

On Mixed Semantics of Path Description Dependencies in FunDL

Eva Feng¹, David Toman² and Grant Weddell²

¹Department of Computer Science, Oxford University, Oxford, UK

²Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Abstract

The FunDL family of description logics replace roles with partial functions (*features*) and have a concept constructor called a *path functional dependency* (PFD) that can be used to capture a variety of equality-generating dependencies commonly part of conceptual designs as well as schemata of object-relational data sources. Recent work has considered replacing the PFD concept constructor with a more general *path description dependency* (PDD) in which inverse features are now allowed in characterizing feature path reachability in interpretations. This leads to a circumstance in which the value of a feature path for a given entity might be set-valued. This work has focused on cases in which feature path agreement is based exclusively on either a set intersection semantics or a non-empty set equality semantics. In this paper, we consider a mixed mode case for PDDs in which individual component feature paths can be assigned either of these options. Our main results are that this flexibility makes logical consequence undecidable in general, but that restricting an arbitrary mixed-mode for PDDs to conform to a mode-typing assignment on features re-obtains EXPTIME completeness for logical consequence.


1. Introduction

Given a query over a conceptual ontological design for some domain, there are many circumstances in which reasoning about equality generating dependencies is essential in finding an efficient plan over available backend structured data sources such as relational databases, e.g., in resolving identity issues for entities in an underlying domain [1, 2], or in determining when explicit duplicate elimination in query plans is not required [3]. The FunDL family of description logics [4] has been developed largely for this purpose and therefore replace roles with partial functions (*features*) to better align with the ubiquitous notion of an attribute or column value and have a concept constructor called a *path functional dependency* (PFD) that can be used to capture a variety of equality-generating dependencies needed to resolve identity issues or are commonly part of schemata for structured data sources such as keys and (relational) functional dependencies.

Recent work [5] has introduced a new dialect for FunDL called *set-DLFDI* that replaces the PFD concept constructor with a more general *path description dependency* (PDD) in which inverse features are now allowed in characterizing feature path reachability in interpretations, leading to a circumstance in which the value of a feature path for a given entity might be set-valued. This work has explored alternative semantics for PDDs, with a focus on cases in which feature path agreement is based exclusively on either a *set intersection* semantics or a *non-empty set equality* semantics, showing in both cases that allowing more general PDDs in place of PFDs does not change the complexity of logical consequence in any of the Boolean complete FunDL dialects, which remain EXPTIME complete in the worst case.

The primary incentive for *set-DLFDI* stems from an outline of future work in [6] which recognized the need for plural entities in formally capturing JSON arrays, in particular, for the more general expressiveness of PDDs to capture how such array entities can be identified.

The following inclusion dependencies are derived from a running example in [5] and illustrate this

 DL 2024: 37th International Workshop on Description Logics, June 18–21, 2024, Bergen, Norway

 eva.feng@cs.ox.ac.uk (E. Feng); david@uwaterloo.ca (D. Toman); gweddell@uwaterloo.ca (G. Weddell)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

(where also CLIENT-FRIEND \sqsubseteq CLIENT is assumed):

$$\begin{aligned} \text{CLIENT-FRIEND} \sqsubseteq \text{CLIENT} &: \text{firstname}^\cap, (\text{phone-dom}^- . \text{phone-ran.dialnum})^\cap \rightarrow \text{id}^\cap \\ \text{CLIENT-FRIEND} \sqsubseteq \text{CLIENT} &: (\text{phone-dom}^- . \text{phone-ran.dialnum})^\approx \rightarrow \text{id}^\approx \end{aligned}$$

Feature path reachability in the dependencies corresponds to four *path descriptions* (PDs): “*firstname*”, “*phone-dom⁻.phone-ran.dialnum*” and “*id*”. The first and last are PDs that also qualify as *path functions* (PFs) since their interpretations will be partial functions. The first dependency employs a PDD on its right-hand-side with set intersection semantics (indicated by the “ \cap ” superscript on PDs) to express a key or uniqueness condition for any client that is also a friend: *among all clients, they will have a unique combination of a first name and the dial number of any of their phones*. The second employs a PDD on its right-hand-side with non-empty set equality semantics (indicated by the “ \approx ” superscript on PDs) to also express a key or uniqueness condition: *among all clients with at least one phone, they have a unique set of dial numbers*.

Including both dependencies in a TBox has so far not been possible since this requires two distinct modes for the PD “*phone-dom⁻.phone-ran.dialnum*” in a PDD that appeal to both set intersection and non-empty set equality semantics for PD agreement, a circumstance now enabled by introducing *annotated* PDs in which a superscript arbitrates between a choice of semantics. Indeed, our primary concern in this paper is to study logical consequence for a new member of the FunDL family called *set-DLFDL* in which PDDs now allow annotated PDs as components. Our main results are that, unlike earlier work, this flexibility makes logical consequence undecidable in general, but that restricting an arbitrary mixed-mode for PDDs to conform to a mode-typing assignment on features re-obtains EXPTIME completeness for logical consequence for all members of the FunDL family.

