow embedding *)
consts mkHOMMLatom::"AtomExpression⇒σ"
consts MapDone::"FatioL⇒σ"
consts MapEntailment::"Sign⇒Formula⇒Formula⇒σ"
```

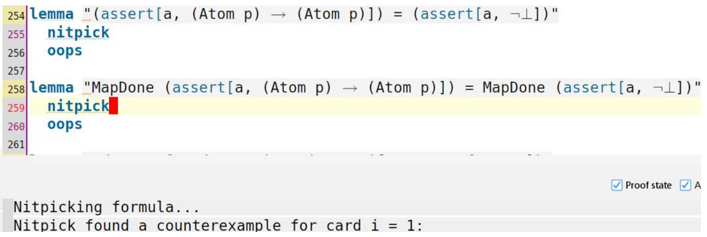
```
(* mapping of Formula, from deep to shallow *)
fun Map1::"Formula⇒σ" ("{_}") where
  "Map1 ((Atom x)) = mkHOMMLatom (x)"
| "Map1 (Bot) = mbot"
| "Map1 (Neg φ) = (¬(Map1 φ))"
| "Map1 ((And φ ψ)) = ((Map1  (φ)) ∧ (Map1 (ψ)))"
| "Map1 ((Or φ ψ)) = ((Map1  (φ)) ∨ (Map1 (ψ)))"
| "Map1 ((Imp φ ψ)) = ((Map1  (φ)) → (Map1 (ψ)))"
```

We show then some experiments to clarify the differences between the deep and the shallow embedding. First, while for the deeply embedded logic of L1.1 we can find a counterexample to show that it does not hold that $p \to\ p = \neg\bot$, when this is mapped to the shallowly embedded logic of L1.2 we can prove that it holds that `Map1`$(p \to\ p) =$ `Map1`$(\neg\bot)$ (on the left below). Indeed, the deep embedding is motivated by the need to differentiate two same Fatio locutions that involve different but semantically equivalent formulas of the content language, while with the shallow embedding we cannot distinguish between formulas like $p \to\ p$ and $\neg\bot$. We can find a counterexample to show that it does not hold that `assert`$(a, p \to\ p) =$ `assert`$(a, \neg\bot)$, but also there is a counterexample to show that it does not hold that `Done` $[assert(a, p \to\ p)] =$ `Done` $[assert(a, \neg\bot)]$ (on the right below). This is the intended behaviour for `Done`, as in this case we might be in a dialogue where agent a has asserted $\neg\bot$ but not $p \to\ p$.



```
lemma "(Map1 ((Atom p) → (Atom p))) = (Map1 ( ¬⊥ ))"
  by (simp add: mbot_def mimp_def mneg_def)

lemma "((Atom p) → (Atom p)) = ( ¬⊥ )"
  nitpick
  oops
```

```
Nitpicking formula...
Nitpick found a counterexample for card i = 1:
```



```
lemma "(assert[a, (Atom p) → (Atom p)]) = (assert[a, ¬⊥])"
  nitpick
  oops

lemma "MapDone (assert[a, (Atom p) → (Atom p)]) = MapDone (assert[a, ¬⊥])"
  nitpick
  oops
```

```
Nitpicking formula...
Nitpick found a counterexample for card i = 1:
```

We proceed now with the modelling of the axiomatic semantics of the protocol. The first ingredient is the *dialectical obligations store* DOS, that we represent as a list of triples (`Speaker, Formula, Sign`). The function to update DOS when a Fatio locution is executed is then shown below here on the right: `assert`, `justify`, and `challenge` add an element to DOS, `retract` removes an element from DOS, while `question` does not change DOS.

