Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Solving Unification in the Description Logic $\mathcal{FL}_\perp$

Barbara Morawska, **Dariusz Marzec**
Institute of Computer Science, University of Opole, Poland
{barbara.morawska, dariusz.marzec}@uni.opole.pl

Abstract

We present an algorithm for solving the unification problem in the description logic $\mathcal{FL}_\perp$. This logic extends $\mathcal{FL}_0$ with the bottom constructor, and thus supports conjunction, value restrictions, top and bottom constructors. Unification of concepts can be a useful tool for ontology maintenance; however, little is known about unification even in small, restricted description logics. The unification problem has been solved only for $\mathcal{FL}_0$ and $\mathcal{EL}$. This paper contributes to the ongoing effort to extend these results to richer logics. Our algorithm runs in exponential time with respect to the size of the problem.

1 Introduction

Description Logics (DLs) are designed as a formal framework to represent and manipulate information stored in the form of an ontology – a set of definitions of concepts using simple concepts called *names* and complex concepts. Complex concepts are constructed from primitive names using relations and constructors. There are many DLs that differ in the constructors they provide for building complex concepts out of simple ones.

Unification of concepts in DLs was proposed in (Baader and Narendran 2001) as a non-standard reasoning service that can assist in maintaining large ontologies by identifying and eliminating redundancies.

Nevertheless, little is known about unification in DLs. The problem has been solved and is well understood only for sub-Boolean DLs such as $\mathcal{EL}$ and $\mathcal{FL}_0$. In this work, we focus on the unification of concepts in an extension of $\mathcal{FL}_0$, namely $\mathcal{FL}_\perp$.

The description logic $\mathcal{FL}_\perp$ allows the construction of complex concepts from a countable set of concept names and role names using conjunction, the top ($\top$) and bottom ($\perp$) constructors, and a value restriction of the form $\forall r.C$ where r is a relation with the value of the second argument restricted to a concept C. The description logic $\mathcal{FL}_0$ provides the same constructors except for $\perp$. Both of these lightweight DLs have polynomial-time subsumption algorithms.

In order to define the unification problem in a restricted description logic, we first designate some concept names as variables. Then, given a pair of concepts C and D that may contain variables, we seek a substitution that makes C and D equivalent (or makes one subsumed by the other). For example, consider the pair of concepts:
`Viral_disease`$\sqcap\forall$`has_attribute.Infectious`,
`Infectious_disease`$\sqcap\forall$`has_cause.Virus`.
These concepts can be made equal by treating `Infectious_disease` and `Viral_disease` as variables which can be substituted by other concepts or in other words, further specified. Then a unifier will be a mapping, or a set of definitions:

- `Infectious_disease:=Disease`$\sqcap$ $\forall$`has_attribute.Infectious`
- `Viral_disease:=Disease`$\sqcap\forall$`has_cause.Virus`.

As illustrated in the above example, the substitution for variables that we aim to obtain from a unification algorithm consists of a set of definitions for previously undefined concept names. Nevertheless, in what follows, we will treat the unification problem primarily as a decision problem: "Does such a substitution exist?" If it does, an example solution can then be recovered.

Unification of concepts was first studied and solved for the description logic $\mathcal{FL}_0$, (Baader and Narendran 2001), where the problem was shown to be ExpTime-complete.The unification algorithm presented in that work proceeds by first reducing the unification problem to the problem of solving formal language equations, and then further reducing it to an emptiness test for a tree automaton.

Using similar methods, the unification problem was later solved for the description logic $\mathcal{FL}_{reg}$ in (Baader and Küsters 2001) and for $\mathcal{FL}_{\perp reg}$ in (Baader and Küsters 2002). The description logic $\mathcal{FL}_{reg}$ extends $\mathcal{FL}_0$ by allowing regular expressions constructed over the set of role names, in place of simple role names, for defining value restrictions. The description logic $\mathcal{FL}_{\perp reg}$ further extends $\mathcal{FL}_{reg}$ with the $\perp$ constructor. In the same paper, the authors admitted that the unification in $\mathcal{FL}_\perp$ does not yield to their method. In the latter work, however, the authors noted that unification in $\mathcal{FL}_\perp$ does not yield to their method.

Since then, there have been many–mostly unpublished–attempts to solve the unification problem for $\mathcal{FL}_\perp$. In (Borgwardt and Morawska 2012), a new approach to unification in $\mathcal{FL}_0$ was presented. In (Morawska 2021), an attempt was made to extend this approach to unification in $\mathcal{FL}_\perp$. However, this attempt was flawed: it appears to be impossible to reduce unification in $\mathcal{FL}_\bot$ to that in $\mathcal{FL}_0$, contrary to the conjecture made in that work. Here we present the algorithm which extends the unification in $\mathcal{FL}_0$ to that in $\mathcal{FL}_\bot$, but not by reduction. Instead, we provide a generalization of the unification approach for $\mathcal{FL}_0$ from (Borgwardt and Morawska 2012). Unification in $\mathcal{FL}_0$ is thus a special case of the proposed algorithm.

The paper is organized as follows. In Section 2, we formally present the logic $\mathcal{FL}_\bot$. The unification problem is defined in Section 3. Then, in Sections 4 (overview), 5 (preprocessing), 6, and 7 (main computation), we present the algorithm. In the next three sections (8, 9, and 10), we prove the correctness of the algorithm. The paper concludes with final remarks in Section 11. The extended version of this paper can be found at (Morawska and Marzec 2025)

2 The Description Logic $\mathcal{FL}_\bot$

The concepts in $\mathcal{FL}_\bot$ are generated by the following grammar from a finite set of concept names $\mathbf{N}$ and a finite set of role names $\mathbf{R}$:

$$C ::= A \mid C \sqcap C \mid \forall r.C \mid \top \mid \bot,$$

where $A \in \mathbf{N}$ and $r \in \mathbf{R}$.

The concepts are interpreted as subsets of a non-empty domain. An interpretation is a pair consisting of a domain and an interpreting function, $(\Delta^I, \cdot^I)$. A concept name A is interpreted by a subset of the domain, $A^I \subseteq \Delta^I$, a role name r is interpreted as a binary relation, $r^I \subseteq \Delta^I \times \Delta^I$, and a concept conjunction is interpreted by intersection: $(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$, value restriction $\forall r.C$ is interpreted as the set: $(\forall r.C)^I = \{e \in \Delta^I \mid \forall d \in \Delta^I((e,d) \in r^I \implies d \in C^I)\}$; $\top^I = \Delta^I$ and $\bot^I = \emptyset$.

Based on this semantics, we define the equivalence and subsumption relations between concepts as follows: $C \equiv D$ iff $C^I = D^I$ and $C \sqsubseteq D$ iff $C^I \subseteq D^I$, in every interpretation I.

The **subsumption (equivalence) problem** is then defined as follows.
Input: a pair of concepts C, D.
Output: *"Yes"*, if $C \sqsubseteq D$ ($C \equiv D$), and *"No"* otherwise.
In deciding the subsumption or equivalence problem between concepts in $\mathcal{FL}_\bot$, it is convenient to bring them first into a normal form.

A concept is in **normal form** if it is a conjunction of concepts of the form $\forall r_1.(\forall r_2.(\ldots \forall r_n.A))$, where A is a concept name or $\top$ or $\bot$. The empty conjunction is equivalent to $\top$.

Since for all concepts C_1 and C_2, the equivalence $\forall r.(C_1 \sqcap C_2) \equiv \forall r.C_1 \sqcap \forall r.C_2$ holds in $\mathcal{FL}_\bot$, each $\mathcal{FL}_\bot$-concept is equivalent to a concept in normal form.