The remainder of the paper proceeds as follows. Section 2 introduces the relevant definitions, previous results and open questions regarding *set-DLFDL* that have motivated this paper. Section 3 shows the undecidability of entailment in *set-DLFDL* when a mixed semantics for path agreements is used. Section 3.1 shows how decidability can be regained by imposing a mild typing discipline on path agreements. Section 4 concludes the paper and outlines directions for further investigation.

2. Background and Definitions

In this section we define the description logic *set-DLFDL*. We start with defining *primitive features* and *concepts*, and how they are interpreted:

Definition 1 (Vocabulary of *set-DLFDL*). *Let F and PC be sets of feature names and primitive concept names, respectively. Semantics is defined with respect to a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a domain of objects or entities and $\cdot^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concept names $A \in \text{PC}$ to be subsets of $\Delta^{\mathcal{I}}$ and feature names $f \in \text{F}$ to be partial functions $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{I}}$.*

The primitive syntax and semantics is now extended to *Path Descriptions* and (eventually) to *Concept Descriptions*:

Definition 2 (Path Descriptions in *set-DLFDL*). *A path description is defined by the grammar*

$$\text{Pd} ::= \text{id} \mid f . \text{Pd} \mid f^- . \text{Pd},$$

for $f \in \text{F}$, where f^- is called the inverse of f , with the stipulation that substrings of the form $f . f^-$ and $f^- . f$ do not appear in any path description Pd. Let \mathcal{I} be an interpretation. Then the interpretations of path descriptions Pd are functions $\text{Pd}^{\mathcal{I}} : 2^{\Delta^{\mathcal{I}}} \rightarrow 2^{\Delta^{\mathcal{I}}}$ over the powerset of $\Delta^{\mathcal{I}}$ defined as follows, where $S \subseteq \Delta^{\mathcal{I}}$:

$$\text{Pd}^{\mathcal{I}}(S) = \begin{cases} S & \text{if Pd} = \text{“id”}, \\ \text{Pd}_1^{\mathcal{I}}(\{f^{\mathcal{I}}(x) \mid x \in S\}) & \text{if Pd} = \text{“f.Pd}_1\text{”}, \\ \text{Pd}_1^{\mathcal{I}}(\{x \mid f^{\mathcal{I}}(x) \in S\}) & \text{if Pd} = \text{“f}^- . \text{Pd}_1\text{”}. \end{cases}$$

SYNTAX	SEMANTICS: DEFN OF $(\cdot)^{\mathcal{I}}$	
$C ::= A$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	(primitive concept; $A \in \text{PC}$)
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	(negation)
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(conjunction)
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(disjunction)
$\forall f.C$	$\{x \mid f^{\mathcal{I}}(x) \in C^{\mathcal{I}}\}$	(value restriction)
$\exists f$	$\{x \mid \exists y \in \Delta^{\mathcal{I}}.y = f^{\mathcal{I}}(x)\}$	(existential restriction)
$\exists f^-$	$\{x \mid \exists y \in \Delta^{\mathcal{I}}.x = f^{\mathcal{I}}(y)\}$	(existential restriction)
$C : \text{Pd}_1^{\sim^1}, \dots, \text{Pd}_k^{\sim^k} \rightarrow \text{Pd}^{\sim}$	$\{x \mid \forall y \in C^{\mathcal{I}} : (\text{GD}(\text{Pd}(x)) \wedge \text{GD}(\text{Pd}(y)) \wedge \bigwedge_{i=1}^k \text{Pd}_i(x) \sim_i \text{Pd}_i(y)) \rightarrow \text{Pd}(x) \sim \text{Pd}(y)\}$	(PDD)

Figure 1: SYNTAX AND SEMANTICS OF CONCEPT DESCRIPTIONS.