```
(* DOS Entry *)
datatype DOS = Entry Speaker Formula Sign
```

```
(* DOS update *)
fun
  DOSUpdate::"FatioL ⇒ DOS list ⇒ DOS list"
  where
    "(DOSUpdate assert[i,φ] dos) =
      ((Entry i φ +) # dos)"
  | "(DOSUpdate question[j,i,φ] dos) =
        dos"
  | "(DOSUpdate justify[i, ψ ⊢S φ] dos) =
      ((Entry i ψ +) # dos)"
  | "(DOSUpdate challenge[j,i,φ] dos) =
      ((Entry j φ -) # dos)"
  | "(DOSUpdate retract[i,φ,S] dos) =
      ((remove1 (Entry i φ S) dos))"
```

For layer L1.3, the next step is to encode the pre-conditions of each locution. In order to do that, the formal definitions of the original paper should be clear and not leave space for ambiguity. If we consider the pre-conditions of `question` and `challenge`, we have that $\exists\Delta$ appears after the modal operators $D_jB_kD_j$ in $(\forall k \neq j) D_jB_kD_j\ (\exists\Delta \in \mathcal{A})\,\texttt{Done}\,[\texttt{justify}(P_i, \Delta \vdash^{+} \varphi)]$. It is not completely clear if the authors actually intended the modal operators to operate on a quantified formula, or if they intended the existential quantifier as a meta-language element. To better understand the situation, we experimented with the expression and tried to move the existential quantifier outside of the modal operators, obtaining $(\forall k \neq j)\,(\exists\Delta \in \mathcal{A})\,D_jB_kD_j\texttt{Done}\,[\texttt{justify}(P_i, \Delta \vdash^{+} \varphi)]$. Indeed, the two formulas are not equivalent, and we found out that while the second one implies the first one (`lemma ent3`), the inverse does not hold (counterexample for `lemma ent2`). For now, we decided to follow strictly the contents of the paper, so we kept the first formula for the pre-conditions, but this might be a knowledge representation choice that can be changed in favor of the formula that better encodes the desired effects.

```
lemma ent2:"(⊨BD DⱼBₖDⱼ (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))) ⟶
            (∃Δ. (⊨BD DⱼBₖDⱼ (MapDone justify[i, Δ ⊢⁺ φ])))"
  unfolding FatioDefs
  unfolding Defs
  nitpick
  oops

lemma ent3:"(∃Δ. (⊨BD DⱼBₖDⱼ (MapDone justify[i, Δ ⊢⁺ φ])) ⟶
            (⊨BD DⱼBₖDⱼ (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))))"
  unfolding FatioDefs
  unfolding Defs
  by simp
```

We can then proceed with the encoding of the pre-conditions. We write function `PreCond` that checks whether the pre-conditions of a locution are satisfied, given a current *dialectical obligations store* DOS and a current set Γ of shallowly embedded formulas that represent the encoded world knowledge of layer L2.1.

```
fun PreCond::"FatioL ⇒ (DOS list) ⇒ (σ ⇒ bool) ⇒ bool"
  where
    "(PreCond assert[i,φ] dos Γ) =
     (¬(List.member dos (Entry i φ +)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DiBjBi {φ})))"
  | "(PreCond question[j,i,φ] dos Γ) =
      (¬(j = i) ∧ (List.member dos (Entry i φ +)) ∧
        (∀k. ¬(k = j) ⟶ ( (Γ ⊨ DjBkDj (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))))))"
  | "(PreCond justify[i, ψ ⊢S φ] dos Γ) =
      ((List.member dos (Entry i φ S)) ∧
       ( ∃j. ( (Γ ⊨ (MapDone question[j,i,φ]))  ∨  (Γ ⊨ (MapDone challenge[j,i,φ])) )
       ∧
      (∀k. ¬(k = i) ⟶ (Γ ⊨ DiBkBi (MapEntailment S ψ φ) )) ))"
  | "(PreCond challenge[j,i,φ] dos Γ) =
      (¬(j = i) ∧ (List.member dos (Entry i φ +)) ∧
        (∀k. ¬(k = j) ⟶ ((Γ ⊨ DjBk ¬Bj {φ}) ∧ (Γ ⊨ DjBkDj (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))))))"
  | "(PreCond retract[i,φ,S] dos Γ) =
      (if S=+ then
        ((List.member dos (Entry i φ +)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DiBj ¬Bi {φ})))
     else
        ((List.member dos (Entry i φ -)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DiBj ¬¬Bi {φ})))
    )"
```

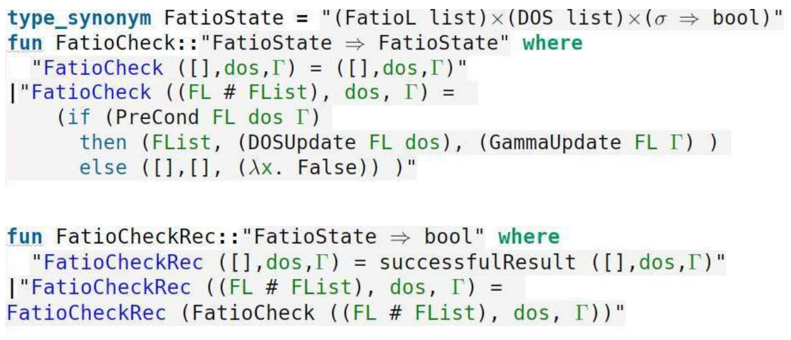
The post-conditions are encoded with the GammaAdd function, that takes a locution and gives the set of formulas representing its post-conditions. At this point, in the axiomatic semantics of the paper the post-conditions of question and challenge contain a formula $(\exists\Delta \in \mathcal{A})$ Done [justify$(P_i, \Delta \vdash^{+} \varphi)$], which means that the relevant justify locution is uttered. However, we think that this would be better handled by combination rules or operational semantics, and the justify locution cannot have been uttered already when the post-conditions of question and challenge are imposed. For this reason, we simplified the post-conditions of these two locutions.

