TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Dismatching and Local Disunification in $\mathcal{EL}$

Franz Baader and Stefan Borgwardt and Barbara Morawska

LTCS-Report 15-03

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Abstract

Unification in Description Logics has been introduced as a means to detect redundancies in ontologies. We try to extend the known decidability results for unification in the Description Logic $\mathcal{EL}$ to disunification since negative constraints on unifiers can be used to avoid unwanted unifiers. While decidability of the solvability of general $\mathcal{EL}$-disunification problems remains an open problem, we obtain NP-completeness results for two interesting special cases: *dismatching problems*, where one side of each negative constraint must be ground, and *local* solvability of disunification problems, where we restrict the attention to solutions that are built from so-called atoms occurring in the input problem. More precisely, we first show that dismatching can be reduced to local disunification, and then provide two complementary NP-algorithms for finding local solutions of (general) disunification problems.

1 Introduction

Description logics (DLs) [9] are a family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application areas, but their most notable success so far is the adoption of the DL-based language OWL [20] as standard ontology language for the semantic web. DLs allow their users to define the important notions (classes, relations) of the domain using concepts and roles; to state constraints on the way these notions can be interpreted using terminological axioms; and to deduce consequences such as subsumption (subclass) relationships from the definitions and constraints. The expressivity of a particular DL is determined by the constructors available for building concepts.

The DL $\mathcal{EL}$, which offers the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has drawn considerable attention in the last decade since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even with respect to expressive terminological axioms [15]. On the other hand, though quite inexpressive, $\mathcal{EL}$ is used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1] For these reasons, the most recent OWL version, OWL 2, contains the profile OWL 2 EL,[2] which is based on a maximally tractable extension of $\mathcal{EL}$ [10].

Unification in Description Logics was introduced in [4] as as a novel inference service that can be used to detect redundancies in ontologies. It is shown there that unification in the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ in that existential restriction

[1] http://www.ihtsdo.org/snomed-ct/

[2] http://www.w3.org/TR/owl2-profiles/

is replaced by value restriction ($\forall r.C$), is ExpTime-complete. The applicability of this result was not only hampered by this high complexity, but also by the fact that $\mathcal{FL}_0$ is not used in practice to formulate ontologies.

In contrast, as mentioned above, $\mathcal{EL}$ is employed to build large biomedical ontologies for which detecting redundancies is a useful inference service. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe head injury* as

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Head\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_finding} \sqcap \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}). \tag{2}$$

Formally, these two concepts are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names $\mathsf{Head\_injury}$ and $\mathsf{Severe\_finding}$ as variables, and substituting the first one by $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and the second one by $\exists \mathsf{severity}.\mathsf{Severe}$. In this case, we say that the concepts are unifiable, and call the substitution that makes them equivalent a *unifier*. In [1], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity than unification in $\mathcal{FL}_0$: the decision problem for $\mathcal{EL}$ is NP-complete. The main idea underlying the proof of this result is to show that any solvable $\mathcal{EL}$-unification problem has a local unifier, i.e., a unifier built from a polynomial number of so-called atoms determined by the unification problem. However, the brute-force "guess and then test" NP-algorithm obtained from this result, which guesses a local substitution and then checks (in polynomial time) whether it is a unifier, is not useful in practice. We thus developed a goal-oriented unification algorithm for $\mathcal{EL}$, which is more efficient since nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. Another option for obtaining a more efficient unification algorithm is a translation to satisfiability in propositional logic (SAT): in [2] it is shown how a given $\mathcal{EL}$-unification problem Γ can be translated in polynomial time into a propositional formula whose satisfying valuations correspond to the local unifiers of Γ.