Concepts of the form $\forall r_1.(\forall r_2.(\ldots \forall r_n.A)\ldots)$ are called *particles*. We identify a conjunction of particles with the set of its conjuncts. In other words, every concept in normal form is a set of particles, and the empty set of particles corresponds to $\top$.

For brevity, a particle of the form $\forall r_1.(\forall r_2.(\ldots \forall r_n.A)\ldots)$ will be written as $\forall v.A$, where $v = r_1 r_2 \ldots r_n$, and A may only be a concept name, $\top$ or $\bot$. The symbol 'v' is a word over $\mathbf{R}$, and we call it a *role string*. We say that the size of a particle is the length of its role string, $|v|$. Relation $\leq$ on role strings denotes the prefix relation, and $<$ denotes the proper prefix relation.

Definition 1. *(prefix) We say that a particle P is a* prefix *of P' in the following sense.*

- *$P = \forall v.\bot$, $P' = \forall v'.\bot$ and $v < v'$ or*
- *$P = \forall v.\bot$, $P' = \forall v'.A$ and $v \leq v'$, where A is a concept name. In particular, $\bot$ is a prefix of A.*

We call a particle of the form $\forall v.\top$, a $\top$-particle (top-particle), a particle of the form $\forall v.\bot$, a $\bot$-particle (bottom-particle), and a particle of the form $\forall v.A$, an A-particle, for a concept name A. Notice that only a $\bot$-particle may be a prefix of another particle.

Since $\forall v.\top \equiv \top$, any $\top$-particle occurring in a concept C may be simply deleted. Since $\forall v.\bot \sqsubseteq \forall v'.A$, where $v \leq v'$ and A is a concept name or $\bot$, if a concept C contains a particle $\forall v.\bot$, then for every v' such that $v \leq v'$, any particle $\forall v'.A$ may likewise be deleted from C. Since $\bot \sqsubseteq D$ for any concept D, if a concept C contains $\bot$ as a particle ($\bot \in C$), all other particles in C may be deleted.

If there are no deletions applicable to a concept C, we say that C is **reduced**. Below we consider only reduced concepts in normal form.

For concepts in normal form we have a simple polynomial-time way of deciding the subsumption problem. This follows directly from the following lemma.

Lemma 1. *(Characterization of subsumption in $\mathcal{FL}_\bot$) Let C, D be an instance of a subsumption problem, where C, D are concepts in normal form: $C = \{P_1, P_2, \ldots, P_m\}$ and $D = \{P'_1, P'_2, \ldots, P'_n\}$. Then $C \sqsubseteq D$ if and only if for every $P'_i \in D$, one of the two conditions holds:*

1. *$P'_i \in C$*
2. *$P'_i = \forall v.A$, where A is a constant or $\bot$, and there exists a bottom particle $\forall v'.\bot$ (where v' may be empty) in C such that $v' \leq v$.*

Proof. The lemma is justified by the properties of conjunction with respect to concepts in normal form. The first statement follows from the fact that each concept is a set of particles, and subsumption is a partial order on such sets. The second statement follows from the fact that $\bot$ is always interpreted as the empty set; hence, $\bot \sqsubseteq C$ for every concept C. This property is preserved for value restrictions constructed by prepending $\bot$ and C with a role string v: $\forall v.\bot \sqsubseteq \forall v.C$. ☐

When deciding a possible subsumption $\{P_1, \ldots, P_n\} \sqsubseteq \forall v.A$, where A is a concept name, we can ignore any P_i involving a concept name other than A. The reason is that any particle with a concept name different than A cannot affect passing any of the tests in Lemma 1.

Similarly, when deciding a possible subsumption $\{P_1, \ldots, P_n\} \sqsubseteq \forall v.\bot$, we can ignore any P_i not containing $\bot$, since according to the subsumption tests in Lemma 1 we check only for $\forall v.\bot \in \{P_1, \ldots, P_n\}$ or for $\forall v'.\bot \in \{P_1, \ldots, P_n\}$, where $v' \leq v$.

3 Unification Problem

In order to define the unification problem, we partition the set of concept names $\mathbf{N}$ into two disjoint sets $\mathbf{C}$ and $\mathbf{Var}$, called constants and variables, respectively. Variables are concept names that can be substituted by other concepts.

A substitution is a mapping from variables to $\mathcal{FL}_\perp$-concepts constructed over a finite set of variables and constants and it is extended to all concepts in a usual way. If a variable X is assigned $\top$ by a substitution γ and γ is clear from the context, we call X a $\top$-variable, and if it is assigned $\perp$, we call it a $\perp$-variable.

The input for unification problem is defined as a set of pairs of $\mathcal{FL}_\perp$-concepts, $\{(E_1, F_1), \ldots, (E_n, F_n)\}$. We call such pairs *possible subsumptions* or *goal subsumptions* for intuitive reasons. The formulation below uses the symbol $\sqsubseteq^?$ instead of a comma, in order to indicate what we want to achieve in a solution.

The unification problem
Input: $\Gamma = \{E_1 \sqsubseteq^? F_1, \ldots, E_n \sqsubseteq^? F_n\}$, where $E_1, \ldots, E_n, F_1, \ldots F_n$ are $\mathcal{FL}_\perp$ concepts in normal form, and $F_1, \ldots, F_n$ are particles. The concepts may contain variables.
Output: "Yes" if there exists a substitution γ (called a solution or a unifier of Γ) such that $\gamma(E_1) \sqsubseteq \gamma(F_1), \ldots, \gamma(E_n) \sqsubseteq \gamma(F_n)$; otherwise, "No".

In deciding unification, it is sufficient to consider ground unifiers. A substitution γ is ground if it assigns to variables concepts that contain no variables. Based on our earlier observations about the subsumption problem, we may further assume that the unification problem involves at most one constant. If multiple constants occur, we can decompose the problem into several subproblems, one for each constant $A_1, \ldots, A_n$ in Γ. For each A_i, let Γ_{A_i} be the problem obtained from Γ by replacing all constants other than A_i with $\top$. To solve Γ, we solve $\Gamma_{A_1}, \ldots, \Gamma_{A_n}$ separately. If $\gamma_1, \ldots, \gamma_n$ are solutions to these subproblems, then $\gamma = \gamma_1 \cup \cdots \cup \gamma_n$ is clearly a solution to Γ.

4 Overview of the Algorithm

Our unification algorithm consists of two stages: normalization and solving the normalized problem.

1. Normalization is a non-deterministic polynomial-time procedure. Some unification problems can already be determined to be unifiable at this stage; however, certain non-deterministic choices may lead to failure for the current branch. If all possible choices lead to failure, the algorithm terminates with failure as the final answer – the problem is not unifiable.
2. If the problem passes the first stage, we obtain a partial substitution (for variables set to $\perp$ or $\top$) and a unification problem in a special form, consisting of subsumptions of the type shown in Figure 3. If there are no start subsumptions, the problem has a solution assigning $\top$ to all variables. If there are no flat subsumptions, the problem has a solution. The construction is explained in Figure 4.

If the set of flat unsolved subsumptions is non-empty, the algorithm enters the next stage, namely *computing with shortcuts* (Section 6). At this stage, our algorithm can fail for the current choices in the normalization stage, or it can return success. If it returns success, this means that the problem is unifiable. We can recover an example of a unifier from the computations.

5 Normalization

The input for the algorithm is $\Gamma = \{E_1 \sqsubseteq^? P_1, \ldots, E_n \sqsubseteq^? P_n\}$, where $E_1, ..., E_n$ are sets of particles and $P_1, \ldots, P_n$ are particles that can contain variables or are $\perp$-particles or A-particles where A is a constant. We assume that Γ contains only one constant A. Since all concepts are reduced, $\top$ does not occur in Γ.