Path Descriptions that do not contain inverse features are called *path functions*. Before we define concept descriptions of *set-DLFDL*, we need the notion of *path description agreement*; this agreement is parameterized by how the results of *applying* a path description on a pair of objects are compared. We define and explore three possibilities here, the *set equality*, the *non-empty set equality*, and the *set intersection based agreements*. Formally:

Definition 3 (Path Description Agreement). *Let \mathcal{I} be an interpretation and o_1 and o_2 be two $\Delta^{\mathcal{I}}$ elements. We say that o_1 and o_2 \sim -agree on Pd , written $\text{Pd}(o_1) \sim \text{Pd}(o_2)$, if*

- $\text{Pd}^{\mathcal{I}}(\{o_1\}) = \text{Pd}^{\mathcal{I}}(\{o_2\})$ (*set equality*), when \sim is “=”,
- $\text{Pd}^{\mathcal{I}}(\{o_1\}) = \text{Pd}^{\mathcal{I}}(\{o_2\}) \neq \emptyset$ (*non-empty set equality*), when \sim is “ \approx ”, and
- $\text{Pd}^{\mathcal{I}}(\{o_1\}) \cap \text{Pd}^{\mathcal{I}}(\{o_2\}) \neq \emptyset$ (*set intersection*), when \sim is “ \cap ”.

We use the Path Description Agreements (defined above) to define the logic *set-DLFDL* as follows:

Definition 4 (Concept Descriptions, Subsumptions, and TBoxes in *set-DLFDL*). *A concept description C is constructed from primitive concepts using Boolean concept constructors \sqcap, \sqcup , and \neg , value restrictions on features $\forall f.C$, unqualified existential restrictions on features and inverse features $\exists f$ and $\exists f^-$, and the path description dependency (PDD) of the form*

$$C : \text{Pd}_1^{\sim^1}, \dots, \text{Pd}_k^{\sim^k} \rightarrow \text{Pd}^{\sim}.$$

The semantics of all the derived concept descriptions C is defined in Figure 1 where $\text{GD}(\text{Pd}(z))$ is

- true (*unconstrained*), when \sim is “=” , and
- $\text{Pd}^{\mathcal{I}}(z) \neq \emptyset$ (*non-empty*), when \sim is “ \approx ” or “ \cap ”.

A subsumption is an expression of the form $C_1 \sqsubseteq C_2$, where the C_i are concepts, and where PDDs occur only in C_2 but not within the scope of negation.¹ A terminology (TBox) \mathcal{T} consists of a finite set of subsumptions, and a posed question \mathcal{Q} is a single subsumption. An interpretation \mathcal{I} satisfies a subsumption $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ and is a model of \mathcal{T} , written $\mathcal{I} \models \mathcal{T}$, if it satisfies all subsumptions in \mathcal{T} . Given a terminology \mathcal{T} and posed question \mathcal{Q} , the logical consequence problem asks if \mathcal{Q} is satisfied in all models of \mathcal{T} , written $\mathcal{T} \models \mathcal{Q}$. \square

Observe that the proposed logic lacks *qualified* existential restrictions. Indeed, entailment for *partial-DLFDL* with qualified existential restrictions over inverse features was shown in [9] to be

¹Violating this latter condition leads immediately to undecidability [7, 8].

undecidable, which will therefore also be the case with *set-DLFDL*. However, for a feature f , one can substitute the subsumption $A \sqsubseteq \exists f.B$ with two subsumptions $A \sqsubseteq \exists f$ and $\forall f.B \sqsubseteq A$, which completely characterizes the behaviour of the qualified existential restriction. Note that an analogous substitution for $A \sqsubseteq \exists f^{-}.B$, namely $A \sqsubseteq \exists f^{-}$ and $\forall f.A \sqsubseteq B$, will only partially capture the behaviour of an existential restriction for inverse features, a limitation needed to regain decidability of entailment. In the remainder of the paper, we allow using $\exists \text{Pd}.C$ to serve as a shorthand for applying the above substitutions systematically on Pd by splitting the Pd to individual features and introducing auxiliary primitive concepts.

On Guards in PDDs

One may wonder if the additional guards $\text{GD}(\text{Pd}(x))$ and $\text{GD}(\text{Pd}(y))$ imposed on the consequent path descriptions in a PDD are needed with the $\{\cap, \approx\}$ adornments. Consider the following pair of subsumptions, for each $0 < i \leq k, k \geq 2$:

$$A \sqsubseteq A : (id)^{\cap} \rightarrow (f^{-}.g_i)^{\cap} \text{ and } \forall g_i.\top \sqsubseteq B_i.$$

It is easy to see that these pairs of subsumptions entail $A \sqsubseteq \exists f^{-}.B_i$. In the absence of the guards in the PDD semantics one can add subsumptions $B_i \cap B_j \sqsubseteq \perp$ for $i \neq j$ that will force every A individual to have a distinct f predecessor for each B_i . This leads immediately to undecidability without guards since it allows one to simulate qualified existential restrictions over inverse features [9]. Note that including the guards with a PDD ensures that a right-hand-side will not “force” path existence.

Past Work

The above definition of PDDs can be specialized in several ways, some of which have been considered in the past.

