Tree Description Dependencies

David Toman[1], Grant Weddell[1]

[1]Cheriton School of Computer Science, University of Waterloo, 200 University Ave W., Waterloo, ON N2L 3G1, Canada

Abstract

We introduce a general variety of *equality-generating dependencies* (EGDs) based on *tree descriptions* for an expressive dialect of the FunDL family of description logics called *tree description dependencies* (TDDs). In general, TDDs enable capturing equality generating dependencies in an ontology. The new capability is tailored to *structurally matching* parts of structured documents/JSON objects. We show that logical entailment for this new FunDL dialect is complete for EXPTIME if a given ontology appeals to an exclusive use of TDDs, but that logical entailment becomes undecidable when more coarse grained varieties of TDDs (called Path Description Dependencies, PDDs) are simultaneously allowed in the ontology. The main application for EGDs derives from the fundamental problem of identification in information systems. We tie this application by appeal to earlier work on developing a domain specific language for referring expressions. An extension to this language for defining concepts to serve as referring expressions that in turn depend on structural equality of tree descriptions is also presented and is tied to a diagnosis of a singularity condition for such concepts to logical entailment of TDDs for an ontology.

1. Introduction

Structured data sources abound, and ontology based data access is all about querying such sources via an ontological understanding of their content. Here, effective ways to communicate answers to queries will depend critically on communicating references to underlying entities via referring expressions, also called definite descriptions [1]. Earlier work has introduced the notion of referring expression types, each of which will define a set of possible referring expressions [2, 3]. That such descriptions achieve unambiguous reference will in turn depend on ontological knowledge of so-called equality generating dependencies, for example, knowing that a person will have a unique social insurance number, or that a room, when non-empty, will have a unique combination of people occupying the room.

In this paper, we introduce a more expressive variety of such dependencies when an ontological understanding is expressed in terms of a description logic, in particular, in terms of an expressive dialect of the FunDL family of description logics [4]. For a better alignment with common data sources such as relational databases, all such logics are feature-based instead of role-based, that is, consider facts to be captured with partial functions instead of more general binary relationships. Such logics have recently included a concept constructor called a *path description dependency* (PDD) in which component path descriptions can be annotated to define progressively richer conditions for equality generation [5, 6, 7]. There were three possible annotations that have been considered. Both relate to the respective non-empty sets of entities reachable by a path description: a "set intersection" annotation that is satisfied when there is at least one such entity in common, a "set equality" annotation that is satisfied when the respective non-empty sets of reachable entities are the same, and a more expressive "structural equality" annotation [8] for path descriptions in which equality only follows according to a structured alignment of non-empty sets of facts about an entity. However, even the most fine-grained variant of PDDs, based on the structural equality, do not satisfactorily describe the structural properties of hierarchical data (such as JSON documents). Consider the two fragments in Figure 1 that capture the information about two persons and their common parent. Trying to specify that such a parent is uniquely identified by the names of their children, one is tempted to use a (structural equality-based [8]) PDD of the form

$$P \sqsubseteq P : parent^{-}.fn^{\simeq}, parent^{-}.ln^{\simeq} \rightarrow id\ .$$

DL 2026: 39th International Workshop on Description Logics, July 17–19, 2026, Lisbon, Portugal

david@uwaterloo.ca (D. Toman); gweddell@uwaterloo.ca (G. Weddell)

Unfortunately, such a constraint forces the persons P_0 and P_1 in the figure to be equal (which is not desirable). The issue stems from the common *parent*$^-$ prefix of both the Pds in the PDD. The

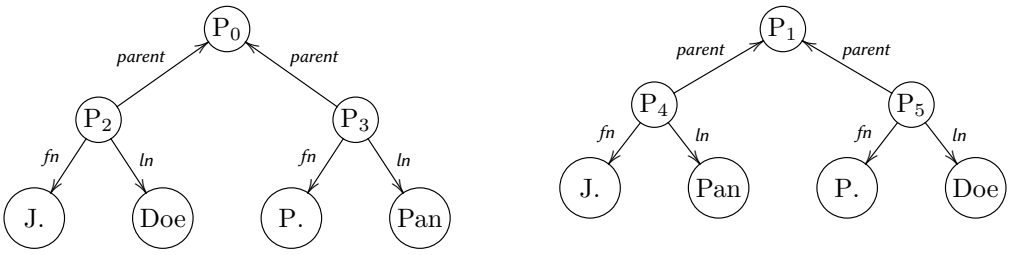

Figure 1: Tree Description-based structural equality.

main contribution of this paper is the introduction of *Tree Description*s (Tds) and the associated Tree Description Dependencies (TDDs) that generalize PDDs and allow us to write

$$P \sqsubseteq P : \mathit{parent}^-.(\mathit{fn}, \mathit{ln}) \to \mathit{id}$$

instead of the PDD above (formal definitions to follow in Section 2). This TDD distinguishes between P_0 and P_1 in our example while correctly capturing the fact that a parent is uniquely identified by the first and last names of their children as it *groups* first and last names for a *particular child*. This new TDD constructor fully generalizes PDDs based on structural equality [8].

In this paper, we show in Section 4 that logical entailment for a very expressive dialect of FunDL is decidable if a given ontology appeals only to this new structural equality based on TDDs. We also show in Section 5 that logical entailment becomes undecidable when either of the more coarse grained set intersection or set equality semantics for PDDs are also allowed together with TDDs. Section 2 provides the necessary background and definitions. Section 3 outlines the application to entity identification in information systems where concept descriptions are viewed as referring expressions [2]. Here, an extension to a domain specific language for defining concepts in this description logic to serve as referring expressions to accommodate structural identification is given and is tied to a diagnosis of a singularity condition for such concepts to logical entailment of TDDs for a given ontology. We conclude with summary comments in Section 6.

2. Background and Definitions

In this section we define $\mathcal{DLTDI}$ that is a new member of the FunDL dialects of description logic [4], beginning with *primitive features* and *concepts*, and how they are interpreted:

Definition 1 (Vocabulary of $\mathcal{DLTDI}$). *Let* PC *and* F *be sets of primitive concept names and feature names, respectively. Semantics is defined with respect to a structure* $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$*, where* $\triangle$ *is a domain of objects or entities and* $\cdot^{\mathcal{I}}$ *an interpretation function that interprets primitive concept names* $A \in \mathsf{PC}$ *as subsets of* $\triangle$ *and feature names* $f \in \mathsf{F}$ *as partial functions* $f^{\mathcal{I}} : \triangle \to \triangle$*.* □

In the rest of the paper we use $f, g, h, \ldots$ to denote features and $A, B, C, D, \ldots$ to denote concepts. One of the main features of the $\mathcal{DLTDI}$ dialect (as we alluded to in the introduction) is the ability to formulate complex *tree description dependency* (TDD) concepts using tree patterns built from features and their inverses called *tree descriptions*. Formally:

Definition 2 (Tree Descriptions). *A* Tree Description *is defined by the grammar*

$$\mathsf{Td} ::= \mathit{id} \mid f.\mathsf{Td} \mid f^-.\mathsf{Td} \mid \mathrm{A}? \mid (\mathsf{Td}, \mathsf{Td}),$$

where $f \in \mathsf{F}$ *and* $\mathrm{A} \in \mathsf{PC}$*. We will write* $(\mathsf{Td}_1, \ldots, \mathsf{Td}_k)$ *for* $((\ldots(\mathsf{Td}_1, \mathsf{Td}_2), \ldots), \mathsf{Td}_k)$ *and require that in the tree description of the form* $f.\mathsf{Td}$ *the tree description* Td *is not of the form* $(\ldots, f^-.\mathsf{Td}, \ldots)$ *and*

$$
\begin{array}{ll}
x = y & \textit{if } \mathsf{Td} = id, \\
\exists x_1, y_1.(x_1 = f^{\mathcal{I}}(x)) \wedge (y_1 = f^{\mathcal{I}}(y)) \rightarrow \mathsf{Td}_1(x_1, y_1) & \textit{if } \mathsf{Td} = f.\,\mathsf{Td}_1, \\
\exists x_1.(f^{\mathcal{I}}(x_1) = x) \wedge \exists y_1.(f^{\mathcal{I}}(y_1) = y) & \\
\quad \wedge \forall x_1.(f^{\mathcal{I}}(x_1) = x) \rightarrow \exists y_1.(f^{\mathcal{I}}(y_1) = y) \wedge \mathsf{Td}_1(x_1, y_1) & \\
\quad \wedge \forall y_1.(f^{\mathcal{I}}(y_1) = y) \rightarrow \exists x_1.(f^{\mathcal{I}}(x_1) = y) \wedge \mathsf{Td}_1(x_1, y_1) & \textit{if } \mathsf{Td} = f^{-}.\,\mathsf{Td}_1, \\
x \in \mathrm{A}^{\mathcal{I}} \wedge y \in \mathrm{A}^{\mathcal{I}} & \textit{if } \mathsf{Td} = \mathrm{A}? \\
\mathsf{Td}_1(x, y) \wedge \mathsf{Td}_2(x, y) & \textit{if } \mathsf{Td} = (\mathsf{Td}_1, \mathsf{Td}_2)
\end{array}
$$

Figure 2: Tree Description-based Structural Equality $\mathsf{Td}(x, y)$.

vice versa, i.e., the paths through a tree description may not contain f followed immediately by f^- nor f^- followed immediately by f. □

Tree descriptions that use the last production only in the form $(\mathrm{A}?, \mathsf{Td})$ coincide with *path descriptions* (PDs) and PDs that do not contain inverse features are called *path functions*. Note that the concept A in the test A? can be generalized to arbitrary concepts (not mentioning TDDs) since such concepts can be made equivalent to a primitive concept in a TBox. In preparation for defining concept descriptions in *set-$\mathcal{DLFDI}$* dialects, we need the notion of *structural path description agreement*:

Definition 3 (Tree Description-based Structural Equality). *Let* Td *be a tree description, $\mathcal{I}$ an interpretation and x and y be two $\triangle$ elements. We say that x and y* structurally agree on Td *(or just* agree on Td*), written* $\mathsf{Td}(x, y)$*, if they obey the definition in Figure 2.*

Definition 4 (Concepts, Subsumptions, and TBoxes).
In $\mathcal{DLTDI}$, a member of the set-$\mathcal{DLFDI}$ family, 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 $\exists f$ and inverse features $\exists f^-$, and the TDD concept constructor of the form*

$$C : \mathsf{Td} \rightarrow \mathsf{Td}'.$$

The semantics of all the derived concept descriptions C is defined in the standard way; for a TDD the semantics is given by:

$$(C : \mathsf{Td} \rightarrow \mathsf{Td}')^{\mathcal{I}} = \{x \mid \forall\, y \in C^{\mathcal{I}} : \mathsf{GD}_{\mathsf{Td}'}(x) \wedge \mathsf{GD}_{\mathsf{Td}'}(y) \wedge (\mathsf{Td}(x, y) \rightarrow \mathsf{Td}'(x, y))\},$$

where, for an individual $x \in \triangle$, $\mathsf{GD}_{\mathsf{Td}}(x)$ is defined as follows:

$$\mathsf{GD}_{\mathsf{Td}}(x) = \begin{cases} \textit{true} & \textit{if } \mathsf{Td} = id, \\ \exists y.y = f^{\mathcal{I}}(x) \wedge \mathsf{GD}_{\mathsf{Td}_1}(y) & \textit{if } \mathsf{Td} = f.\,\mathsf{Td}_1, \\ \exists y.x = f^{\mathcal{I}}(y) \wedge \mathsf{GD}_{\mathsf{Td}_1}(y) & \textit{if } \mathsf{Td} = f^{-}.\,\mathsf{Td}_1, \\ x \in A^{\mathcal{I}} & \textit{if } \mathsf{Td} = \mathrm{A}?, \\ \mathsf{GD}_{\mathsf{Td}_1}(x) \wedge \mathsf{GD}_{\mathsf{Td}_2}(x) & \textit{if } \mathsf{Td} = (\mathsf{Td}_1, \mathsf{Td}_2). \end{cases}$$

A subsumption *is an expression of the form $C_1 \sqsubseteq C_2$, where the C_i are concepts, and where TDDs 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 entailment problem *asks if $\mathcal{Q}$ is satisfied in all models of $\mathcal{T}$, that is, if $\mathcal{Q}$ is* entailed by *$\mathcal{T}$, written $\mathcal{T} \models \mathcal{Q}$.* □

[1] Violating this latter condition leads immediately to undecidability [9, 10].

Note that the *guard* $\mathsf{GD}_{\mathsf{Td}'}$ conditions in the definition of TDD semantics are needed for the consequent in order not to force creation of additional individuals/paths in models of such constraints. This is needed to guarantee decidability by disallowing *agreements* induced by empty sets of f-predecessors [5]: there must be at least one instance of every inverse feature f^- used in a Td, see again Figure 2.

Also note that every PDD of the form $C : \mathsf{Pd}_1^{\simeq}, ..., \mathsf{Pd}_k^{\simeq} \rightarrow \mathsf{Pd}^{\simeq}$ [8] can now be equivalently written as the TDD $C : (\mathsf{Pd}_1, ..., \mathsf{Pd}_k) \rightarrow \mathsf{Pd}$ where all occurrences of $\mathrm{A}?.\,\mathsf{Pd}$ are replaced by $(\mathrm{A}?, \mathsf{Pd})$. Hence the logic $\{\simeq\}$-$\mathcal{DLFDI}$ defined in [8] is a special case of $\mathcal{DLTDI}$.