Initially all goal subsumptions are marked as *unsolved*.

1. Initial steps. For each variable X we have to guess if in a solution: X should be $\top$, or X should be $\perp$, or if X should be neither $\top$ nor $\perp$, but it should contain a constant A. The last guess is expressed by maintaining a Boolean variable A_X, which should be true if X contains A under a solution or false in the opposite case. If X is guessed to be $\perp$ or $\top$, then it is replaced by $\perp$ or $\top$, respectively, everywhere in the goal subsumptions.
2. Implicit Solver. We exhaustively apply the rules from Figure 1–in the given order–to the goal subsumptions. Implicit Solver may detect unifiability of a given problem at this stage or detect failure for the choices made for each variable in the first step.

   Note that after applying rule (a) and (b) of Figure 1, all variables that are guessed to be $\perp$ are eliminated from the goal. Hence in particular there is no $\perp$ on the left side of any goal subsumption after applying these rules.
3. Normalization loop. At this point, the algorithm enters a loop in which it alternates between applying the rules from Figure 2 and Figure 1.

   (a) Flattening (Applying rules from Figure 2). We choose ('don't care' nondeterministically) an unsolved non-flat subsumption from the unification problem.

   An unsolved goal subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? P$ is **non-flat** if

   - $P = \forall r.P'$, where P' is a particle or
   - there is i, $1 \leq i \leq n$ such that $C_i = \forall r.C'_i$, where C'_i is a particle.
   - there is i, $1 \leq i \leq n$ such that $C_i = A$.

   We flatten the subsumption by applying a rule from Figure 2.

   In Figure 2 we use the following notation. If P is a particle and r a role name ($r \in \mathbf{R}$), we define an operation on particles P^{-r} in the following way:

$$P^{-r} = \begin{cases} P^r & \text{if } P \text{ is a variable and} \\ & P^r \text{ its } \textit{decomposition variable}, \\ P' & \text{if } P = \forall r.P', \\ \top & \text{in all other cases.} \end{cases}$$

Implicit Solver rules:

(a) If $\bot$ is found in a subsumption s on its left side at top level, then we label s as solved.
(b) If $\bot$ is found on the right side of a subsumption s, and not on the left side, then return **failure**.
(c) If a $\top$-particle is found in a subsumption s on its right side, then we label s as solved.
(d) If $\top \sqsubseteq^? X$ is an unsolved subsumption and X is not chosen to be $\top$, then return **failure**.
(e) If a $\top$-particle is found in a subsumption s on its left side, we delete this particle from s.
(f) If a particle P occurs on the right side of s and also on the left side (at the top level), then label s as solved.
(g) If the constant A occurs on the right side of s and there is a variable X on the left side, such that A_X is true, then label s as solved.
(h) If the constant A occurs on the right side of s, but on the left side of s there is neither A nor a variable X with A_X true, then return **failure**.
(i) If A occurs on the left side of a subsumption s and X is on its right side, where A_X is false, then delete this occurrence of A.
(j) If A_X is true and X occurs on the right side of s, but neither A nor a variable Y with A_Y true occurs on the left hand side of s, then return **failure**.
(k) If there are no unsolved subsumptions left, return **success**.

Figure 1: Implicit Solver rules

If s is a subsumption of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $C_1, \ldots, C_n, D$ are particles, we define $s^{-r} = C_1^{-r} \sqcap \cdots \sqcap C_n^{-r} \sqsubseteq^? D^{-r}$.

Let P is a particle, then we define P^A, for a constant A in the following way:

$$P^A = \begin{cases} P & \text{if } P \text{ is } A \text{ or a variable,} \\ \top & \text{in all other cases.} \end{cases}$$

Decomposition variables The flattening rules applied to unsolved and non-flat goal subsumptions may generate new variables. These variables are called *decomposition variables*. For example if $X \sqsubseteq^? \forall r.Y$ is a goal subsumptions, then the first flattening rule applies and the subsumption is replaced by $X^r \sqsubseteq^? Y$. The intended meaning of X^r is that in a solution γ, $\gamma(X^r) = \{P \mid \forall r.P \in \gamma(X)\}$. Decomposition variables are created only when necessary, via the flattening rules; hence, not all variables have decomposition variables. For each role name r and a variable X, there can be at most one decomposition variable X^r, sometimes called the r-decomposition variable of X with X being a *parent* of X^r.

To enforce the intended meaning for a decomposition variable X^r, we create an **increasing subsumption** of the form $X \sqsubseteq^? \forall r.X^r$. These subsumptions are kept separately and are not subject to flattening.

However, an increasing subsumption alone does not ensure that, for every particle $\forall r.P$ in the substitution for X, P is in the substitution for X^r. For this reverse direction, we introduce a **decreasing rule**. We say that a substitution γ *obeys the decreasing rule* if and only if for every variable X in Γ, whenever X^r is defined, the following condition holds:

Decreasing rule. *If a particle of the form $\forall r.P \in \gamma(X)$, then $P \in \gamma(X^r)$.*

The decreasing rule is not expressible by a goal subsumption, but is enforced by our algorithm. Creation of new variables triggers making choices for them as described in the initial steps.

(b) Implicit Solver (Applying rules from Figure 1). After a flattening step we apply the rules of Implicit Solver are applied exhaustively to all goal subsumptions—whether flat or not yet flat—excluding start and increasing subsumptions.

Consider a **non-flat, unsolved** goal subsumption, $s = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? P$.

i. If P is of the form $\forall r.P'$, replace s with s^{-r}.
ii. If P is a variable X, replace s by the following set of goal subsumptions: $\{s^{-r} \mid r \in \mathbf{R}\}$, and if A_X true, add $C_1^A \sqcap \cdots \sqcap C_n^A \sqsubseteq^? A$ to the goal.

Figure 2: Flattening rules

The normalization process terminates in polynomial time with a polynomial increase in the size of the goal. Due to variable choices (such as $\top$, $\bot$, and A_X), it is non-deterministic. It produces a normalized goal consisting of the four sets of subsumptions in Figure 3. The following example illustrates the necessity of the decreasing rule for obtaining correct results.

Example 1. *Let our unification problem contain the goal subsumptions: $\forall rr.\bot \sqsubseteq^? Z, Z \sqsubseteq^? X, X \sqsubseteq^? \forall r.\bot$. The normalized goal is then:*

- *Solved subsumptions*: they are marked as *solved* by the rules of Implicit Solver and *deleted* from the goal.
- *Start subsumptions*: $\Gamma_{start} = \{X \sqsubseteq A \mid A_X \text{ is true}\} \cup \{X \sqsubseteq \bot \mid X \text{ is guessed to be a } \bot\text{-variable}\}$.
- *Increasing subsumptions*: $\{X \sqsubseteq^? \forall r.X^r \mid X^r \text{ is defined for } X\}$.
- *Flat subsumptions*: $\Gamma_{flat} = \{X_1 \sqcap \cdots \sqcap X_n \sqsubseteq^? Y \mid \text{all } X_1, \ldots, X_n, Y \text{ are variables}\}$ (unsolved subsumptions).

Figure 3: Normalized goal

start subsumption: $X^r \sqsubseteq^? \bot$; flat subsumption: $Z \sqsubseteq^? X$; *the increasing subsumptions are omitted.*

The first start subsumption forces X^r to be a $\bot$-variable, and thus, by the increasing subsumption $X \sqsubseteq^? \forall r.X^r$, $\forall r.\bot$ must appear in the substitution for X. By the flat subsumption, we know that $\forall r.\bot$ must also appear in the substitution for Z. However, there is nothing that can force $\bot$ into Z^r unless the decreasing rule is applied. If we do apply the decreasing rule, then $\bot$ is forced into Z^r—but in that case, we discover that the goal is not unifiable, because $\forall r.\bot \not\sqsubseteq \bot$.