Intuitively, a unifier of two $\mathcal{EL}$ concepts proposes definitions for the concept names that are used as variables: in our example, we know that, if we define $\mathsf{Head\_injury}$ as $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\mathsf{Severe\_finding}$ as $\exists \mathsf{severity}.\mathsf{Severe}$, then the two concepts (1) and (2) are equivalent w.r.t. these definitions. Of course, this example was constructed such that the unifier (which is actually local) provides sensible definitions for the concept names used as variables. In general, the existence of a unifier only says that there is a structural similarity between the two concepts. The developer that uses unification as a tool for finding redundancies in an ontology or between two different ontologies needs to inspect the unifier(s) to see whether the definitions it suggests really make sense. For example, the substitution that replaces $\mathsf{Head\_injury}$ by $\mathsf{Patient} \sqcap \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\mathsf{Severe\_finding}$ by $\mathsf{Patient} \sqcap \exists \mathsf{severity}.\mathsf{Severe}$ is also a local unifier, which however does not make sense. Unfortunately, even small unification problems like the one in our example can have too many local unifiers for manual inspection. In [12] we propose to restrict the attention to so-called minimal unifiers, which form a subset of all local unifiers. In our example, the nonsensical unifier is indeed not minimal. In general, however, the restriction to minimal unifiers may preclude interesting local unifiers. In addition, as shown in [12], computing minimal unifiers is actually harder than computing local unifiers (unless the polynomial hierarchy collapses). In the present paper, we propose disunification as a more direct approach for avoiding local unifiers that do not make sense. In addition to positive constraints (requiring equivalence or subsumption between concepts), a disunification problem may also contain negative constraints (preventing equivalence or subsumption between concepts). In our example, the nonsensical unifier can be avoided by adding the dissubsumption constraint

$$\mathsf{Head\_injury} \not\sqsubseteq^? \mathsf{Patient} \tag{3}$$

to the equivalence constraint (1) $\equiv^?$ (2).

Unification and disunification in DLs is actually a special case of unification and disunification modulo equational theories (see [4] and [1] for the equational theories respectively corresponding to $\mathcal{FL}_0$ and $\mathcal{EL}$). Disunification modulo equational theories has, e.g., been investigated in [16, 17]. It is well-known in unification theory that for effectively finitary equational theories, i.e., theories for which finite complete sets of unifiers can effectively be computed, disunification can be reduced to unification: to decide whether a disunification problem has a solution, one computes a finite complete set of unifiers of the equations and then checks whether any of the unifiers in this set also solves the disequations. Unfortunately, for $\mathcal{FL}_0$ and $\mathcal{EL}$, this approach is not feasible since the corresponding equational theories have unification type zero [1, 4], and thus finite complete sets of unifiers need not even exist. Nevertheless, it was shown in [6] that the approached used in [4] to decide unification (reduction to language equations, which are then solved using tree automata) can be adapted such that it can also deal with disunification. This yields the result that disunification in $\mathcal{FL}_0$ has the same complexity (ExpTime-complete) as unification.

For $\mathcal{EL}$, going from unification to disunification appears to be more problematic. In fact, the main reason for unification to be decidable and in NP is locality: if the problem has a unifier then it has a local unifier. We will show that disunification in $\mathcal{EL}$ is not local in this sense by providing an example of a disunification problem that has a solution, but no local solution. Decidability and complexity of disunification in $\mathcal{EL}$ remains an open problem, but we provide partial solutions that are of interest in practice. On the one hand, we investigate *dismatching problems*, i.e., disunification problems where the negative constraints are dissubsumptions $C \not\sqsubseteq^? D$ for which either C or D is ground (i.e., does not contain a variable).

Table 1: Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restr.	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} := \{x \mid \exists y.(x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

Note that the dissubsumption (3) from above actually satisfies this restriction since $\mathsf{Patient}$ is not a variable. We prove that (general) solvability of dismatching problems can be reduced to *local disunification*, i.e., the question whether a given $\mathcal{EL}$-disunification problem has a *local* solution, which shows that dismatching in $\mathcal{EL}$ is NP-complete. On the other hand, we develop two specialized algorithms to solve local disunification problems that extend the ones for unification [1, 2]: a goal-oriented algorithm that reduces the amount of nondeterministic guesses necessary to find a local solution, as well as a translation to SAT. The reason we present two kinds of algorithms is that, in the case of unification, they have proved to complement each other well in first evaluations [11]: the goal-oriented algorithm needs less memory and finds minimal solutions faster, while the SAT reduction generates larger data structures, but outperforms the goal-oriented algorithm on unsolvable problems.

2 Subsumption and dissubsumption in $\mathcal{EL}$

The syntax of $\mathcal{EL}$ is defined based on two sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* and *role names*, respectively. *Concept terms* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for $r \in \mathsf{N_R}$), and *top* ($\top$). An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is extended to concept terms as shown in the semantics column of Table 1.

A concept term C is *subsumed* by a concept term D (written $C \sqsubseteq D$) if for every interpretation $\mathcal{I}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We write a *dissubsumption* $C \not\sqsubseteq D$ to abbreviate the fact that $C \sqsubseteq D$ does not hold. The two concept terms C and D are *equivalent* (written $C \equiv D$) if $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e. they are always interpreted as the same set. Note that we use "$=$" to denote *syntactic* equality between concept terms, whereas "$\equiv$" denotes semantic equivalence.