Observe that the proposed logic lacks *qualified* existential restrictions. Indeed, entailment for an earlier FunDL dialect *partial-*$\mathcal{DLFDI}$ with qualified existential restrictions over inverse features was shown to be undecidable [11], which will therefore also be the case with *set-*$\mathcal{DLFDI}$/$\mathcal{DLTDI}$ dialects. However, for a feature f, one can substitute the subsumption $A \sqsubseteq \exists f.B$ with two subsumptions $A \sqsubseteq \exists f$ and $A \sqsubseteq \forall f.B$, 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 $\exists \mathsf{Pd}.C$ 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. Also observe that disallowing consecutive "$f^-.f$" in a Td is w.l.o.g. since such a sub-path can be replaced by "$(\exists f^-)?$" as f is functional, but that, analogous to the case above, replacing "$f.f^-$" by "$(\exists f)?$" is only an approximation.

3. Plural Referring Expressions and Types

Recall from our introductory comments that referring expressions are concepts induced by a domain specific language for referring expression types. Such types are defined by a pattern language for specifying a set of concepts that can serve as referring expressions in *set-*$\mathcal{DLFDI}$ dialects augmented with an additional concept constructor of the form "$\{a\}$" for nominals; we call a the name of such a nominal. (Note that nominals may not appear in subsumptions.)

Definition 5 (Referring Expression Types). *A* referring expression type *(Rt) is defined by the following grammar, where* A *is a primitive concept.*

$$Rt ::= \{?\} \mid \mathrm{A} \rightarrow Rt \mid \exists f.Rt \mid \exists f^-.Rt \mid Rt_1 \sqcap Rt_2 \mid Rt_1 \,;\, Rt_2$$

The language of referring concepts inhabiting Rt, $\mathcal{L}(Rt)$, is defined as follows:

$$\begin{aligned}
\mathcal{L}(\{?\}) &= \{\{a\} \mid a \text{ is a constant symbol}\} \\
\mathcal{L}(\mathrm{A} \rightarrow Rt) &= \{\mathrm{A} \sqcap C \mid C \in \mathcal{L}(Rt)\} \\
\mathcal{L}(\exists f.Rt) &= \{\exists f.C \mid C \in \mathcal{L}(Rt)\} \\
\mathcal{L}(\exists f^-.Rt) &= \{(\exists f^-.C[\vec{a}/\vec{b}_1]) \sqcap \ldots \sqcap (\exists f^-.C[\vec{a}/\vec{b}_k]) \mid C \in \mathcal{L}(Rt)\} \\
\mathcal{L}(Rt_1 \sqcap Rt_2) &= \{C_1 \sqcap C_2 \mid C_1 \in \mathcal{L}(Rt_1) \wedge C_2 \in \mathcal{L}(Rt_2)\} \\
\mathcal{L}(Rt_1; Rt_2) &= \mathcal{L}(Rt_1) \cup \mathcal{L}(Rt_2)
\end{aligned}$$

where $C[\vec{a}/\vec{b}]$ is the concept C in which all nominal names $\vec{a} = (a_1, \ldots, a_n)$ in C have been replaced by nominal names $\vec{b} = (b_1, \ldots, b_n)$, where n is the number of nominals in C. Observe the above language definition this replacement ranges over all possible distinct choices of tuples of nominal names $\vec{b}_1, \ldots, \vec{b}_k$ for $\vec{b}$ and all $0 < k$. Given a TBox $\mathcal{T}$ and referring expression type Rt, the singularity problem for Rt with respect to $\mathcal{T}$ *is to determine if $|C^{\mathcal{I}}| \leq 1$ for every $C \in \mathcal{L}(Rt)$ and every model $\mathcal{I}$ of $\mathcal{T}$.* □

Definition 6 (Normalized Referring Expression Types). *We use* $\mathsf{Norm}(Rt)$ *to refer to an exhaustive*

application of the following rewrite rules:

$$\begin{aligned}
\mathrm{A} \to (Rt_1; Rt_2) &\mapsto \mathrm{A} \to Rt_1; \mathrm{A} \to Rt_2 \\
Rt \sqcap (Rt_1; Rt_2) &\mapsto Rt \sqcap Rt_1; Rt \sqcap Rt_2 \\
(Rt_1; Rt_2) \sqcap Rt &\mapsto Rt_1 \sqcap Rt \,; Rt_2 \sqcap Rt \\
\exists f.(Rt_1; Rt_2) &\mapsto \exists f.Rt_1; \exists f.Rt_2 \\
\exists f^-.(Rt_1; Rt_2) &\mapsto \exists f^-.Rt_1; \exists f^-.Rt_2
\end{aligned}$$

□

The definition of Norm is an adaptation of referring expression type normalization in [2] with the following consequences: (1) $\mathcal{L}(Rt) = \mathcal{L}(\mathsf{Norm}(Rt))$, and (2) all *preference operators* (";") are at the top level of $\mathsf{Norm}(Rt)$. We call the maximal ";"-free parts of $\mathsf{Norm}(Rt)$ *preference-free components*. The following auxiliary function will then be used in formulating subsumptions with TDDs to statically test for singularity of each preference free component of $\mathsf{Norm}(Rt)$.

$$\begin{aligned}
\mathsf{Td}(\{?\}) &= id \\
\mathsf{Td}(\mathrm{A}) &= \mathrm{A}? \\
\mathsf{Td}(\exists f.Rt) &= f.\,\mathsf{Td}(Rt) \\
\mathsf{Td}(\exists f^-.Rt) &= f^-.\,\mathsf{Td}(Rt) \\
\mathsf{Td}(Rt_1 \sqcap Rt_2) &= (\mathsf{Td}(Rt_1), \mathsf{Td}(Rt_2))
\end{aligned}$$

The function extracts a tree description leading to nominals from the preference-free referring expression type. Altogether, the singularity test in [2, Theorem 20] will now apply:

Theorem 7. *Let $\mathcal{T}$ be a TBox in set-$\mathcal{DLFDI}$ and Rt a referring expression type. Then all referring concepts in $\mathcal{L}(Rt)$ are singular with respect to $\mathcal{T}$ if and only if*

$$\mathcal{T} \models \top \sqsubseteq \top : \mathsf{Td}(Rt') \to id$$

holds for every preference-free component Rt' of $\mathsf{Norm}(Rt)$. □

Note that—unlike in [5, Theorem 10]—the concepts (A) that guard the applicability of the TDD are now part of the Td tree descriptions themselves. In this way, completeness is regained even for Tds that contain inverse features.