Without the decreasing rule, the following substitution would be incorrectly accepted as a solution:
$Z \mapsto \{\forall r.\bot\}, Z^r \mapsto \top, X \mapsto \{\forall r.\bot\}, X^r \mapsto \{\bot\}$.

Next we prove the correctness of the normalization process.

Lemma 2. *(Completeness of normalization)*
Let Γ be a unification problem, and let Γ' be its normalized form obtained through the normalization process described above. If γ is a solution of Γ, then there exists a solution γ' of Γ' extending γ by assigning values to the newly introduced decomposition variables.

Proof. Since we assume γ is a ground unifier of Γ, it can guide the choices in the process of normalization.
The unifier γ determines the initial choices for variables in the following way. For a variable X:

1. if $\gamma(X) = \bot$, then X is a $\bot$-variable,
2. if $\gamma(X) = \top$, then X is a $\top$-variable,
3. if $A \in \gamma(X)$, then A_X is true otherwise A_X is false.

Moreover, if a decomposition variable X^r is introduced during the application of a flattening rule, we extend γ with the assignment $X^r \mapsto \{P \mid \forall r.P \in \gamma(X)\}$. Note that with this extension, γ satisfies both the increasing subsumptions and the decreasing rule. Also observe that if the set $\{P \mid \forall r.P \in \gamma(X)\}$ is empty, then X^r should be chosen as a $\top$-variable.

Given that the normalization process terminates, it suffices to prove the lemma for a single step. One can easily verify that the flattening rules preserve γ, when extended to the decomposition variables as described above. ☐

Notice that the decreasing rule is not mentioned in the formulation of Lemma 2. This is because if a substitution γ is a ground unifier of Γ, which is not yet extended to the decomposition variables, after extending γ by defining $\gamma(X^r) := \{P \mid \forall r.P \in \gamma(X)\}$ for each decomposition variable, the decreasing rule is obviously satisfied. On the contrary, the decreasing rule must be mentioned in the claim of soundness of the normalization procedure.

Lemma 3. *(Soundness of normalization)*
If Γ' is a normalized unification problem obtained from Γ, and if γ is a unifier of Γ' satisfying the decreasing rule for decomposition variables, then it is also a unifier of Γ.

Proof. (sketch) It is enough to consider the rules of Figure 2, because the reduction rules of Implicit Solver (Figure 1) are obviously sound. Let us assume that in the process of flattening we have obtained a subsumption: $s = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq P$. Let us assume that γ unifies this subsumption and it obeys the decreasing rule together with the increasing subsumptions for the decomposition variables. We consider which rule was applied to produce s. This rule must have been applied to some subsumption s', which we denote by $s' \to s$. We have to show that s' is also unified by γ. For the inductive argument, we assume that for every s'' such that $s' \to s''$, s'' is unified by γ. Inspection of the flattening rules then shows that s' is unified by γ. ☐

If the set of Γ_{flat} is empty, the algorithm terminates with *success*. In this case we can construct a unifier γ of the goal as shown in Figure 4.

For each variable X.

1. If the choice for X is $\top$, then define $\gamma(X) := \top$.
2. If the choice for X is $\bot$, then define $\gamma(X) := \bot$.
3. If A_X is true, then define $\gamma(X) := \{A\}$.
4. Repeat until there is no change:
   - If X^r is defined for X, then redefine $\gamma(X) := \gamma(X) \cup \{\forall r.\gamma(X^r)\}$.
5. If $\gamma(X)$ is still undefined, set $\gamma(X) := \top$.

Figure 4: Solution in the case of an empty set of flat subsumptions.

The initial assignments of $\bot$ and A to variables satisfy the start subsumptions. The repeat-step (4) will terminate with all increasing subsumptions satisfied. The solved subsumptions are satisfied by any substitution that respects the choices made when the rules of the Implicit Solver were applied; hence, γ is a unifier.

6 Shortcuts

If the normalization process has not terminated with *success* or *failure*, then the set of flat unsolved subsumptions is not empty. We proceed to the next stage, i.e. computing *shortcuts*.

Informally speaking, a shortcut is a pair of sets of variables from Γ, $(\mathcal{S}, \mathcal{P})$, where $\mathcal{S}$ is non-empty. The set of variables $\mathcal{S}$ is called the *main part* of the shortcut and $\mathcal{P}$–the prefix part.

Here one can see that the concept of shortcut for $\mathcal{FL}_\bot$ is a generalization of a similar concept in the context of $\mathcal{FL}_0$, since in the latter case one set of variables would be sufficient, namely the main part, (Borgwardt and Morawska 2012).

We will use a shortcut to distribute any particle over the flat subsumptions. The particle is placed in the assignment for variables in the main part and some prefixes of this particle should be assigned to the variables in the prefix part. Such a distribution of particles, viewed as a substitution that assigns $\top$ to all remaining variables, should unify all flat subsumptions in Γ_{flat}.

Example 2. *Let the set of flat subsumptions be defined as: $\Gamma_{flat} = \{Y \sqcap X \sqsubseteq^? X^r, X^r \sqsubseteq^? X\}$. Then an example of a shortcut is $(\mathcal{S}, \mathcal{P})$ where $\mathcal{S} = \{X\}$ and $\mathcal{P} = \{X^r, Y\}$. If we substitute X with a particle $\forall v.\bot$ (or $\forall v.A$) and assign certain $\bot$-particles in X^r and Y e.g. $\forall v'.\bot$, where $v' < v$ (or $v' \leq v$ in the case of A-particle), then obviously, Γ_{flat} is unified by such a substitution. On the other hand $(\{X^r\}, \{X\})$ is not a shortcut, because substituting X^r with a particle $\forall v.\bot$ and X with $\forall v'.\bot$ with $v' < v$ does not satisfy the subsumption $X^r \sqsubseteq^? X$. A good shortcut for this problem is also $(\{X, X^r\}, \emptyset)\}$, because if we substitute X and X^r with the same particle P, the flat subsumptions are satisfied.*

Thus the main property of a shortcut $(\mathcal{S}, \mathcal{P})$ is that if the variables in $\mathcal{S}$ are substituted with a particle P, and the variables in $\mathcal{P}$ with prefixes of P the flat subsumptions Γ_{flat} are satisfied. We say that a pair $(\mathcal{S}, \mathcal{P})$ satisfies a subsumption $Y_1 \sqcap \cdots \sqcap Y_n \sqsubseteq^? X \in \Gamma_{flat}$ if the following implications are true:

- if $X \in \mathcal{S}$, then there is Y_i such that $Y_i \in \mathcal{S} \cup \mathcal{P}$,
- if $X \in \mathcal{P}$, then there is Y_i such that, $Y_i \in \mathcal{P}$.

Let $S^{\bot}_{ini} = \{X \mid X \sqsubseteq^? \bot \in \Gamma_{start}\}$ and $S^{A}_{ini} = \{X \mid A_X \text{ is true }\}$.

Definition 2. *(Shortcut)*

A shortcut is a pair $(\mathcal{S}, \mathcal{P})$, where

1. *$\mathcal{S}$, $\mathcal{P}$ are disjoint sets of variables,*
2. *$\mathcal{S} \neq \emptyset$,*
3. *If $\mathcal{S} \cap S^{\bot}_{ini} \neq \emptyset$, then $\mathcal{S} \subseteq S^{\bot}_{ini}$ and $\mathcal{P} = \emptyset$,*
4. *$(\mathcal{S}, \mathcal{P})$ satisfies all flat subsumptions.*

We have two kinds of special shortcuts called *initial*.