```
fun GammaAdd::"FatioL ⇒ (σ ⇒ bool)"
  where
    "(GammaAdd assert[i,φ]) =
      (λx. (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BkDiBjBi {φ})))"
  | "(GammaAdd question[j,i,φ]) =
      (λx. True)" (* simplified *)
  | "(GammaAdd justify[i, ψ ⊢S φ]) =
      (λx. (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BkDiBjBi (MapEntailment S ψ φ) )))"
  | "(GammaAdd challenge[j,i,φ]) =
      (λx. True)" (* simplified *)
  | "(GammaAdd retract[i,φ,S]) =
      (λx. (if S=+ then
              (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BkDiBj ¬Bi {φ}))
            else
              (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BkDiBj ¬¬Bi {φ}))
       ))"
```

The actual evolution of the set of formulas Γ is encoded by GammaUpdate, that outputs the updated set of formulas Γ′ given a current locution and a current set of formulas Γ. It does so by including (1) the new formulas added with GammaAdd, (2) the formulas that are already in Γ and are consistent with (1), and (3) the relevant MapDone formula for the current locution.

```
fun
  GammaUpdate::"FatioL ⇒ (σ ⇒ bool) ⇒ (σ ⇒ bool)"
  where
    "(GammaUpdate statement Γ) =
     (λx. (Γ x ∧ (GammaKeep statement x)) ∨ (GammaAdd statement x) ∨ (x = MapDone statement) )"
```

A FatioState is defined as a triple (FatioL list, *DOS* list, Γ). The function FatioCheck goes from one FatioState to the next by executing its first locution, and the function FatioCheckRec executes an entire dialogue by a series of recursive calls, and outputs True if the final state is successful, that is, the pre-conditions of each executed locution are satisfied after having imposed the post-conditions of the previous locution.



```
type_synonym FatioState = "(FatioL list)×(DOS list)×(σ ⇒ bool)"
fun FatioCheck::"FatioState ⇒ FatioState" where
  "FatioCheck ([],dos,Γ) = ([],dos,Γ)"
|"FatioCheck ((FL # FList), dos, Γ) =
    (if (PreCond FL dos Γ)
      then (FList, (DOSUpdate FL dos), (GammaUpdate FL Γ) )
      else ([],[], (λx. False)) )"