4. TDDs and Decidability

We show by construction that $\mathcal{DLTDI}$ is decidable and complete for EXPTIME. The construction proceeds by transforming the $\mathcal{DLTDI}$ (non-)entailment to satisfiability of an Ackermann class [12] formula, $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$, constructed from the given entailment problem, and then relying on results in [13]. The construction itself is a variation on the approach introduced in [5] adapted for $\mathcal{DLTDI}$. In the absence of qualified existential restrictions on inverse features we can show the following Lemmata corresponding to those in [5]:

Lemma 8. *Let $\mathcal{T}$ be a $\mathcal{DLTDI}$ TBox and $\mathcal{Q} = \mathrm{A} \sqsubseteq \mathrm{B}$ a posed question. For any counterexample $\mathcal{J}$ to $\mathcal{T} \models \mathcal{Q}$, there is a tree-shaped counterexample $\mathcal{I}$ for which $\mathcal{I} \models \mathcal{T}$, $\mathcal{I}$ is rooted by a witness $o \in (\mathrm{A} \sqcap \neg \mathrm{B})^{\mathcal{I}}$, and there is no element of $\triangle^{\mathcal{I}}$ that has two or more incoming features f for any $f \in \mathsf{F}$.*

Proof: The tree-shaped model is constructed by unravelling the original counterexample model, starting from the witness o, as follows. For every $f \in \mathsf{F}$: (i) all f successors of an object are added to $\mathcal{I}$ and subsequently unravelled, and (ii) exactly one of the f-predecessors is also added and unravelled (note that the single f-predecessor can be the parent of the current object in which case we do not add any of the other f-predecessors). It follows that, in such an unravelling, all simple subsumptions in $\mathcal{T}$ must be satisfied since all simple subsumptions that hold in $\mathcal{J}$ remain true in $\mathcal{I}$ and it is sufficient to retain only one r-predecessor to satisfy constraints of the form $\mathrm{A} \sqsubseteq \exists f^-$. Also, since *no object in $\triangle$ that has two or more incoming features f for any $f \in \mathsf{F}$*, all PDDs in $\mathcal{T}$ hold vacuously. □

Note that Pd paths in $\mathcal{I}$ starting from the witness o uniquely identify objects in $\triangle$ reachable from o.

Lemma 9. *Let $\mathcal{T}$ be a $\mathcal{DLTDI}$ TBox and $\mathcal{Q} = \mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Td} \rightarrow \mathsf{Td}'$ a posed question. For any counterexample $\mathcal{J}$ to $\mathcal{T} \models \mathcal{Q}$, there is also a counterexample $\mathcal{I}$, called a* two-tree *counterexample, constructed as follows:*

1. *let $\mathcal{I}_1$ be an unravelling of $\mathcal{J}$ from an object $o_1 \in \mathrm{A}^{\mathcal{J}}$;*
2. *let $\mathcal{I}_2$ be an unravelling of $\mathcal{J}$ from an object $o_2 \in \mathrm{B}^{\mathcal{J}}$; and*
3. *let $\mathcal{I}$ be a disjoint union of $\mathcal{I}_1$ and $\mathcal{I}_2$ modified to agree on a path Pd starting in o_1 and o_2, respectively, if and only if $\mathsf{Td}(o_1, o_2)$.*

Proof: The unravellings $\mathcal{I}_1$ and $\mathcal{I}_2$ are as in Lemma 8. The domain $\triangle$ is then the disjoint union of the domains of $\mathcal{I}_1$ and $\mathcal{I}_2$ factored by the equivalence relation generated by the path agreements. Note that it is sufficient to only consider path agreements that are *symmetric* with respect to o_1 and o_2; this is a consequence of Lemma 8 and of the syntactic structure of the TDD concept constructor. □

The two Lemmata show that (1) rooted models of an $\mathcal{DLTDI}$ TBox can always be unravelled to tree-shaped models (and thus all PDDs in such models are satisfied vacuously) and (2) counterexamples to PDD-based posed questions can always be unravelled into a combination of two rooted tree models in which all equalities induced by the PDD subsumptions are between pairs of nodes that are reached by the same Td from the respective roots.

We will now outline how the existence of the counterexample model can be witnessed by a model of an Ackermann prefix formula $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ constructed from the given entailment problem $\mathcal{T} \models \mathcal{Q}$. There are several issues that the $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ construction must account for:

1. A (Herbrand) model of a satisfiable Ackermann formula can be viewed as a complete k-ary tree with nodes labelled by unary predicate symbols where k is the number of unary function symbols/inner existentials in the quantifier prefix. To simulate the two-tree counterexamples to $\mathcal{DLTDI}$ entailments we need to use $\mathsf{P}^{\mathrm{C}}_L$ and $\mathsf{P}^{\mathrm{C}}_R$ unary predicates to capture concept membership in the left and right part of our counterexample to $\mathcal{T} \models \mathcal{Q}$, respectively, in a single tree model;
2. Terms in the Ackermann formulas are constructed from unary function symbols and constant(s). To account for inverse features in $\mathcal{DLTDI}$ we will use two function symbols, f and $\bar{\mathsf{f}}$, for each feature f, standing for f and f^-, respectively. This is necessary as the Ackermann class does not admit equality;
3. Terms in the Ackermann formulas are total functions while features $\mathcal{DLTDI}$ may be partial. We use unary predicates N_L and N_R to *mark* nodes that correspond to nodes that must exist in the two-tree counterexample; and
4. Equalities between the two parts of the two-tree counterexample are simulated by the E_{id} unary predicate in the Ackermann formula. However, to determine all the necessary equalities, we use additional auxiliary predicates E_{Td} and F_{Td} where Td ranges over all tree sub-descriptions of tree descriptions appearing in a TDD in $\mathcal{T} \cup \{\mathcal{Q}\}$. These auxiliary predicates are constrained in $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ as follows:

$$
\begin{aligned}
\forall x.\mathsf{N}_L(\mathsf{f}(x)) \wedge \mathsf{N}_R(\mathsf{f}(x)) \wedge \mathsf{E}_{f.\,\mathsf{Td}}(x) &\leftarrow \mathsf{E}_{\mathsf{Td}}(\mathsf{f}(x)) \\
\forall x.\mathsf{N}_L(\bar{\mathsf{f}}(x)) \wedge \mathsf{N}_R(\bar{\mathsf{f}}(x)) \wedge \mathsf{E}_{f.\,\mathsf{Td}}(\bar{\mathsf{f}}(x)) &\leftarrow \mathsf{E}_{\mathsf{Td}}(x) \\
\forall x.\mathsf{N}_L(\mathsf{f}(x)) \wedge \mathsf{N}_R(\mathsf{f}(x)) \wedge \mathsf{E}_{f^-.\,\mathsf{Td}}(\mathsf{f}(x)) &\leftarrow \mathsf{E}_{\mathsf{Td}}(x) \\
\forall x.\mathsf{N}_L(\bar{\mathsf{f}}(x)) \wedge \mathsf{N}_R(\bar{\mathsf{f}}(x)) \wedge \mathsf{E}_{f^-.\,\mathsf{Td}}(x) &\leftarrow \mathsf{E}_{\mathsf{Td}}(\bar{\mathsf{f}}(x)) \\
\forall x.\mathsf{E}_{\mathrm{A}?}(x) &\leftarrow \mathsf{P}^{\mathrm{A}}_L(x) \wedge \mathsf{P}^{\mathrm{A}}_R(x) \\
\forall x.\mathsf{E}_{(\mathsf{Td}_1,\mathsf{Td}_2)}(x) &\leftarrow \mathsf{E}_{\mathsf{Td}_1}(x) \wedge \mathsf{E}_{\mathsf{Td}_2}(x)
\end{aligned}
$$