- (the initial shortcut for bottom) $s^{\bot}_{ini} = (S^{\bot}_{ini}, \emptyset)$,
- (an initial shortcut for constant) $s^{A}_{ini} = (S^{A}_{ini}, \mathcal{P})$, where $\mathcal{P}$ is an arbitrary set such that $\mathcal{P} \subseteq S^{\bot}_{ini}$.

We do not specify the initial shortcut for the constant explicitly; any shortcut that meets the shortcut conditions will suffice. $\bot$-variables may occur in the prefix part of such a shortcut due to Definition 1 of a prefix. The concept $\bot$ is considered a prefix of a constant, whereas $\bot$ itself has no proper prefix.

Condition 1 of Definition 2 requires that the main part of a shortcut be disjoint from the prefix part. Otherwise, we could have a particle P and its prefix in the same variable X. This makes no sense, since P will be removed as redundant.

Condition 3 states that the $\bot$-variables occur in main parts of some shortcuts, for which the prefix part must be empty. This is because no particles can be proper prefixes of $\bot$.

A restricted resolving relation is defined between two shortcuts, excluding those whose main components involve bottom variables. We will subsequently extend this relation to cover such cases.

Definition 3. *(resolving relation) Let $s_1 = (\mathcal{S}_1, \mathcal{P}_1)$ and $s_2 = (\mathcal{S}_2, \mathcal{P}_2)$ be two shortcuts. We have two cases:*

1. *$\mathcal{S}_1 \not\subseteq \mathcal{S}_{ini}$. We say that s_1 is restricted-resolved with s_2 w.r.t. a role name r iff the following conditions are satisfied.*
   (a) *There is a decomposition variable X^r in $\mathcal{S}_1$ and for each decomposition variable Y^r in $\mathcal{S}_1$, its parent Y is in $\mathcal{S}_2$.*
   (b) *For each decomposition variable Y^r in $\mathcal{P}_1$, its parent Y is in $\mathcal{P}_2$.*
   (c) *For each variable Z in $\mathcal{S}_2$ such that Z^r is defined, Z^r is in $\mathcal{S}_1$.*
   (d) *For each variable Z in $\mathcal{P}_2$ such that Z^r is defined, Z^r is in $\mathcal{P}_1$.*
2. *If $\mathcal{S}_1 \subseteq \mathcal{S}_{ini}$ we require, in addition to the above conditions, that $\mathcal{P}_2 \subseteq \mathcal{S}_{ini}$.*

The resolving relation between s_1 and s_2 with respect to a role name r is denoted by $s_1 \xrightarrow{r} s_2$. Shortcuts with $\bot$-variables in their main part require special treatment since the prefix part of such shortcuts is empty ($\bot$ has no proper prefixes). In this case, $\mathcal{P}_2$ must contain only $\bot$-variables, as only such variables can serve as prefixes for particles of role depth 1.

We say that a shortcut $s = (\mathcal{S}, \mathcal{P})$ *depends on* a shortcut $s' = (\mathcal{S}', \mathcal{P}')$, denoted by $s \dashrightarrow s'$, if $\mathcal{P} = \mathcal{S}' \cup \mathcal{P}'$. In this case we also say that s' *supports* s.

For example, if $(\{X\}, \{Y, Z\})$ is a shortcut, then it depends on a shortcut $(\{Y\}, \{Z\})$ and this shortcut depends on $(\{Z\}, \emptyset)$. Obviously, every sequence of depending shortcuts has to terminate.

Definition 4. *A shortcut $(\mathcal{S}, \mathcal{P})$ is of height 0 iff $\mathcal{S}$ does not contain any decomposition variables.*

The term *shortcut of height* 0, as introduced in (Borgwardt and Morawska 2012), denotes a shortcut that, when used in constructing a solution, does not produce particles with greater role depth. The height 0 further indicates that such shortcuts are computed at the initial stage of the overall shortcut computation.

Definition 5. *A shortcut $s = (\mathcal{S}, \mathcal{P})$ is* valid *iff*

1. *it is a shortcut of height 0 or for every decomposition variable X^r in $\mathcal{S}$, s is resolved by a valid shortcut w.r.t. to the role r;*
2. *$\mathcal{P} = \emptyset$ or s depends on a valid shortcut.*

We will show that there is a unifier for an $\mathcal{FL}_\bot$-unification problem if and only if we can compute an acyclic graph of valid shortcuts, starting with the shortcuts of height 0 and extending it along the resolving relation with the set of already computed shortcuts, while containing the initial shortcuts.

To decide unification, our algorithm attempts to detect the existence of such an acyclic structure of shortcuts. If no such structure exists, the problem is not unifiable; otherwise, a solution can be computed from it.

6.1 Level-One Shortcuts

This section contains some lemmas that are important for the soundness proof in Section 9. A *level-one shortcut* is a shortcut of the form $(\mathcal{S}, \mathcal{P})$ such that $\mathcal{S} \not\subseteq S^{\bot}_{ini}$ and either $\mathcal{P} \subseteq S^{\bot}_{ini}$ or $\mathcal{P}$ is empty. The technical proofs of the lemmas in this section are given in the longer version of the paper.

Obviously, for each shortcut $(\mathcal{S}, \mathcal{P})$, if $\mathcal{P} \not\subseteq S^{\perp}_{ini}$ one can construct a level-one shortcut: $(\mathcal{S}', \mathcal{P}')$, namely $\mathcal{S}' = \mathcal{S} \cup \mathcal{P} \setminus S^{\perp}_{ini}$ and $\mathcal{P}' = \mathcal{P} \cap S^{\perp}_{ini}$. We call such shortcut the *level-one shortcut for* $(\mathcal{S}, \mathcal{P})$.

To prove the following lemmas, we assume an acyclic structure of valid shortcuts connected by the resolving relation. This acyclic structure will be produced by our algorithm, which computes all valid shortcuts.

Lemma 4. *Let $(\mathcal{S}_1, \mathcal{P}_1)$ and $(\mathcal{S}_2, \mathcal{P}_2)$ be two valid shortcuts such that $\mathcal{P}_1 = \mathcal{S}_2 \cup \mathcal{P}_2$. Then a shortcut $(\mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{P}_2)$ is also valid.*

Proof. The proof is by induction on the length of maximal path of resolving relations starting with $(\mathcal{S}_1, \mathcal{P}_1)$. ☐

We use the above lemma in proving the following fact about level-one shortcuts.

Lemma 5. *Let $(\mathcal{S}', \mathcal{P}')$ be the level-one shortcut for $(\mathcal{S}, \mathcal{P})$. If $(\mathcal{S}, \mathcal{P})$ is valid, then $(\mathcal{S}', \mathcal{P}')$ is also valid.*

Proof. The proof is by induction on the length of a path of supporting shortcuts starting with $(\mathcal{S}, \mathcal{P})$. ☐

Thanks to Lemma 4 we can also prove the following.

Lemma 6. *Let $(\mathcal{S}_1, \mathcal{P}_1)$ be resolved with $(\mathcal{S}_2, \mathcal{P}_2)$ w.r.t. role name r, $\mathcal{P}_1 \neq \emptyset, \mathcal{P}_2 \neq \emptyset$ and both these shortcuts are valid. Then either $\mathcal{P}_1$ does not contain any r-decomposition variable or there is a shortcut s_3 supporting $(\mathcal{S}_1, \mathcal{P}_1)$ and s_4 supporting $(\mathcal{S}_2, \mathcal{P}_2)$ such that s_3 is resolved with s_4 w.r.t. the role r.*

7 Computing Shortcuts

Here we present Algorithm 1 used to solve a normalized unification problem followed by explanations of its subprocedures.