Since conjunction is interpreted as intersection, we can treat $\sqcap$ as a commutative and associative operator, and thus dispense with parentheses in nested conjunctions.

An *atom* is a concept name or an existential restriction. Hence, every concept term C is a conjunction of atoms or $\top$. We call the atoms in this conjunction the *top-level atoms* of C. Obviously, C is equivalent to the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. An atom is *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ with $A \in \mathsf{N_C}$.

Subsumption in $\mathcal{EL}$ is decidable in polynomial time [8] and can be checked by recursively comparing the top-level atoms of the two concept terms.

Lemma 1 ([1]). *For two atoms C, D, we have $C \sqsubseteq D$ iff $C = D$ is a concept name or $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq D'$. If C, D are concept terms, then $C \sqsubseteq D$ iff for every top-level atom D' of D there is a top-level atom C' of C such that $C' \sqsubseteq D'$.*

We obtain the following contrapositive formulation characterizing dissubsumption.

Lemma 2. *For two concept terms C, D, we have $C \not\sqsubseteq D$ iff there is a top-level atom D' of D such that for all top-level atoms C' of C it holds that $C' \not\sqsubseteq D'$.*

In particular, $C \not\sqsubseteq D$ is characterized by the existence of a top-level atom D' of D for which $C \not\sqsubseteq D'$ holds. By further analyzing the structure of atoms, we obtain the following.

Lemma 3. *Let C, D be two atoms. Then we have $C \not\sqsubseteq D$ iff either*

1. *C or D is a concept name and $C \neq D$; or*
2. *$D = \exists r.D'$, $C = \exists s.C'$, and $r \neq s$; or*
3. *$D = \exists r.D'$, $C = \exists r.C'$, and $C' \not\sqsubseteq D'$.*

3 Disunification

As described in the introduction, we now partition the set $\mathsf{N_C}$ into a set of *(concept) variables* ($\mathsf{N_v}$) and a set of *(concept) constants* ($\mathsf{N_c}$). A concept term is *ground* if it does not contain any variables. We define a quite general notion of disunification problems that is similar to the equational formulae used in [17].

Definition 4. A *disunification problem* Γ is a formula built from subsumptions of the form $C \sqsubseteq^? D$, where C and D are concept terms, using the logical connectives $\wedge$, $\vee$, and $\neg$. We use equations $C \equiv^? D$ to abbreviate $(C \sqsubseteq^? D) \wedge (D \sqsubseteq^? C)$, disequations $C \not\equiv^? D$ for $\neg(C \sqsubseteq^? D) \vee \neg(D \sqsubseteq^? C)$, and dissubsumptions $C \not\sqsubseteq^? D$ instead of $\neg(C \sqsubseteq^? D)$. A *basic disunification problem* is a conjunction of subsumptions and dissubsumptions. A *dismatching problem* is a basic disunification problem in which all dissubsumptions $C \not\sqsubseteq^? D$ are such that either C or D is ground. Finally, a *unification problem* is a conjunction of subsumptions.

To define the semantics of disunification problems, we now fix a *finite signature* $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ and assume that all disunification problems contain only concept terms constructed over the symbols in Σ. A *substitution* σ maps every variable in Σ to a ground concept term constructed over the symbols of Σ. This mapping can be extended to all concept terms (over Σ) in the usual way. A substitution σ *solves* a subsumption $C \sqsubseteq^? D$ if $\sigma(C) \sqsubseteq \sigma(D)$; it *solves* $\Gamma_1 \wedge \Gamma_2$ if it solves both Γ_1 and Γ_2; it solves $\Gamma_1 \vee \Gamma_2$ if it solves Γ_1 or Γ_2; and it solves $\neg\Gamma$ if it does not solve Γ. A substitution that solves a given disunification problem is called a *solution* of this problem. A disunification problem is *solvable* if it has a solution.

In contrast to unification, in disunification it does make a difference whether or not solutions may contain variables from $\mathsf{N_v} \cap \Sigma$ or additional symbols from $(\mathsf{N_C} \cup \mathsf{N_R}) \setminus \Sigma$ [16]. In the context of the application sketched in the introduction, restricting solutions to ground terms over the signature of the ontology to be checked for redundancy is appropriate: since a solution σ is supposed to provide definitions for the variables in Σ, it should not use the variables themselves to define them; moreover, definitions that contain newly generated symbols would be meaningless to the user.