$$
\begin{aligned}
\forall x.\mathsf{N}_L(\mathsf{f}(x)) \wedge \mathsf{N}_R(\mathsf{f}(x)) \wedge \mathsf{F}_{f.\,\mathsf{Td}}(x) &\rightarrow \mathsf{F}_{\mathsf{Td}}(\mathsf{f}(x)) \\
\forall x.\mathsf{N}_L(\bar{\mathsf{f}}(x)) \wedge \mathsf{N}_R(\bar{\mathsf{f}}(x)) \wedge \mathsf{F}_{f.\,\mathsf{Td}}(\bar{\mathsf{f}}(x)) &\rightarrow \mathsf{F}_{\mathsf{Td}}(x) \\
\forall x.\mathsf{N}_L(\mathsf{f}(x)) \wedge \mathsf{N}_R(\mathsf{f}(x)) \wedge \mathsf{F}_{f^-.\,\mathsf{Td}}(\mathsf{f}(x)) &\rightarrow \mathsf{F}_{\mathsf{Td}}(x) \\
\forall x.\mathsf{N}_L(\bar{\mathsf{f}}(x)) \wedge \mathsf{N}_R(\bar{\mathsf{f}}(x)) \wedge \mathsf{F}_{f^-.\,\mathsf{Td}}(x) &\rightarrow \mathsf{F}_{\mathsf{Td}}(\bar{\mathsf{f}}(x)) \\
\forall x.\mathsf{F}_{\mathrm{A}?}(x) &\rightarrow \mathsf{P}^{\mathrm{A}}_L(x) \wedge \mathsf{P}^{\mathrm{A}}_R(x) \\
\forall x.\mathsf{F}_{(\mathsf{Td}_1,\mathsf{Td}_2)}(x) &\rightarrow \mathsf{F}_{\mathsf{Td}_1}(x) \wedge \mathsf{F}_{\mathsf{Td}_2}(x)
\end{aligned}
$$

together with $\forall x.\mathsf{N}_L(x) \wedge \mathsf{N}_R(x) \wedge \mathsf{F}_{id}(x) \rightarrow \mathsf{E}_{id}(x)$. Intuitively the E_{Td} tells us where the two tree models already agree (on Td) and F_{Td} tells us where equalities need to be asserted (for existing objects only). Note that due to the use of $\mathsf{GD}_{\mathsf{Td}}$ in the semantics definition of a TDD, all the required paths must already exist in the L/R parts of the model. With the help of these auxiliary definitions, a TDD $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Td} \rightarrow \mathsf{Td}'$ will be translated to

$$\forall x.\mathsf{N}_L(x) \wedge \mathsf{N}_R(x) \wedge \mathsf{P}^{\mathrm{A}}_L(x) \wedge \mathsf{P}^{\mathrm{B}}_R(x) \wedge \mathsf{E}_{\mathsf{Td}}(x) \rightarrow \mathsf{F}_{\mathsf{Td}'}(x) \text{ and}$$
$$\forall x.\mathsf{N}_L(x) \wedge \mathsf{N}_R(x) \wedge \mathsf{P}^{\mathrm{A}}_R(x) \wedge \mathsf{P}^{\mathrm{B}}_L(x) \wedge \mathsf{E}_{\mathsf{Td}}(x) \rightarrow \mathsf{F}_{\mathsf{Td}'}(x).$$

The Ackermann formula $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ first *constructs* a prototypical counterexample to $\mathcal{Q}$ using ground atomic conjuncts added to $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$. The rest of the $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ conjuncts then simulate the subsumptions in $\mathcal{T}$ (for details see [5]). For example:

- For $A \sqsubseteq B \in \mathcal{T}$, we add $\forall x.\mathsf{N}_L(x) \wedge \mathsf{P}^{\mathrm{A}}_L(x) \rightarrow \mathsf{P}^{\mathrm{B}}_L(x)$ and $\forall x.\mathsf{N}_R(x) \wedge \mathsf{P}^{\mathrm{A}}_R(x) \rightarrow \mathsf{P}^{\mathrm{B}}_R(x)$ as conjuncts in $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$;
- For $A \sqsubseteq \exists f \in \mathcal{T}$, we add $\forall x.\mathsf{N}_L(\bar{\mathsf{f}}(x)) \wedge \mathsf{P}^{\mathrm{A}}_L(\bar{\mathsf{f}}(x)) \rightarrow \mathsf{N}_L(x)$ and $\forall x.\mathsf{N}_L(x) \wedge \mathsf{P}^{\mathrm{A}}_L(x) \rightarrow \mathsf{N}_L(\mathsf{f}(x))$ for x not of the form $\bar{\mathsf{f}}(y)$ (and analogously for the R versions);
- etc., for all subsumptions in $\mathcal{T}$.

In both cases, models of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ can be converted to interpretations that are counterexamples to $\mathcal{T} \models \mathcal{Q}$ (and vice versa).

The properties of $\mathcal{DLTDI}$ models established by Lemmata 8 and 9 allow one to reduce the existence of a counterexample to a posed question to satisfiability of a formula in the Ackermann class prefix $\exists^*\forall\exists^*$ [12]. We use the Skolemized version of the problem that replaces the inner existential variables by appropriate unary function symbols and the outer ones by constants.

Theorem 10. *Let $\mathcal{T}$ be a $\mathcal{DLTDI}$ TBox and $\mathcal{Q}$ a posed question. Then $\mathcal{T} \models \mathcal{Q}$ if and only if $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ is not satisfiable.*

Proof: We define an auxiliary map of paths in a DL interpretation to paths in an interpretation of an Ackermann formula as follows. Given a path Pd in a tree-shaped DL interpretation we set

$$\mathsf{pm}(id) = 0, \mathsf{pm}(f.\,\mathsf{Pd}) = \mathsf{f}(\mathsf{pm}(\mathsf{Pd})), \text{ and } \mathsf{pm}(f^{-}.\,\mathsf{Pd}) = \bar{\mathsf{f}}(\mathsf{pm}(\mathsf{Pd})).$$

Now for a posed question $\mathcal{Q}$ of the form $\mathrm{A} \sqsubseteq \mathrm{B}$ and a tree-shaped counter-example $\mathcal{I}$ starting from o (see Lemma 8) we create a model of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ as follows: we assert

- $\mathsf{N}_L(\mathsf{pm}(\mathsf{Pd}))$ for every path $o.\,\mathsf{Pd}^{\mathcal{I}}$ that exists in $\mathcal{I}$, and
- $\mathsf{P}^{\mathrm{A}}_L(\mathsf{pm}(\mathsf{Pd}))$ whenever $o.\,\mathsf{Pd}^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$.

in $\mathcal{I}$ and verify that all formulæ in $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ are true.