Algorithm 1 Main

procedure MAIN(Γ) ▷ *Γ is a normalized unification problem defined w.r.t. current choices for variables*
  Var is a set of all variables in Γ
  $\mathcal{S}^{\perp}_{ini} = \{X \in \mathbf{Var} \mid X \sqsubseteq^{?} \perp \in \Gamma\}$
  For each constant A, $\mathcal{S}^{A}_{ini} = \{X \in \mathbf{Var} \mid X \sqsubseteq^{?} A \in \Gamma\}$
  $\Gamma_{flat} \leftarrow$ flat subsumptions of Γ
  $\mathfrak{S} \leftarrow$ ALLSHORTCUTS(Γ)
  if $\mathcal{S}^{\perp}_{ini} == \emptyset$ **then** ▷ *$\mathcal{FL}_0$-unification*
    if $s^{A}_{ini} \in \mathfrak{S}$, for each constant A **then**
      return success
  else if For each constant A, $\mathcal{S}^{A}_{ini} == \emptyset$ **then** ▷ *pure $\mathcal{FL}_{\perp}$-unification*
    if $s^{\perp}_{ini} \notin \mathfrak{S}$ **then**
      return failure
    else
      repeat
        $\mathfrak{S} \leftarrow$CHECKEXISTENCE($\mathfrak{S}$)
        $\mathfrak{S} \leftarrow$CHECKVALIDITY($\mathfrak{S}$)
      until there is no change in $\mathfrak{S}$
      if $s^{\perp}_{ini} \in \mathfrak{S}$ **then**
        return success
  else ▷ *Full $\mathcal{FL}_{\perp}$-unification*
    if $s^{\perp}_{ini} \notin \mathfrak{S}$ or for a constant A, $s^{A}_{ini} \notin \mathfrak{S}$ **then**
      return failure
    else
      repeat
        $\mathfrak{S} \leftarrow$CHECKEXISTENCE($\mathfrak{S}$)
        $\mathfrak{S} \leftarrow$CHECKVALIDITY($\mathfrak{S}$)
      until there is no change in $\mathfrak{S}$
      if $s^{\perp}_{ini} \in \mathfrak{S}$ and for each constant A, $s^{A}_{ini} \in \mathfrak{S}$ **then**
        return success
  return failure

Subprocedure ALLSHORTCUTS This procedure runs through all pairs of subsets of variables in **Var**, and for each one:

1. Identifies the candidate shortcuts of height 0 and verifies whether they satisfy the criteria of a shortcut (Definition 2), and of height 0 (Definition 4). The shortcuts that pass the checks are added to the set of computed shortcuts.

2. If there are no shortcuts of height 0 then the algorithms returns *failure*. Otherwise it enters a loop, computing next shortcuts based on the already computed ones. For each pair $(\mathcal{S}, \mathcal{P})$ of subsets of variables in **Var** that is not yet in the set of computed shortcuts it checks conditions of Definition 2, and verifies whether there exists an already computed shortcut that resolves $(\mathcal{S}, \mathcal{P})$, as defined in Definition 3.

   If a shortcut passes the checks, it is added to the set of computed shortcuts. Note that at this stage, shortcuts of heights greater than zero are also added. The loop terminates when no new shortcuts can be computed.

Subprocedure CHECKEXISTENCE This procedure takes the set of all shortcuts computed and checks if for each one, there is a supporting shortcut in the set. Hence it checks if for each shortcut s_1, there is a shortcut s_2 such that $s_1 \dashrightarrow s_2$. If no s_2 is found, s_1 is removed from the set of all computed shortcuts. The execution of this procedure ensures that the conditions of Definition 5 are satisfied.

Subprocedure CHECKVALIDITY After the deletions performed by CHECKEXISTENCE, this procedure verifies for each shortcut s_1 whether a shortcut of height 0, say s_0, is reachable from s_1 via the resolving relation, i.e., $s_1 \xrightarrow{*} s_0$. If s_1 is not resolved in this way, it is removed from the set of computed shortcuts.

8 Termination and Complexity

Theorem 1. *Let Γ be an $\mathcal{FL}_{\perp}$ unification problem. The algorithm terminates in at most exponential time in the size of Γ.*

Proof. (sketch) We mention only the most costly steps.

Line 6 is exponential, because there are exponentially many possible shortcuts (pairs of subsets of variables). This step is executed only once. Similarly, membership checks in lines 8, 18 and 21 may be exponential.

The repeat-loop (line 14, 24) causes deletions in the set of already computed shortcuts and hence it can be executed only at most exponentially many times. Each time it has to perform two sequential checks, each of them costing exponentially many steps.

Hence overall the time needed for the algorithm to terminate is exponential in the size of the problem. □

9 Soundness

Theorem 2. *Let Γ be the normalized unification problem and let the main algorithm on Γ terminate with success, then there exists a ground pure $\mathcal{FL}_\perp$-unifier γ of Γ.*

Proof. We assume that the algorithm terminated with success. Since the problem is normalized and the algorithm has run on Γ, we can assume that the set of start subsumptions and the set Γ_{flat} of flat subsumptions is not empty.

The algorithm has computed shortcuts $\mathfrak{S}$ in a special order, starting from the shortcuts of height 0. We can see this as if the algorithm has produced a directed acyclic graph defined on the set of all computed shortcuts. The graph is (V, E), where $V = \mathfrak{S}$, and E – the set of edges given by the resolving relation (Definition 3) between shortcuts.

The algorithm provides additional connections between the corresponding shortcuts in the following way: if $s = (\mathcal{S}, \mathcal{P})$ and $s' = (\mathcal{S}', \mathcal{P}')$, where $\mathcal{P} = \mathcal{S}' \cup \mathcal{P}'$, then we have an additional edge $s \dashrightarrow s'$. A dashed edge indicates where the $\perp$-particles in the prefix-part of a shortcut, should be created.

Now let us take a connected sub-graph of this construction containing the initial shortcuts. We show that a unifier can be constructed based on this graph. The construction begins with the substitution $\gamma_0 = \{X \mapsto \top \mid X \in \mathbf{Var}\}$ and proceeds by extending it with particles, until a unifier is obtained. For the description we use the following notions.

We say that a ground particle P of the role depth i *is created in a shortcut* $S = (\mathcal{S}, \mathcal{P})$ at step i, if an extended substitution γ_i is defined such that, for every variable $X \in \mathcal{S}$, $\gamma_i(X) = \gamma_{i-1}(X) \cup \{P\}$. If $X^r \in \mathcal{S}$, then we say that P is *active* in S at step i. *Step i* of the construction creates all the particles of the role depth i. Hence, the construction of a unifier consists of *deactivating* active particles of role depth $i-1$.

Invariant The invariant of the construction is that at any time the current substitution satisfies the flat subsumptions of Γ.

The invariant is obviously true for the initial substitution γ_0. The following condition has to be satisfied for the invariant to be true at any step.

Condition Let $S = (\mathcal{S}, \mathcal{P})$ be a shortcut and $S' = (\mathcal{S}', \mathcal{P}')$ be a shortcut corresponding to $\mathcal{P}$, i.e. $\mathcal{P} = \mathcal{S}' \cup \mathcal{P}'$.

We allow a particle P to be created in S at step i only if a prefix of P was already created in S' at a step smaller than i.

The above Condition forces us to construct a solution in a *bottom-up* way.