1. In [5] we have considered the *set intersection semantics* for *set-DLFDL*, a semantics in which the \sim agreements in all PDDs were defined as “ \cap ” (non-empty set intersection). However, we have required that all the path descriptions involved in any PDD must *exist* before the PDD applies, i.e., the semantics was defined as

$$(C : \text{Pd}_1, \dots, \text{Pd}_k \rightarrow \text{Pd})^{\mathcal{I}} = \{x \mid \forall y \in C^{\mathcal{I}} : \text{Pd}^{\mathcal{I}}(\{x\}) \neq \emptyset \wedge \text{Pd}^{\mathcal{I}}(\{y\}) \neq \emptyset \wedge (\bigwedge_{i=1}^k \text{Pd}_i^{\mathcal{I}}(\{x\}) \cap \text{Pd}_i^{\mathcal{I}}(\{y\}) \neq \emptyset) \rightarrow \text{Pd}^{\mathcal{I}}(\{x\}) \cap \text{Pd}^{\mathcal{I}}(\{y\}) \neq \emptyset\}.$$

Under the set intersection semantics, the logical consequence problem for *partial-DLFDL*, and consequently its derivative *set-DLFDL*, is *EXPTIME-complete*. We have shown that the problem is in EXPTIME by constructing a two-tree model and then reducing to the satisfiability of an Ackermann formula encoding the logical consequence problem. Completeness then follows from EXPTIME-hardness of the implication problem for the $\{D_1 \cap D_2, \forall f.D_1\}$ and $\{\top, \top : \text{Pf}_1, \text{Pf}_2 \rightarrow \text{Pf}\}$ fragments of FunDL [10].

2. Also in [5], we have considered the *set equality semantics* for *set-DLFDL*, a semantics in which the \sim agreements in all PDDs were defined as “ $=$ ”,

$$(C : \text{Pd}_1, \dots, \text{Pd}_k \rightarrow \text{Pd})^{\mathcal{I}} = \{x \mid \forall y \in C^{\mathcal{I}} : (\bigwedge_{i=1}^k \text{Pd}_i^{\mathcal{I}}(\{x\}) = \text{Pd}_i^{\mathcal{I}}(\{y\})) \rightarrow \text{Pd}^{\mathcal{I}}(\{x\}) = \text{Pd}^{\mathcal{I}}(\{y\})\}.$$

We have shown that under the set equality semantics, the ability to assert equality of non-existent paths in a PDD’s precondition allows us to create nominal-like concepts which leads immediately to *undecidability*.

Set-Intersection vs. Set-Equality	Empty Set	Type	Complexity
Set-Intersection	No (by definition)	–	EXPTIME-complete [5]
Set-Equality	Yes	–	Undecidable [5]
Set-Equality	No	–	EXPTIME-complete (new)
Both	No for Set-Equality	Mixed	Undecidable (new)
Both	No for Set-Equality	Typed	EXPTIME-complete (new)

Figure 2: Complexity of *set-DLFDI* Based on Various Semantics

3. To regain decidability under the set equality semantics, we preclude the equality of empty sets by a *non-empty set equality semantics* defined as follows,

$$(C : \text{Pd}_1, \dots, \text{Pd}_k \rightarrow \text{Pd})^{\mathcal{I}} = \{x \mid \forall y \in C^{\mathcal{I}} : \text{Pd}^{\mathcal{I}}(\{x\}) \neq \emptyset \wedge \text{Pd}^{\mathcal{I}}(\{y\}) \neq \emptyset \wedge (\bigwedge_{i=1}^k \text{Pd}_i^{\mathcal{I}}(\{x\}) = \text{Pd}_i^{\mathcal{I}}(\{y\}) \neq \emptyset) \rightarrow \text{Pd}^{\mathcal{I}}(\{x\}) = \text{Pd}^{\mathcal{I}}(\{y\})\}.$$

which brings the complexity back to EXPTIME-complete. The proof for *non-empty set equality semantics* is essentially the same as the one for *set intersection semantics* [5].

In this paper we consider the remaining possibilities (some of which were posed as open problems).

4. We have conjectured that mixing the set intersection and non-empty set equality semantics, both of which are decidable, could lead to undecidability. In this paper, we show that allowing combinations of the EXPTIME-complete semantics in the PDs leads to undecidability by reduction of the unconstrained tiling problem. Note that for path functions there is no difference between the set intersection agreements and non-empty set agreements; hence we omit the *type* of the agreement in the remaining constructions for path functions.
5. Curiously, we can restrict arbitrary mixed semantics by virtue of fixing the type of semantics for each feature f and inverse feature f^- , which generalizes to PDs. In Section 3.1, we show that adopting such *typed mixed semantics* allows us to regain EXPTIME-completeness.

We summarize the complexity results for *set-DLFDI* based on various combinations of set intersection and set equality-based semantics in Figure 2.