Similarly, given a model of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ we construct $\mathcal{I}$ as follows:

- $\triangle = \{o_{\mathsf{Pd}} \mid \mathsf{N}_L(\mathsf{pm}(\mathsf{Pd})) \text{ holds}\}$;
- $\mathrm{A}^{\mathcal{I}} = \{o_{\mathsf{Pd}} \mid \mathsf{N}_L(\mathsf{pm}(\mathsf{Pd})) \wedge \mathsf{P}^{\mathrm{A}}_L(\mathsf{pm}(\mathsf{Pd})) \text{ holds}\}$; and
- $f^{\mathcal{I}} = \{(o_{\mathsf{Pd}}, o_{f.\,\mathsf{Pd}}) \mid \mathsf{N}_L(\mathsf{f}.\,\mathsf{pm}(\mathsf{Pd}))\} \cup \{(o_{f^{-}.\,\mathsf{Pd}}, o_{\mathsf{Pd}}) \mid \mathsf{N}_L(\bar{\mathsf{f}}.\,\mathsf{pm}(\mathsf{Pd}))\}$,

and verify that $\mathcal{I}$ is a model of $\mathcal{T}$ and o_{id} falsifies $\mathcal{Q}$.

For a posed question $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Td} \rightarrow \mathsf{Td}'$ and a two-tree counterexample $\mathcal{I}$ starting in o_1 and o_2 (see Lemma 9) we create a model of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$ as follows: we assert

- $\mathsf{N}_L(\mathsf{pm}(\mathsf{Pd}))$ for every path $o_1.\,\mathsf{Pd}^{\mathcal{I}}$ that exists in $\mathcal{I}$;
- $\mathsf{P}^{\mathrm{A}}_L(\mathsf{pm}(\mathsf{Pd}))$ whenever $o_1.\,\mathsf{Pd}^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$;
- $\mathsf{N}_R(\mathsf{pm}(\mathsf{Pd}))$ for every path $o_2.\,\mathsf{Pd}^{\mathcal{I}}$ that exists in $\mathcal{I}$;
- $\mathsf{P}^{\mathrm{A}}_R(\mathsf{pm}(\mathsf{Pd}))$ whenever $o_2.\,\mathsf{Pd}^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$; and
- $\mathsf{E}_{id}(\mathsf{pm}(\mathsf{Pd}))$ whenever $o_1.\,\mathsf{Pd}^{\mathcal{I}} = o_2.\,\mathsf{Pd}^{\mathcal{I}}$.

We extend this to the remaining auxiliary predicates E_{Td} and F_{Td} in a natural way and verify (by inspection of all cases) that this indeed yields the required model of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$.

The other direction constructs a two-tree model of $\mathcal{T}$ that falsifies $\mathcal{Q}$ using the construction for $\mathrm{A} \sqsubseteq \mathrm{B}$ twice (once for L starting in o^L_{id} and once for R starting in o^R_{id}) and equating tree descriptions $o^L_{id}.\,\mathsf{Pd}$ and $o^R_{id}.\,\mathsf{Pd}$ for which $E_{id}(\mathsf{pm}(\mathsf{Pd}))$ holds in the model of $\mathsf{ToAck}(\mathcal{T}, \mathcal{Q})$. That yields a two-tree model of $\mathcal{T}$ in which o^L_{id} and o^R_{id} witness the violation of $\mathcal{Q}$. □

Applying the result from [13] and the observation that EXPTIME is closed under complements we have:

Corollary 11. *The entailment problem in $\mathcal{DLTDI}$ is complete for EXPTIME.* □

5. Undecidable Combinations of Agreements

In this section we show that combining variants of *path* agreements (recall that the structural agreements for PDDs are a special case of tree description agreements for TDDs) lead to undecidability even in the special case of PDDs. In the following section we write $\mathsf{Pd}^{\simeq}(x, y)$ to state that x and y structurally agree on the PDD Pd (regarded to be a TDD, see Section 2). Alternative forms of PDD agreements introduced in [6] have been defined as follows:

Definition 12 (Sub-structural Path-Based Agreement(s)). *Let $\mathcal{I}$ be an interpretation and o_1 and o_2 be two $\triangle$ elements. We write $\mathsf{Pd}^{\cap}(o_1, o_2)$ to express $\mathsf{Pd}^{\mathcal{I}}(\{o_1\}) \cap \mathsf{Pd}^{\mathcal{I}}(\{o_2\}) \neq \emptyset$ (set intersection agreement) and $\mathsf{Pd}^{\approx}(o_1, o_2)$ to express $\mathsf{Pd}^{\mathcal{I}}(\{o_1\}) = \mathsf{Pd}^{\mathcal{I}}(\{o_2\}) \neq \emptyset$ (non-empty set agreement) where*

$$\mathsf{Pd}^{\mathcal{I}}(S) = \begin{cases} S & \text{if } \mathsf{Pd} = id, \\ \mathsf{Pd}_1^{\mathcal{I}}(\{f^{\mathcal{I}}(x) \mid x \in S\}) & \text{if } \mathsf{Pd} = f.\,\mathsf{Pd}_1, \\ \mathsf{Pd}_1^{\mathcal{I}}(\{x \mid f^{\mathcal{I}}(x) \in S\}) & \text{if } \mathsf{Pd} = f^{-}.\,\mathsf{Pd}_1, \\ \mathsf{Pd}_1^{\mathcal{I}}(\{x \mid x \in \mathrm{A}^{\mathcal{I}} \cap S\}) & \text{if } \mathsf{Pd} = \mathrm{A}?.\,\mathsf{Pd}_1\,. \end{cases}$$ □

Note that in the above definition we only consider agreements based on *non-empty* set equality as general set equality-based agreements were shown to lead to undecidability of entailment in [5]. The following is a straightforward consequence of our definitions:

Lemma 13. *Structural path equality ($\simeq$) and non-empty set equality ($\approx$) coincide when Pd contains at most one inverse feature.* □

It has been shown that entailment in $\{\cap, \approx\}$-$\mathcal{DLFDI}$, a member of *set-$\mathcal{DLFDI}$* that allows $\cap$ and $\approx$ path agreements in PDDs, is undecidable [6]. The same construction shows that $\{\cap, \simeq\}$-$\mathcal{DLFDI}$ entailment is also undecidable as $\approx$-equality and $\simeq$-equality coincide on PDDs that contain a single inverse feature (see Lemma 13). In the remainder of this section we show that entailment for $\{\simeq, \approx\}$-$\mathcal{DLFDI}$ is undecidable as well.

To show undecidability we construct a $\{\simeq, \approx\}$-$\mathcal{DLFDI}$ TBox $\mathcal{T}(T, H, V)$ for a given unconstrained tiling problem (T, H, V) such that (T, H, V) has a solution if and only if $\mathcal{T}(T, H, V) \not\models \mathrm{A} \sqsubseteq \mathrm{DA} : f \to id$.