Steps of the construction

- (Step 0) In order to satisfy the start subsumptions, we create the bottom particle $\perp$ in the initial shortcut $s^{\perp}_{ini} = (\mathcal{S}^{\perp}_{ini}, \emptyset)$ and likewise, we initialize the A-variables in $s^{A}_{ini} = (\mathcal{S}^{A}_{ini}, \emptyset)$. Observe that such an extension of γ_0 to $\gamma_1 = \{[X \mapsto \gamma_0(X) \cup \{\perp\}] \mid X \in S^{\perp}_{ini}\} \cup \{[X \mapsto \gamma_0(X) \cup \{A\}] \mid X \in S^{A}_{ini}\}$ satisfies the invariant. Some particles become active at this step in $s^{\perp}_{ini}$ or s^{A}_{ini}. If there are no active particles, we terminate with success. The current substitution is a unifier.
- (Step 1) In order to *deactivate* the active particles, we follow the labeled arrows in the graph of shortcuts, representing how the shortcuts are resolved with each other.

  At this step one can choose to create arbitrarily particles of the form $\forall r.\perp$ in level-one shortcuts of the form $(\mathcal{S}, \emptyset)$, but only if no variable in $\mathcal{S}$ has r-decomposition variable defined. In other words, we can extend $\gamma(Z)$ with $\forall r.\perp$, where $Z \in \mathcal{S}$, only if Z^r is not defined. If Z^r is defined, the shortcut $(\mathcal{S}, \emptyset)$ cannot be used for creation of $\forall r.\perp$. Otherwise, the decreasing rule could be violated by such an extension.
- (Step i+1) Now assume that the substitution γ is defined for the variables, assigning to them sets of particles of the role depth at most i. Let us assume that $s = (\mathcal{S}, \mathcal{P})$ and a particle P of depth i is active in s. Hence P was created in s at step i and it is added to the substitution for a decomposition variable X^r in $\mathcal{S}$. ($P \in \gamma_i(X^r)$) Since the particles of depth $i+1$ are not yet created, the increasing subsumption: $X \sqsubseteq^? \forall r.X^r$ is not satisfied.

  The algorithm gives us an arrow from s to $s' = (\mathcal{S}', \mathcal{P}')$ with the label r, $s \xrightarrow{r} s'$. Hence $X \in \mathcal{S}'$. We create a particle $\forall r.P$ in s', extending the substitution to $\gamma_{i+1}(Y) = \gamma_i(Y) \cup \{\forall r.P\}$ for every $Y \in \mathcal{S}'$. In this way the particle P is deactivated in s and the increasing subsumption $X \sqsubseteq^? \forall r.X^r$ is satisfied w.r.t. to this particle ($\forall r.P \in \gamma_{i+1}(X)$ and $P \in \gamma_{i+1}(X^r)$).

□

The following example illustrates a graph of shortcuts. One can construct a unifier using the construction in the proof of Theorem 2.

Example 3. *Let the normalized goal be the following.*

Flat subsumptions: $\{Y^r \sqsubseteq^? Z, U \sqcap Y \sqsubseteq^? Z^s\}$,
Start subsumptions: $\{Y^r \sqsubseteq^? A, U^{rs} \sqsubseteq^? \perp\}$.

The algorithm yields the following shortcuts: $s^{\perp}_{ini} = (\{U^{rs}\}, \emptyset)$, $s^{A}_{ini} = (\{Y^r\}, \{U^{rs}\})$, $s_1 = (\{U^r, Y^r\}, \{U^{rs}\})$, $s_2 = (\{U, Y, Z^s\}, \emptyset)$, $s_3 = (\{Z\}, \{Y^r, U^{rs}\})$. *Notice that* s_3 *is of height* 0 *and it depends on* s^{A}_{ini}. s^{A}_{ini} *is resolved by* s_2. *The graph of resolving relations between the shortcuts is presented below.*

$$\begin{tikzcd}
s^{\perp}_{ini} \arrow[r, "s"] & s_1 \arrow[r, "r"] \arrow[l, dashed, bend left] & s_2 \arrow[r, "s"] & s_3 \arrow[dlll, dashed, bend left] \\
s^{A}_{ini} \arrow[urr, "r"'] & & &
\end{tikzcd}$$

Next we prove that Condition of the construction can be satisfied at each step and hence the construction in the proof of Theorem 2 is always possible.

Lemma 7. *Let P be a particle that should be created in $s = (\mathcal{S}, \mathcal{P})$ at some step i.*

Then the prefixes of P have been created at previous steps at the variables in $\mathcal{P}$, so that Condition of the construction is satisfied.

Proof. (sketch) The proof proceeds by induction on i, where i is the index of a step in the construction. This index also corresponds to the role depth of every particle created at step i. For the proof, we use Lemma 5 and Lemma 6 to demonstrate how the prefixes are formed. □

The next lemma shows that the construction in the proof of Theorem 2 terminates.

Lemma 8. *Let Γ be a normalized unification problem. If the algorithm terminates with success, then the construction of a unifier terminates.*

Proof. The reason is that the graph of shortcuts is acyclic, so every path in this graph has finite length—no longer than the total number of shortcuts. Therefore, the construction of a solution terminates after creating only finitely many particles, each with a role depth at most exponential in the size of Γ. □

10 Completeness

Theorem 3. *Let Γ be a normalized unification problem and let γ be a ground unifier of Γ obeying the decreasing rule. Then the main algorithm will terminate with success.*

Proof. The proof demonstrates how the solution γ guarantees a non-failing run of the algorithm. We assume that γ is reduced w.r.t. to the properties of $\mathcal{FL}_\perp$ and is minimal in size; that is, the sum of the role depths of the particles in the range of γ is minimal.

We identify the set of $\perp$-variables as $\mathcal{S}^{\perp}_{ini} = \{X \in \mathbf{Var} \mid \gamma(X) = \perp\}$ and the set of variables containing constant A as $\mathcal{S}^{A}_{ini} = \{X \in \mathbf{Var} \mid A \in \gamma(X)\}$

If all these sets are empty, we can construct a solution sending all variables to $\top$.

Otherwise, we identify shortcuts defined by γ in the following way: for a given particle P in the range of γ, there is a shortcut, $(\mathcal{S}_P, \mathcal{P}_P)$ where:

$\mathcal{S}_P = \{X \in \mathbf{Var} \mid P \in \gamma(X)\}$,

$\mathcal{P}_P = \{Y \in \mathbf{Var} \mid P' \in \gamma(Y), P'$ is a prefix of $P\}$.

For a shortcut $(\mathcal{S}_P, \mathcal{P}_P)$, $\mathcal{P}_P$ satisfies the conditions of Definition 2. The algorithm calls the sub-procedure ALLSHORTCUTS which computes all possible shortcuts defined in this way. First let us notice that there will be shortcuts of height 0, namely the shortcuts defined for the particles of maximal role depth in γ. Hence, the set of computed shortcuts will not be empty.

Now it is easy to see that since γ satisfies increasing subsumptions and obeys the decreasing rule, then in the set of all shortcuts defined by γ, there will be shortcuts properly resolved between themselves and thus they will be computed by ALLSHORTCUTS and will not be deleted during the run of the algorithm.

Hence, for each particle P in the range of γ, a shortcut defined by γ will be computed and not deleted. Now we consider the particular cases determined by the initial shortcuts. Here we justify completeness only in the most interesting case, i.e. when $\mathcal{S}^{\perp}_{ini}$ is empty. In this case we do not have any $\perp$-variables. If there are no $\perp$-particles in the range of γ, then γ is an $\mathcal{FL}_0$-unifier. There are no prefixes for the A-particles, hence in each shortcut, the prefix part is empty. The initial shortcut $(\mathcal{S}^{A}_{ini}, \emptyset)$ is computed by ALLSHORTCUTS. This clearly shows that the algorithm is a generalization of the unification algorithm for $\mathcal{FL}_0$.