fun FatioCheckRec::"FatioState ⇒ bool" where
  "FatioCheckRec ([],dos,Γ) = successfulResult ([],dos,Γ)"
|"FatioCheckRec ((FL # FList), dos, Γ) =
FatioCheckRec (FatioCheck ((FL # FList), dos, Γ))"
```

4. Discussion

We presented a first modelling and implementation of the Fatio protocol in the LogiKEy framework. The next step is obviously to carry out more extensive experiments on it, in order to test it and possibly to improve it accordingly. At this point, we ran only a few experiments with some small dialogues. While the reasoning system is promising, and up to our knowledge it is the first implementation of Fatio that actually uses the intended modal logic semantics, the modelling is not simple and there are a few sources of complexity that emerged:

- *complexity of the relationship between the embeddings of the logics* (the deep embedding is being mapped to the shallow embedding, and the Fatio language, that uses the deeply embedded logic, is again being mapped to the shallow embedding);
- *complexity in terms of how Isabelle/HOL can cope with the modelling*, as
  - we had a few technical issues related to the parsing of the formulas combining different logics, and in order to solve them at this point we are using a large quantity of brackets to speed-up the disambiguation process, and
  - we had some issues regarding the responsiveness of the theorem provers called by `sledgehammer`, caused by the complexity of the involved formulas;
- *complexity in terms of human usability/readability for experimentation* (the formulas to be used to encode the world knowledge and to satisfy the pre-conditions in the axiomatic semantics involve complex nesting of beliefs and desires that might be hard to grasp for a human user of the protocol, and moreover as already mentioned the formulas have to involve large numbers of brackets to help the parser).

An additional challenge in this kind of work is that in the original paper there are extra-logical aspects that have to be clarified in order to implement the protocol. As it is natural with implementations, there are differences between the paper and the code, and during the implementation things can be simplified or might need to be made more explicit. In our case, for example, we simplified the post-conditions of `question` and `challenge`, and we modelled the *dialectical obligations store DOS* in a slightly simpler way compared to the original paper. At the same time, we had to make (temporary) choices about how to model the `Done` operator and on where to put the existential quantifier in the pre-conditions of `question` and `challenge`. These are cases where the experimental approach proposed with LogiKEy can be impactful, as we made these choice with the support of small experiments.

The point of how to assess the correctness of our method is important to discuss. A pen and paper proof would be possible if we had a proper target semantics, but in the process of implementing the protocol we actually had to make changes to the original semantics. Therefore, a contribution of this paper is in making more clear how the target semantics should look like, so to possibly simplify it. In general, an overall future goal is to simplify the modelling to solve some of the challenges listed above. For instance, as already stated, the relationship between the shallow and the deep embedding is not straightforward. Therefore, streamlining it (e.g., by only using a shallow embedding, even though a more sophisticated one) would help to fix some of the complexities we have with the integration of the different reasoning mechanisms.

Acknowledgments

This work was supported by the Luxembourg National Research Fund (FNR) through the project Logical methods for Deontic Explanations (INTER/DFG/23/17415164/LODEX) and the project Deontic Logic for Epistemic Rights (OPEN O20/14776480).
The authors would like to thank Prof. Leon van der Torre for sharing the enthusiasm about Fatio, and David Fuenmayor and Daniel Kirchner for helpful comments.

References

[1] C. Benzmüller, X. Parent, L. van der Torre, Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support, Artificial Intelligence 287 (2020) 103348. doi:10.1016/j.artint.2020.103348.

[2] E. Black, N. Maudet, S. Parsons, Argumentation-based dialogue, Handbook of Formal Argumentation, Volume 2 (2021).

[3] C. Benzmüller, D. Fuenmayor, B. Lomfeld, Modelling value-oriented legal reasoning in LogiKEy, Logics 2 (2024) 31–78. doi:10.3390/logics2010003, preprint arXiv:2006.12789.

[4] X. Parent, C. Benzmüller, Automated verification of deontic correspondences in Isabelle/HOL – first results, in: C. Benzmüller, J. Otten (Eds.), ARQNL 2022: Automated Reasoning in Quantified Non-Classical Logics. Proceedings of the 4th International Workshop on Automated Reasoning in Quantified Non-Classical Logics (ARQNL 2022) affiliated with the 11th International Joint Conference on Automated Reasoning (IJCAR 2022). Haifa, Israel, August 11, 2022, volume 3326, CEUR Workshop Proceedings, CEUR-WS.org, 2023, pp. 92–108. URL: https://ceur-ws.org/Vol-3326/.

[5] C. Benzmüller, Universal (meta-)logical reasoning: Recent successes, Science of Computer Programming 172 (2019) 48–62. doi:10.1016/j.scico.2018.10.008.

[6] P. McBurney, S. Parsons, Locutions for argumentation in agent interaction protocols, in: R. M. van Eijk, M.-P. Huget, F. Dignum (Eds.), Agent Communication, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 209–225.

[7] Fipa. communicative act library specification, Standard SC00037J, 2002. Foundation for Intelligent Physical Agents.

[8] P. McBurney, S. Parsons, Games that agents play: A formal framework for dialogues between autonomous agents, J. Log. Lang. Inf. 11 (2002) 315–334. URL: https://doi.org/10.1023/A:1015586128739. doi:10.1023/A:1015586128739.

[9] T. Nipkow, L. C. Paulson, M. Wenzel, Isabelle/HOL - A Proof Assistant for Higher-Order Logic, volume 2283 of *Lecture Notes in Computer Science*, Springer, 2002.