The TBox $\mathcal{T}(T, H, V)$ consists of the following collection of subsumptions. Constraints enforcing a grid for a counterexample to $\mathrm{A} \sqsubseteq \mathrm{DA} : f \to id$ proceeds as follows (see Figure 3, top two rows). We start by requiring certain features to be total and certain inverse features to exist:

$$\begin{aligned} &\top \sqsubseteq \forall h.\top \sqcap \forall u.\top \sqcap \forall v.\top \sqcap \forall w.\top \\ &\mathrm{A} \sqcup \mathrm{DA} \sqcup \mathrm{B} \sqcup \mathrm{DB} \sqcup \mathrm{C} \sqcup \mathrm{DC} \sqsubseteq \exists f^{-} \sqcap \exists g^{-} \end{aligned} \tag{1}$$

We then force DA objects to have two (or more) incoming f features and A's only one incoming f. This is achieved by requiring that A and DA that agree of f must $\approx$-agree on the Pd $f^{-}.g^{-}.h$. However, for these two objects not to be forced to be the same, they must $\simeq$-disagree on the same Pd. This is only

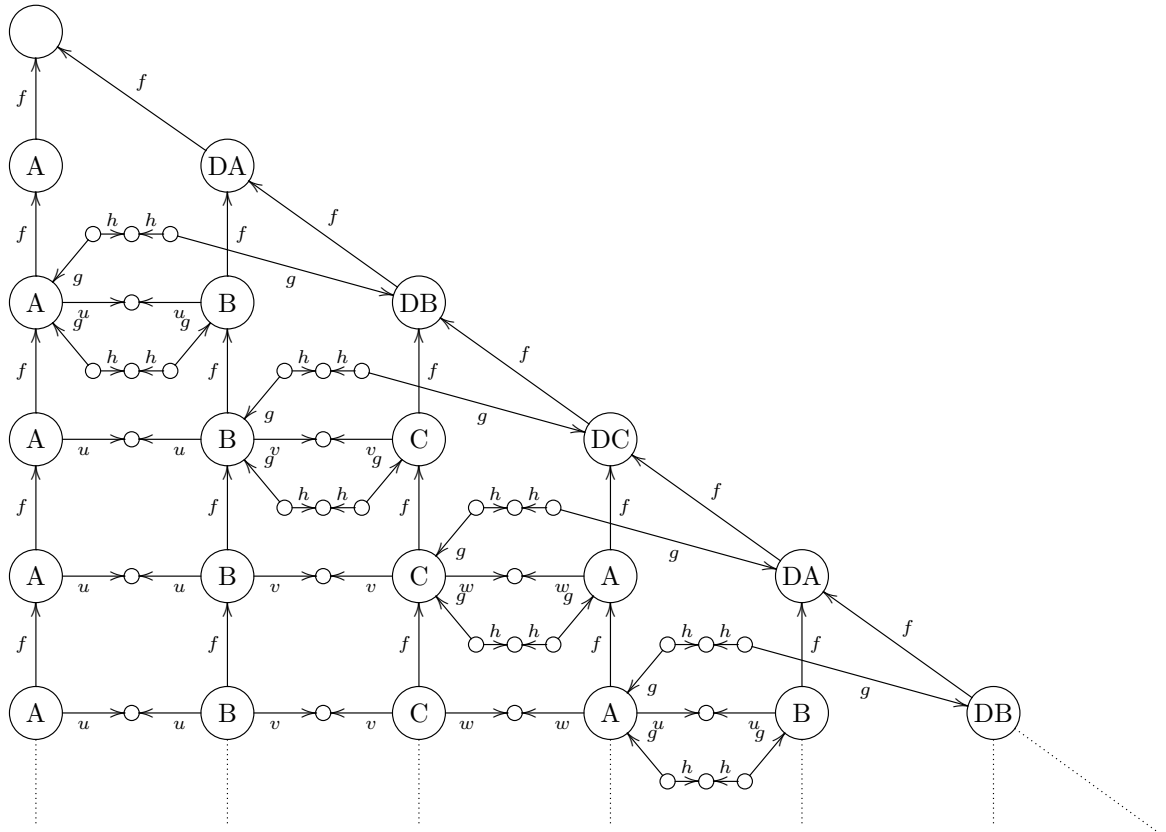

Figure 3: Construction of a Grid for Tiling for $\mathrm{A} \sqsubseteq \mathrm{DA} : f \to id$.

possible if Pd *branches* on f^- starting from DA but on g^- starting from A (as f is a key on As). We proceed similarly for the B–DB and C–DC pairs of objects in the subsequent rows in Figure 3.

$$\begin{array}{l} \mathrm{A} \sqsubseteq \mathrm{DA} : f \to (f^-.g^-.h)^{\approx} \\ \mathrm{A} \sqsubseteq \mathrm{DA} : (f^-.g^-.h)^{\simeq} \to id \\ \mathrm{B} \sqsubseteq \mathrm{DB} : f \to (f^-.g^-.h)^{\approx} \\ \mathrm{B} \sqsubseteq \mathrm{DB} : (f^-.g^-.h)^{\simeq} \to id \\ \mathrm{C} \sqsubseteq \mathrm{DC} : f \to (f^-.g^-.h)^{\approx} \\ \mathrm{C} \sqsubseteq \mathrm{DC} : (f^-.g^-.h)^{\simeq} \to id \end{array} \tag{2}$$

We force f-predecessors of A also to be As. We assert the same for B and C.

$$\forall f.\mathrm{A} \sqsubseteq \mathrm{A} \quad \forall f.\mathrm{B} \sqsubseteq \mathrm{B} \quad \forall f.\mathrm{C} \sqsubseteq \mathrm{C} \tag{3}$$

We force the two f-predecessors of DA to be B and DB, respectively; the key constraints for f on both Bs and DBs ensure that (and, moreover, that there must be exactly two f-predecessors). Analogous constraints constrain the C–DC and A–DA pairs of objects in the subsequent rows.

$$\begin{array}{l} \forall f.\mathrm{DA} \sqsubseteq \mathrm{B} \sqcup \mathrm{DB} \\ \forall f.\mathrm{DB} \sqsubseteq \mathrm{C} \sqcup \mathrm{DC} \\ \forall f.\mathrm{DC} \sqsubseteq \mathrm{A} \sqcup \mathrm{DA} \\ \mathrm{A} \sqsubseteq \mathrm{A} : f \to id \quad \mathrm{DA} \sqsubseteq \mathrm{DA} : f \to id \\ \mathrm{B} \sqsubseteq \mathrm{B} : f \to id \quad \mathrm{DB} \sqsubseteq \mathrm{DB} : f \to id \\ \mathrm{C} \sqsubseteq \mathrm{C} : f \to id \quad \mathrm{DC} \sqsubseteq \mathrm{DC} : f \to id \end{array} \tag{4}$$

We use the features u, v, and w, to establish the horizontal grid lines between A and B, B and C, and C and A, respectively.