If there are $\perp$-particles in the range of γ, we show that there is a smaller unifier γ', which contradicts our assumption about minimality of γ.

For that, let us consider a $\perp$-particle P_{min} of minimal role depth assigned to a variable by γ. We know that it is not $\perp$. Now, consider the shortcut $(\mathcal{S}_{min}, \mathcal{P}_{min})$ defined by γ for P_{min}. Let us notice that in this case $\mathcal{P}_{min}$ must be empty, since otherwise γ assigns some prefixes of P_{min} to some variables, which contradicts minimality of P_{min}.

Now assume w.l.o.g. that the particle P_{min} is of the form $\forall r.P'$. It implies that all variables in $\mathcal{S}_{min}$ do not have any r-decomposition variables defined. Otherwise, since γ obeys the decreasing rule, we would have a shortcut for P' and P' is smaller than P_{min}.

Hence, $(\mathcal{S}_{min}, \emptyset)$ is a resolved shortcut that has no r-incoming edges in the graph of shortcuts. Notice also that the variables in $\mathcal{S}_{min}$ are not $\perp$-variables, since they contain P_{min} under the substitution γ.

We obtain a different substitution γ' by creating $\perp$ in $(\mathcal{S}_{min}, \emptyset)$. This changes all variables in $\mathcal{S}_{min}$ to $\perp$-variables. Now, we create new substitution γ' using the old structure of shortcuts defined by γ. Notice that now Condition 3 of Definition 2 is not satisfied, because $\perp$-variables from $\mathcal{S}_{min}$ may occur in the main part of some shortcuts.

In order to satisfy the increasing subsumptions, we use the same paths of resolving relation as in the case of γ starting with the shortcut $(\mathcal{S}_{min}, \emptyset)$ as the new $s^{\perp}_{ini}$.

The construction from the proof of Theorem 2 goes through, although the new substitution γ' may be not reduced. After reductions, γ' is strictly smaller than γ, since at least the $\perp$-particles assigned by γ to variables in $\mathcal{S}_{min}$, in γ' are replaced by $\perp$. Other particles might also be reduced by introducing $\perp$ in γ'.

The construction is illustrated in Example 4.

The conclusion of this argument is that if a solution to a problem uses $\perp$ in its particles, the set of $\perp$-variables must be non-empty or else the solution is not minimal. □

Example 4. *Let the goal be* $\Gamma = \{X \sqsubseteq^? \forall r.Y, Y \sqsubseteq^? \forall r.X, X \sqsubseteq^? \forall r.A\}$ *Let the normalized* Γ *be* $\{X^r \sqsubseteq^? Y, Y^r \sqsubseteq^? X, X^r \sqsubseteq^? A\}$ *with increasing subsumptions omitted. Let* $X^r \sqsubseteq^? A$ *be a start subsumption.*

Now let a solution γ *be:* $[X^r \mapsto A \sqcap \forall r.\bot, Y^r \mapsto \forall r.\bot, X \mapsto \forall r.A \sqcap \forall rr.\bot, Y \mapsto \forall rr.\bot]$.

One can see that γ *is a solution. It satisfies the increasing subsumptions and the decreasing rule.*

For $\forall r.\bot$, γ *defines shortcut* $(\{X^r, Y^r\}, \emptyset)$ *and this is* $(\mathcal{S}_{min}, \emptyset)$ *from the above argument. If* $\bot$ *is created in* $(\{X^r, Y^r\}, \emptyset)$*, the structure can be reused to generate* γ'.

$[X^r \mapsto A \sqcap \bot, Y^r \mapsto \bot, X \mapsto \forall r.A \sqcap \forall r.\bot, Y \mapsto \forall r.\bot]$.

After removing redundant particles, we have a smaller solution with $\bot$*-variables:* X^r *and* Y^r.

11 Conclusions

The procedure presented in this paper shows that the $\mathcal{FL}_\bot$-unification is in the ExpTime-class of problems. Theorem 1 in (Baader and Fernández Gil 2022) further shows that our problem is also ExpTime-hard.

The algorithm described here is a generalization of the unification algorithm that can be extracted from (Borgwardt and Morawska 2012) for the description logic $\mathcal{FL}_0$. A more direct exposition of that algorithm may be found in (Morawska et al. 2025). The result extends the area of description logics for which unification is known, i.e. the description logic $\mathcal{EL}$ and $\mathcal{FL}_0$.

One can attempt to extend similar techniques to the logic $\mathcal{FLE}$, which combines $\mathcal{FL}_0$ with $\mathcal{EL}$. However, the main challenge remains the logic $\mathcal{ALC}$, which adds negation to the constructors and is therefore a full Boolean description logic. Very little is known about unification in $\mathcal{ALC}$, except that it is undecidable in an extension of this logic (Wolter and Zakharyaschev 2008).

Our aim is to implement the algorithm presented in this paper and perform experiments that may lead to useful applications in the area of knowledge representation.

Acknowledgments

This research is part of the project No 2022/47/P/ST6/03196 co-funded by the National Science Centre and the European Union Framework Programme for Research and Innovation Horizon 2020 under the Marie Skłodowska-Curie grant agreement No. 945339. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

Baader, F., and Fernández Gil, O. 2022. Restricted unification in the description logic $\mathcal{FL}_\bot$. In M. Cerna, D., and Morawska, B., eds., *Proceedings of the 36th International Workshop on Unification (UNIF 2022).*

Baader, F., and Küsters, R. 2001. Unification in a description logic with transitive closure of roles. In Goble, C. A.; McGuinness, D. L.; Möller, R.; and Patel-Schneider, P. F., eds., *Working Notes of the 2001 International Description Logics Workshop (DL-2001), Stanford, CA, USA, August 1-3, 2001*, volume 49 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Baader, F., and Küsters, R. 2002. Unification in a description logic with inconsistency and transitive closure of roles. In Horrocks, I., and Tessaris, S., eds., *Proceedings of the 2002 International Workshop on Description Logics (DL2002), Toulouse, France, April 19-21, 2002*, volume 53 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Baader, F., and Narendran, P. 2001. Unification of concept terms in description logics. *Journal of Symbolic Computation* 31(3):277–305.

Borgwardt, S., and Morawska, B. 2012. Finding finite Herbrand models. In Bjørner, N., and Voronkov, A., eds., *Proc. of the 18th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-18)*, volume 7180 of *Lecture Notes in Computer Science*, 138–152. Springer.

Morawska, B., and Marzec, D. 2025. Solving unification in the description logic $\mathcal{FL}_\bot$. *arXiv preprint arXiv:2405.00912*. https://arxiv.org/abs/2405.00912.

Morawska, B.; Kost, S.; Marzec, D.; and Henne, M. 2025. FILO: unication solver for FL0. In *UNIF 2025 - Informal Proceedings of the 39th International Workshop on Unification.*

Morawska, B. 2021. Unification in the description logic $\mathcal{FL}_\bot$. In Homola, M.; Ryzhikov, V.; and Schmidt, R. A., eds., *Proceedings of the 34th International Workshop on Description Logics (DL 2021) part of Bratislava Knowledge September (BAKS 2021), Bratislava, Slovakia, September 19th to 22nd, 2021*, volume 2954 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Wolter, F., and Zakharyaschev, M. 2008. Undecidability of the unification and admissibility problems for modal and description logics. *ACM Trans. Comput. Log.* 9(4):25:1–25:20.