Reduction to basic disunification problems We will consider only basic disunification problems in the following. The reason is that there is a straightforward NP-reduction from solvability of arbitrary disunification problems to solvability of basic disunification problems. In this reduction, we view all subsumptions occurring in the disunification problem as propositional variables and guess a satisfying valuation of the resulting propositional formula. It then suffices to check solvability of the basic disunification problem obtained as the conjunction of all subsumptions evaluated to true and the negations of all subsumptions evaluated to false. Since the problems considered in the following sections are all NP-complete, the restriction to basic disunification problems does not affect our complexity results. In the following, we thus restrict the attention to basic disunification problems, which we simply call *disunification problems* and consider them to be sets of subsumptions and dissubsumptions.

Reduction to flat disunification problems We further simplify our analysis by considering *flat* disunification problems, which means that they may only contain *flat* dissubsumptions of the form $C_1 \sqcap \dots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \dots \sqcap D_m$ for flat atoms $C_1, \dots, C_n, D_1, \dots, D_m$ with $m, n \geq 0$,[3] and *flat* subsumptions of the form $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? D_1$ for flat atoms $C_1, \dots, C_n, D_1$ with $n \geq 0$.

The restriction to flat disunification problems is without loss of generality: to flatten concept terms, one can simply introduce new variables and equations to abbreviate subterms [1]. Moreover, a subsumption of the form $C \sqsubseteq^? D_1 \sqcap \dots \sqcap D_m$

[3]Recall that the empty conjunction is $\top$.

is equivalent to $C \sqsubseteq^? D_1, \ldots, C \sqsubseteq^? D_m$. Any solution of a disunification problem Γ can be extended to a solution of the resulting flat disunification problem Γ', and conversely every solution of Γ' also solves Γ.

This flattening procedure also works for unification problems. However, dismatching problems cannot without loss of generality be restricted to being flat since the introduction of new variables to abbreviate subterms may destroy the property that one side of each dissubsumption is ground (see also Section 4).

For solving flat unification problems, it has been shown that it suffices to consider so-called local solutions [1], which are restricted to use only the atoms occurring in the input problem. We extend this notion to disunification as follows.

Let Γ be a flat disunification problem. We denote by At the set of all (flat) atoms occurring as subterms in Γ, by Var the set of variables occurring in Γ, and by $\mathsf{At}_{\mathsf{nv}} := \mathsf{At} \setminus \mathsf{Var}$ the set of *non-variable atoms* of Γ. Let $S\colon \mathsf{Var} \to 2^{\mathsf{At}_{\mathsf{nv}}}$ be an *assignment (for* Γ*)*, i.e. a function that assigns to each variable $X \in \mathsf{Var}$ a set $S_X \subseteq \mathsf{At}_{\mathsf{nv}}$ of non-variable atoms. The relation $>_S$ on Var is defined as the transitive closure of $\{(X, Y) \in \mathsf{Var}^2 \mid Y \text{ occurs in an atom of } S_X\}$. If this defines a strict partial order, i.e. $>_S$ is irreflexive, then S is called *acyclic*. In this case, we can define the substitution σ_S inductively along $>_S$ as follows: if X is minimal, then

$$\sigma_S(X) := \bigsqcap_{D \in S_X} D;$$

otherwise, assume that $\sigma_S(Y)$ is defined for all $Y \in \mathsf{Var}$ with $X > Y$, and define

$$\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D).$$

It is easy to see that the concept terms $\sigma_S(D)$ are ground and constructed from the symbols of Σ, and hence σ_S is a valid candidate for a solution of Γ according to Definition 4.

Definition 5. Let Γ be a flat disunification problem. A substitution σ is called *local* if there exists an acyclic assignment S for Γ such that $\sigma = \sigma_S$. The disunification problem Γ is *locally solvable* if it has a local solution, i.e. a solution that is a local substitution. *Local disunification* is the problem of checking flat disunification problems for local solvability.

Note that assignments and local solutions are defined only for *flat* disunification problems.

Obviously, local disunification is decidable in NP: We can guess an assignment S, and check it for acyclicity and whether the induced substitution solves the disunification problem in polynomial time. The corresponding complexity lower bound follows from NP-hardness of (local) solvability of unification problems in $\mathcal{EL}$ [1].

Fact 6. *Deciding local solvability of flat disunification problems in $\mathcal{EL}$ is* NP*-complete.*

It has been shown that unification in $\mathcal{EL}$ is *local* in the sense that the equivalent flattened problem has a local solution iff the original problem is solvable, and hence (general) solvability of unification problems in $\mathcal{EL}$ can be decided in NP [1]. The next example shows that disunification in $\mathcal{EL}$ is *not local* in this sense.