$$\begin{array}{ll} \mathrm{A} \sqsubseteq \mathrm{B} : f.f \to u & \mathrm{A} \sqsubseteq \mathrm{B} : f.u \to u \\ \mathrm{B} \sqsubseteq \mathrm{C} : f.f \to v & \mathrm{B} \sqsubseteq \mathrm{C} : f.v \to v \\ \mathrm{C} \sqsubseteq \mathrm{A} : f.f \to w & \mathrm{C} \sqsubseteq \mathrm{A} : f.w \to w \end{array} \tag{5}$$

We use the feature f between As, Bs, and Cs as the vertical lines of the grid. Hence, to finish the construction of the tiling we need to assign tiles T_i to all grid points A, B, and C:

$$\mathrm{A} \sqcup \mathrm{B} \sqcup \mathrm{C} \sqsubseteq \sqcup_{t_i \in T} \mathrm{T}_i, \tag{6}$$

and enforce the horizontal,

$$\begin{array}{l} \mathrm{T}_i \sqcap \mathrm{A} \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{B}) : u \to id, \\ \mathrm{T}_i \sqcap \mathrm{B} \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{C}) : v \to id, \\ \mathrm{T}_i \sqcap \mathrm{C} \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{A}) : w \to id, \quad \text{for all } (t_i, t_j) \notin H, \end{array} \tag{7}$$

and vertical tiling rules,

$$\begin{array}{l} \forall f.(\mathrm{T}_i \sqcap \mathrm{A}) \sqcap \mathrm{T}_j \sqsubseteq \bot, \\ \forall f.(\mathrm{T}_i \sqcap \mathrm{B}) \sqcap \mathrm{T}_j \sqsubseteq \bot, \\ \forall f.(\mathrm{T}_i \sqcap \mathrm{C}) \sqcap \mathrm{T}_j \sqsubseteq \bot, \quad \text{for all } (t_i, t_j) \notin V. \end{array} \tag{8}$$

In summary, a TBox $\mathcal{T}(T, H, V)$ containing all the subsumptions (1), (2), (3), (4), (5), (6), (7), and (8) 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 (TM) 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 overlayed 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.

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 (7) as follows:

$$\begin{array}{l} \forall f.(\mathrm{T}_i \sqcap \mathrm{A}) \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{B}) : u \to id, \\ \forall f.(\mathrm{T}_i \sqcap \mathrm{B}) \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{C}) : v \to id, \\ \forall f.(\mathrm{T}_i \sqcap \mathrm{C}) \sqsubseteq (\mathrm{T}_j \sqcap \mathrm{A}) : w \to id, \quad \forall (t_i, t_j) \notin H, \end{array} \tag{7'}$$

The above constructions yield reductions of entailment in $\{\simeq, \approx\}$-$\mathcal{DLFDI}$. Together with the observation at the beginning of this section we have:

Theorem 14. *The entailment problems in the description logics $\{\simeq, \approx\}$-$\mathcal{DLFDI}$ and $\{\simeq, \cap\}$-$\mathcal{DLFDI}$ are undecidable.*

Proof: Immediately follows from the preceding tiling construction. □

This also shows undecidability of combining $\mathcal{DLTDI}$ with various members of the *set-$\mathcal{DLFDI}$* family (that are not special cases of $\mathcal{DLTDI}$).

6. Summary

We have introduced an extension to the *set-$\mathcal{DLFDI}$* family of DLs by introducing a new type of agreement, namely the *tree description agreement*, that refines the path agreements proposed in the past [5, 6, 8]. We show that the computational properties of $\mathcal{DLTDI}$ remain the same as for the

logics $\{\cap\}$-$\mathcal{DLFDI}$, $\{\approx\}$-$\mathcal{DLFDI}$, and $\{\simeq\}$-$\mathcal{DLFDI}$. In particular, entailment remains EXPTIME-complete. We also show that in DLs allowing any (binary) combination of the various path agreements, namely $\{\cap, \simeq\}$-$\mathcal{DLFDI}$ and $\{\approx, \simeq\}$-$\mathcal{DLFDI}$, the entailment becomes undecidable. This carries over to $\mathcal{DLTDI}$ as $\{\approx\}$-$\mathcal{DLFDI}$ is a special case of $\mathcal{DLTDI}$. Thus, our results completely characterize the computational properties for all (currently known) members of this family of DLs.

Future work. There are several possible directions:

- In [6], a typing discipline based on the last component of a path description was introduced to regain decidability for $\{\cap, \approx\}$-$\mathcal{DLFDI}$. This approach is not directly applicable to structural agreements and developing appropriate restrictions to regain decidability in $\{\cap, \simeq\}$-$\mathcal{DLFDI}$, $\{\approx, \simeq\}$-$\mathcal{DLFDI}$, and combinations of sub-structural agreements with $\mathcal{DLTDI}$ remains open.
- With the TDD concept constructor, we can enrich the test concepts $C?$ in Tds to $(C_1, C_2)?$ with the intention that the pairs of paths (starting from x and y in Definition 4) in the semantics of PDDs must simultaneously traverse objects belonging to C_1 and C_2, respectively (rather than just to C). While this extension is easy to incorporate in our decision procedure, it is not clear if it increases the expressive power of the language.
- The current approach only allows posed questions of the form $\mathrm{A} \sqsubseteq \mathrm{B}$ or $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Td} \to \mathsf{Td}'$ where A and B are primitive concepts or concepts definable in $\mathcal{DLTDI}$. Extending the language of posed questions, e.g., with *qualified existential restrictions* would allow one to consider, in turn, more complex referring expression types.
- Finally, we propose studying fragments of *set-*$\mathcal{DLFDI}/\mathcal{DLTDI}$ with preferable computational properties, e.g., Horn fragments of the proposed family of DLs. This direction can be based on the use of syntactically restricted TDDs along the lines of restricting PFDs in FunDL-Lite logics [4].

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] B. Russell, On denoting, Mind 14 (1905) 479–493. URL: http://www.jstor.org/stable/2248381.

[2] 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.

[3] A. Borgida, E. Franconi, D. Toman, G. E. Weddell, Understanding document data sources using ontologies with referring expressions, in: AI 2022: Advances in Artificial Intelligence, volume 13728 of *LNCS*, Springer, 2022, pp. 367–380.

[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*, 2023.

[6] E. Feng, D. Toman, G. E. Weddell, On mixed semantics of path description dependencies in FunDL, in: Description Logics, volume in press of *CEUR Workshop Proceedings*, 2024.

[7] E. Feng, E. Franconi, P. F. Patel-Schneider, D. Toman, G. E. Weddell, Equality generating dependencies in description logics via path agreements, in: AI (2), volume 15443 of *Lecture Notes in Computer Science*, 2024, pp. 214–227.

[8] D. Toman, G. E. Weddell, Structural equality generating dependencies and definite descriptions, in: Description Logics, CEUR Workshop Proceedings, 2025.
[9] 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.
[10] D. Toman, G. E. Weddell, On keys and functional dependencies as first-class citizens in description logics, J. Aut. Reasoning 40 (2008) 117–132.
[11] 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.
[12] W. Ackermann, Uber die Erfullbarkeit 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.