3. Undecidability of Mixed PDDs

In this section, we show that allowing both non-empty set equality and set intersection semantics simultaneously in the PDDs leads to undecidability by reduction of a tiling problem that allows one to simulate runs of a Turing Machine on an empty input tape, a problem that is known to be undecidable [11]. An instance U of the tiling problem is a triple (T, H, V) , consisting of a set T of tile types and $H, V \subseteq T \times T$ two binary relations. A solution to U is a function $t : \mathbb{N} \times \mathbb{N} \rightarrow T$ such that, for $i < j$, we have $(t(i, j), t(i + 1, j)) \in H$ and $(t(i, j), t(i, j + 1)) \in V$. This tiling solution covers an infinite *triangle* of successive longer and longer *tapes* (instantaneous descriptions) and this way simulates a run of a given Turing Machine². We construct a terminology along with a posed question, denoted $\mathcal{T}(T, H, V)$, for a given tiling problem U , in the following steps.

Theorem 5. *An instance (T, H, V) of the unconstrained tiling problem admits a solution if and only if $\mathcal{T}(T, H, V) \not\models A \sqsubseteq D : f \rightarrow id$ for a $\mathcal{T}(T, H, V)$ TBox corresponding to the instance (T, H, V) .*

Proof (sketch): Given a tiling problem (T, H, V) we construct a TBox $\mathcal{T}(T, H, V)$ that utilizes pairwise disjoint primitive concepts A, B , and C to serve as *grid points* and an auxiliary *diagonal* primitive concept D . We assume that the features f, g, h , and k are total, and the inverse f^- is total on A, B, C , and D . These are captured by $\mathcal{T}(T, H, V)$ subsumptions

²The reduction is essentially the same as for the classical tiling of a full quadrant, but starts with an empty tape.

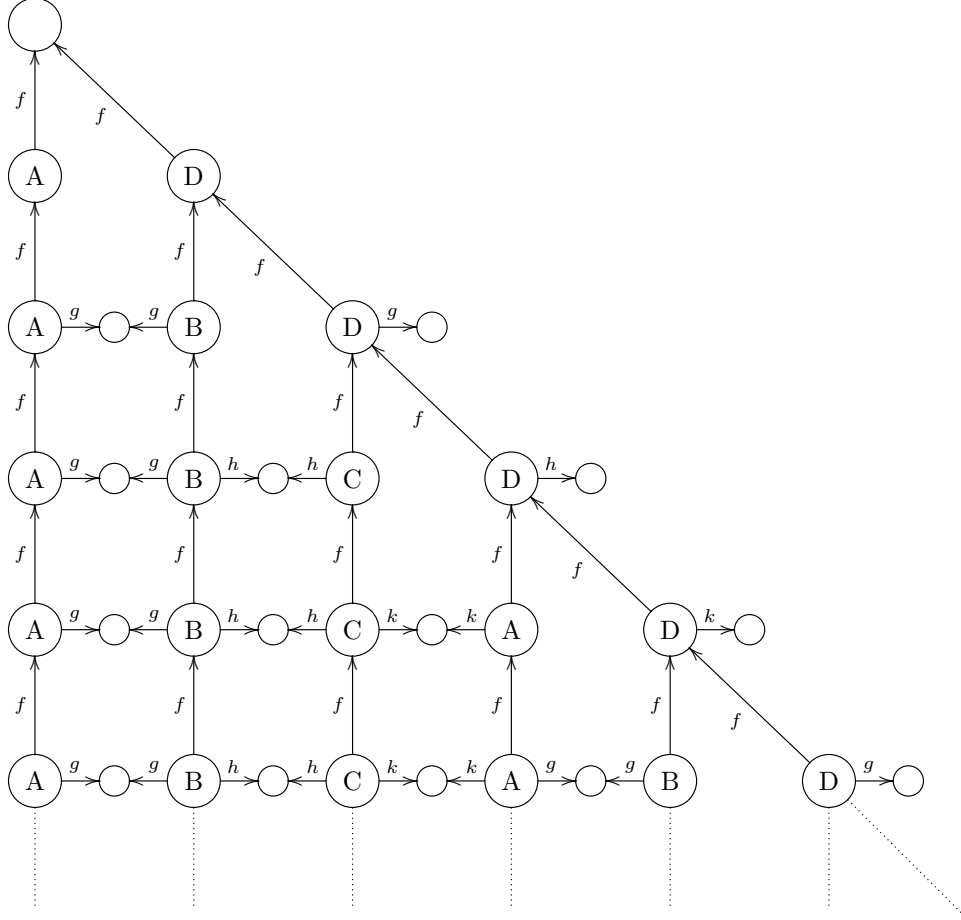


Figure 3: Construction of a Grid for Tiling.