Example 7. Consider the flat disunification problem

$$\Gamma := \{X \sqsubseteq^? B,\ A \sqcap B \sqcap C \sqsubseteq^? X,\ \exists r.X \sqsubseteq^? Y,\ \top \not\sqsubseteq^? Y,\ Y \not\sqsubseteq^? \exists r.B\}$$

with concept variables X, Y and concept constants A, B, C. The substitution σ with $\sigma(X) := A \sqcap B \sqcap C$ and $\sigma(Y) := \exists r.(A \sqcap C)$ is a solution of Γ. For σ to be local, the atom $\exists r.(A \sqcap C)$ would have to be of the form $\sigma(D)$ for a non-variable atom D occurring in Γ. But the only candidates for D are $\exists r.X$ and $\exists r.B$, none of which satisfy $\exists r.(A \sqcap C) = \sigma(D)$.

We show that Γ cannot have another solution that is local. Assume to the contrary that Γ has a local solution γ. We know that $\gamma(Y)$ cannot be $\top$ since γ must solve the first dissubsumption. Furthermore, none of the constants A, B, C can be a top-level atom of $\gamma(Y)$ since this would contradict the third subsumption. That leaves only the non-variable atoms $\exists r.\gamma(X)$ and $\exists r.B$, which are ruled out by the last dissubsumption since both $\gamma(X)$ and B are subsumed by B.

The decidability and complexity of general solvability of disunification problems is still open. In the following, we first consider the special case of solving dismatching problems, for which we show a similar result as for unification: every dismatching problem can be polynomially reduced to a flat problem that has a local solution iff the original problem is solvable. The main difference is that this reduction is nondeterministic. In this way, we reduce dismatching to local disunification. We then provide two different NP-algorithms for the latter problem by extending the rule-based unification algorithm from [1] and adapting the SAT encoding of unification problems from [2]. These algorithms are more efficient than the brute-force "guess and then test" procedure on which our argument for Fact 6 was based.

4 Reducing dismatching to local disunification

Our introduction of dismatching problems was motivated in part by the work on *matching* in description logics, where similar restrictions are imposed on unification problems [3, 7, 22]. In particular, the matching problems for $\mathcal{EL}$ investigated in [3] are similar to our dismatching problems in that there subsumptions are restricted to ones where one side is ground. Another motivation comes from our experience that matching problems already suffice to formulate most of the negative constraints one may want to put on unification problems, as described in the introduction.

As mentioned in Section 3, we cannot restrict our attention to flat dismatching problems without loss of generality. Instead, the nondeterministic algorithm we present in the following reduces any dismatching problem Γ to a flat *disunification* problem Γ' with the property that local solvability of Γ' is equivalent to the solvability of Γ. Since the algorithm takes at most polynomial time in the size of Γ, this shows that dismatching in $\mathcal{EL}$ is NP-complete. For simplicity, we assume that the subsumptions and the non-ground sides of the dissubsumptions have already been flattened using the approach mentioned in the previous section. This retains the property that all dissubsumptions have one ground side and does not affect the solvability of the problem.

Our procedure exhaustively applies a set of rules to the (dis)subsumptions in a dismatching problem (see Figures 1 and 2). In these rules, $C_1, \dots, C_n$ and $D_1, \dots, D_m$ are atoms. The rule Left Decomposition includes the special case where the left-hand side of $\mathfrak{s}$ is $\top$, in which case $\mathfrak{s}$ is simply removed from the problem. Note that at most one rule is applicable to any given (dis)subsumption. The choice which (dis)subsumption to consider next is don't care nondeterministic, but the choices in the rules Right Decomposition and Solving Left-Ground Dissubsumptions are don't know nondeterministic.

Algorithm 8. Let Γ_0 be a dismatching problem. We initialize $\Gamma := \Gamma_0$. While any of the rules of Figures 1 and 2 is applicable to any element of Γ, choose one such element and apply the corresponding rule. If any rule application fails, then return "failure".

Note that each rule application takes only polynomial time in the size of the chosen (dis)subsumption. In particular, subsumptions between ground atoms can be checked in polynomial time [8].

Lemma 9. *Every run of Algorithm 8 terminates in time polynomial in the size of the input problem.*

Proof. Let $\Gamma_0, \dots, \Gamma_k$ be the sequence of disunification problems created during a run of the algorithm, i.e.