$$\begin{aligned}
& \top \sqsubseteq \exists f \sqcap \exists g \sqcap \exists h \sqcap \exists k \\
& A \sqcup B \sqcup C \sqcup D \sqsubseteq \exists f^-; \text{ and} \\
& A \sqcap B \sqsubseteq \perp, A \sqcap C \sqsubseteq \perp, A \sqcap D \sqsubseteq \perp, B \sqcap C \sqsubseteq \perp, B \sqcap D \sqsubseteq \perp, C \sqcap D \sqsubseteq \perp.
\end{aligned} \tag{1}$$

To show the non-entailment we need to construct a counterexample to the posed question, $A \sqsubseteq D : f \rightarrow id$ that satisfies all subsumptions in $\mathcal{T}(T, H, V)$. We construct the TBox $\mathcal{T}(T, H, V)$ for an instance of a tiling problem (T, H, V) in such a way that the counterexample will correspond to a solution to this tiling problem. In what follows we list all the subsumptions needed in $\mathcal{T}(T, H, V)$ to achieve this goal. The subsumption

$$A \sqsubseteq D : f \rightarrow (f^- . g)^\cap \quad (\text{and } B \sqsubseteq D : f \rightarrow (f^- . h)^\cap, C \sqsubseteq D : f \rightarrow (f^- . k)^\cap), \tag{2}$$

stating that A and D that agree on f must \cap -agree on $f^- . g$ (and that B and D that agree on f must \cap -agree on $f^- . h$ and C and D that agree on f must \cap -agree on $f^- . k$, respectively), the subsumption

$$A \sqsubseteq D : (f^- . g)^\approx \rightarrow id \quad (\text{and } B \sqsubseteq D : (f^- . h)^\approx \rightarrow id, C \sqsubseteq D : (f^- . k)^\approx \rightarrow id), \tag{3}$$

stating that A and D above must \approx -disagree on those paths, in order not to equate an A object with an D object (and similarly for B and D, and C and D, respectively), together with subsumption extending the A (and similarly B and C, respectively) class memberships along f predecessors

$$\forall f . A \sqsubseteq A \quad (\text{and } \forall f . B \sqsubseteq B, \forall f . C \sqsubseteq C) \tag{4}$$

and subsumptions ensuring those predecessors—with respect to a particular class membership—are unique,

$$\begin{aligned} D \sqsubseteq D : f \rightarrow id, \\ A \sqsubseteq A : f \rightarrow id, \quad (\text{and } B \sqsubseteq B : f \rightarrow id, C \sqsubseteq C : f \rightarrow id), \end{aligned} \quad (5)$$

force this counterexample to look like the first three top rows in Figure 3 (with the exception of class membership of the final two objects in the third row). Note that the A object's f predecessors are also A objects due to (4). Moreover the A predecessors are unique due to (5). Hence the D object at the end of row 2 must have at least two incoming f features to satisfy (2) and (3) simultaneously while not violating the disjointness of A and D in (1). The same holds for the right-most B and C objects used in the construction of the subsequent rows.

The now existing horizontal *right neighbours* of the object A in the 3rd row are assigned concept membership using the following subsumptions:

$$\begin{aligned} A \sqsubseteq \neg B : g \rightarrow id \quad (\text{and } B \sqsubseteq \neg C : h \rightarrow id, C \sqsubseteq \neg A : k \rightarrow id); \\ B \sqsubseteq \neg D : f \rightarrow id \quad (\text{and } C \sqsubseteq \neg D : f \rightarrow id, A \sqsubseteq \neg D : f \rightarrow id), \end{aligned} \quad (6)$$

i.e., the right ($g.g^-$) neighbour of A must be B, etc. In particular, the object D at the end of row 2 must have exactly two incoming f features, one from an B and second from a D objects. This completes the construction of the top three rows in Figure 3. To replicate this process for row 4 in Figure 3 we use the second subsumptions in (2), (3), (4), (5), (6), and (7).

To complete the left part(s) of the grid—the squares below A, B, and C objects we use the following subsumptions:

$$A \sqsubseteq B : f.g \rightarrow g \quad (\text{and } B \sqsubseteq C : f.h \rightarrow h, C \sqsubseteq A : f.k \rightarrow k); \quad (7)$$

Analogously, the 3rd subsumptions in (2), (3), (4), (5), (6), and (7) will construct row 5. The end of row 5 presents a situation that is a copy of the pattern in row 2 and the hence the construction starts repeating itself indefinitely and extends to an infinite triangle by simply repeating the above steps.

To finish the construction we need to assign tiles to all grid points,

$$A \sqcup B \sqcup C \sqsubseteq \sqcup_{t_i \in T} T_i, \quad (8)$$

and enforce the horizontal,

$$\begin{aligned} T_i \sqcap A \sqsubseteq (T_j \sqcap B) : g \rightarrow id, \\ T_i \sqcap B \sqsubseteq (T_j \sqcap C) : h \rightarrow id, \\ T_i \sqcap C \sqsubseteq (T_j \sqcap A) : k \rightarrow id, \quad \forall (t_i, t_j) \notin H, \end{aligned} \quad (9)$$

and vertical tiling rules,

$$\begin{aligned} \forall f. (T_i \sqcap A) \sqcap T_j \sqsubseteq \perp, \\ \forall f. (T_i \sqcap B) \sqcap T_j \sqsubseteq \perp, \\ \forall f. (T_i \sqcap C) \sqcap T_j \sqsubseteq \perp, \quad \forall (t_i, t_j) \notin V. \end{aligned} \quad (10)$$

In summary, a TBox $\mathcal{T}(T, H, V)$ containing all the subsumptions (1), (2), (3), (4), (5), (6), (7), (8), (9), and (10) does *not entail* the posed question if and only if a tiling exists.

The existence of a non-terminating computation of a given Turing machine starting with an empty tape can now be witnessed by the existence of a tiling using the standard encoding of instantaneous descriptions of the TM's computations as rows in the tiling overlaid over Figure 3; the computation steps then correspond to the consecutive rows of tiles. Note that tiling of the infinite triangle is sufficient as the TM's head can move at most one cell to the right for every step of the computation. \square

Alternatively, to *tile* a full quadrant, we can use the vertical f -antichains as columns (as above) and *diagonals* starting from the left-most A-labelled column as *rows*. This needs a slight adjustment to the horizontal tiling subsumptions (9) as follows:

$$\begin{aligned} \forall f. (T_i \sqcap A) \sqsubseteq (T_j \sqcap B) & : g \rightarrow id, \\ \forall f. (T_i \sqcap B) \sqsubseteq (T_j \sqcap C) & : h \rightarrow id, \\ \forall f. (T_i \sqcap C) \sqsubseteq (T_j \sqcap A) & : k \rightarrow id, \quad \forall (t_i, t_j) \notin H, \end{aligned} \tag{9'}$$

However, now the *standard* tiling argument applies and also yields undecidability.

3.1. Typed Mixed Case

We observe from the previous section that inconsistent semantics for two instances of the same PD leads to undecidability. In this section, we show that under a restricted form of the mixed semantics, called the *typed mixed semantics*, *set-DLFDL* regains EXPTIME-completeness. The idea of enforcing a *type restriction* is to make the semantics of path agreements consistent among PDs ending with the same (inverse) feature. This type restriction preserves the two-tree property of *set-DLFDL* under the set intersection and non-empty set equality semantics.

Definition 6 (Type Restriction). *Let \mathcal{T} be a set-DLFDL TBox and \mathcal{Q} a posed question. Let $\text{end}(\text{Pd})$ denote the last (inverse) feature of Pd. We say that \mathcal{T} and \mathcal{Q} are type restricted if for any pair of PDs $\text{Pd}_1^{\sim 1}$ and $\text{Pd}_2^{\sim 2}$ in $\mathcal{T} \cup \mathcal{Q}$, we have $\sim_1, \sim_2 \in \{\approx, \sqcap\}$ and*

$$\text{end}(\text{Pd}_1^{\sim 1}) = \text{end}(\text{Pd}_2^{\sim 2}) \text{ implies } \sim_1 = \sim_2 .$$

The typed mixed semantics immediately follow from the syntactic type restriction on all PDs in the TBox and posed question.

Theorem 7. *Let \mathcal{T} be a set-DLFDL TBox and \mathcal{Q} a posed question, where $\mathcal{T} \cup \mathcal{Q}$ are type restricted. Then the logical consequence problem $\mathcal{T} \models \mathcal{Q}$ is complete for EXPTIME.*

Proof (sketch): To establish the EXPTIME bound, we reduce the logical consequence problem to checking the unsatisfiability of an Ackermann formula [12] in a similar fashion as in [5]. First, we show that if there exists a counterexample \mathcal{I} for $\mathcal{T} \models \mathcal{Q}$, then we can construct a two-tree counterexample by unravelling \mathcal{I} . Note that under the imposed type discipline, there is only *one form of path agreement* applicable to any symmetric pair of individuals in the two-tree unravelling of \mathcal{I} and therefore there is no longer any need feature agreements that go beyond the *symmetric* agreements generated by PDDs. Since we can check the (un)satisfiability of the constructed Ackermann formula in EXPTIME [13], the same bound applies to the logical consequence problem for *set-DLFDL*. \square

4. Summary

The paper explores the computational complexity of logical consequence for a new member of the FunDL family of descriptions logics called *set-DLFDL*. This new dialect introduces a PDD concept constructor for capturing a richer variety of equality generating dependencies under arbitrary combinations of set intersection semantics and non-empty set equality semantics for path agreements in component path descriptions. In particular, the paper shows that an unconstrained option for choosing either semantics makes logical consequence undecidable. This is in contrast to the two cases where one or the other semantics is chosen exclusively for a given TBox which has been shown decidable in earlier work. The paper then proposes a typing discipline on path agreements to regain decidability, thereby accommodating some form of mixed-mode semantics for PDDs occurring in a given TBox.

Future Work

There are four main directions for further inquiry:

1. Adding arbitrary PDL-style test concepts [14] in path descriptions. In the case of PDDs this means adding concepts of the form $(C_1, C_2)?$, stating that for a path agreement to hold, the paths from the two objects, x and y , in the PDD constructor must simultaneously pass through an individual that belongs to C_1 and C_2 , respectively. We conjecture that adding such test concepts to path descriptions in *set-DLFDL* will not change the computational properties of the logic.
2. Studying additional or alternative notions of *path agreement* in *set-DLFDL*. For example, the *set intersection* semantics of path agreements can be generalized by requiring the intersection to be of a certain cardinality (e.g., *at least two* or a *majority* of paths agree). We conjecture that such extensions will again lead to undecidability of entailment.
3. Studying *set-DLFDL* fragments with preferable computational properties, such as various Horn fragments. This direction suggests the use of (perhaps limited) PDDs in FunDL-Lite logics [4].
4. Using PDD constructs in *plural entity identification*. While our introductory example has already sketched such a use case, the full consequences of using *path descriptions* in place of *path functions* are yet to be explored.

References

- [1] A. Borgida, D. Toman, G. Weddell, On referring expressions in query answering over first order knowledge bases, in: Proc. Principles of Knowledge Representation and Reasoning, KR 2016, 2016, pp. 319–328.
- [2] A. Borgida, D. Toman, G. E. Weddell, On referring expressions in information systems derived from conceptual modelling, in: I. Comyn-Wattiau, K. Tanaka, I. Song, S. Yamamoto, M. Saeki (Eds.), Conceptual Modeling - 35th International Conference, ER 2016, volume 9974 of *Lecture Notes in Computer Science*, 2016, pp. 183–197.
- [3] D. Toman, G. E. Weddell, Using feature-based description logics to avoid duplicate elimination in object-relational query languages, *Künstliche Intell.* 34 (2020) 355–363. URL: <https://doi.org/10.1007/s13218-020-00666-7>. doi:10.1007/s13218-020-00666-7.
- [4] S. McIntyre, D. Toman, G. E. Weddell, FunDL - A family of feature-based description logics, with applications in querying structured data sources, in: Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, 2019, pp. 404–430.
- [5] E. Feng, A. Borgida, E. Franconi, P. F. Patel-Schneider, D. Toman, G. E. Weddell, Path description dependencies in feature-based dls, in: Description Logics, volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023.
- [6] A. Borgida, E. Franconi, D. Toman, G. E. Weddell, Understanding document data sources using ontologies with referring expressions, in: H. Aziz, D. Corrêa, T. French (Eds.), AI 2022: Advances in Artificial Intelligence - 35th Australasian Joint Conference, AI 2022, Perth, WA, Australia, December 5-8, 2022, Proceedings, volume 13728 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 367–380.
- [7] D. Toman, G. Weddell, On Keys and Functional Dependencies as First-Class Citizens in Description Logics, in: Proc. of Int. Joint Conf. on Automated Reasoning (IJCAR), 2006, pp. 647–661.
- [8] D. Toman, G. E. Weddell, On keys and functional dependencies as first-class citizens in description logics, *J. Aut. Reasoning* 40 (2008) 117–132.
- [9] D. Toman, G. E. Weddell, On the interaction between inverse features and path-functional dependencies in description logics, in: Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 2005, pp. 603–608.

- [10] D. Toman, G. Weddell, On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem, in: Description Logics 2001, CEUR-WS vol.49, 2001, pp. 76–85.
- [11] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.
- [12] W. Ackermann, Über die Erfüllbarkeit gewisser Zahlausdrucke, Mathematische Annalen 100 (1928) 638–649.
- [13] M. Fürer, Alternation and the Ackermann Case of the Decision Problem, L’Enseignement Math. 27 (1981) 137–162.
- [14] M. J. Fischer, R. E. Ladner, Propositional dynamic logic of regular programs, J. Comput. Syst. Sci. 18 (1979) 194–211.