- Γ_0 is the input dismatching problem;
- for all j, $0 \le j \le k-1$, Γ_{j+1} is the result of successfully applying one rule to a (dis)subsumption in Γ_j; and
- either no rule is applicable to any element of Γ_k, or a rule application to a (dis)subsumption in Γ_k failed.

Right Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \cdots \sqcap D_m$ if $m = 0$ or $m > 1$, and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ are atoms.
Action: If $m = 0$, then *fail*. Otherwise, choose an index $i \in \{1, \ldots, m\}$ and replace $\mathfrak{s}$ by $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_i$.

Left Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$ if $n = 0$ or $n > 1$, $C_1, \ldots, C_n$ are atoms, and D is a non-variable atom.
Action: Replace $\mathfrak{s}$ by $C_1 \not\sqsubseteq^? D$, $\ldots$, $C_n \not\sqsubseteq^? D$.

Atomic Decomposition:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? D$ if C and D are non-variable atoms.
Action: Apply the first case that matches $\mathfrak{s}$:

a) if C and D are ground and $C \sqsubseteq D$, then *fail*;

b) if C and D are ground and $C \not\sqsubseteq D$, then remove $\mathfrak{s}$ from Γ;

c) if C or D is a constant, then remove $\mathfrak{s}$ from Γ;

d) if $C = \exists r.C'$ and $D = \exists s.D'$ with $r \neq s$, then remove $\mathfrak{s}$ from Γ;

e) if $C = \exists r.C'$ and $D = \exists r.D'$, then replace $\mathfrak{s}$ by $C' \not\sqsubseteq^? D'$.

Figure 1: Decomposition rules

We prove that k is polynomial in the size of Γ_0 by measuring the size of (dis)subsumptions by the function c defined as follows:

$$c(C \not\sqsubseteq^? D) := c(C \sqsubseteq^? D) := |C| \cdot |D|,$$

where $|C|$ is the size of the concept term C; the latter is measured in the number of symbols it takes to write down C, where we count each concept name as one symbol, and "$\exists r.$" is also one symbol. Note that we always have $|C| \geq 1$ since C must contain at least one concept name or $\top$, and thus also $c(\mathfrak{s}) \geq 1$ for any (dis)subsumption $\mathfrak{s}$. We now define the size $\mathsf{Cl}(\Gamma)$ of a disunification problem Γ as the sum of the sizes $c(\mathfrak{s})$ for all $\mathfrak{s} \in \Gamma$ to which a rule is applicable.

Since $c(\Gamma_0)$ is obviously polynomial in the size of Γ_0, it now suffices to show that $c(\Gamma_j) > c(\Gamma_{j+1})$ holds for all j, $0 \leq j \leq k-1$. To show this, we consider the rule that was applied to $\mathfrak{s} \in \Gamma_j$ in order to obtain Γ_{j+1}:

- **Right Decomposition:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \ldots D_m$ and we must have $m > 1$ since we assumed that the rule application was successful. Thus, we get $|C_1 \sqcap \cdots \sqcap C_n| \cdot |D_1 \sqcap \cdots \sqcap D_m| > |C_1 \sqcap \cdots \sqcap C_n| \cdot |D_i|$ for every choice of $i \in \{1, \ldots, m\}$, and thus $c(\Gamma_j) > c(\Gamma_{j+1})$.

Flattening Right-Ground Dissubsumptions:

Condition: This rule applies to $\mathfrak{s} = X \not\sqsubseteq^? \exists r.D$ if X is a variable and D is ground and is not a concept name. **Action:** Introduce a new variable X_D and replace $\mathfrak{s}$ by $X \not\sqsubseteq^? \exists r.X_D$ and $D \sqsubseteq^? X_D$.

Flattening Left-Ground Subsumptions:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m \sqsubseteq^? X$ if $m > 0$, X is a variable, $C_1, \ldots, C_n$ are flat ground atoms, and $\exists r_1.D_1, \ldots, \exists r_m.D_m$ are non-flat ground atoms. **Action:** Introduce new variables $X_{D_1}, \ldots, X_{D_m}$ and replace $\mathfrak{s}$ by $D_1 \sqsubseteq^? X_{D_1}$, $\ldots$, $D_m \sqsubseteq^? X_{D_m}$ and $C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.X_{D_1} \sqcap \cdots \sqcap \exists r_m.X_{D_m} \sqsubseteq^? X$.

Solving Left-Ground Dissubsumptions:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ if X is a variable and $C_1, \ldots, C_n$ are ground atoms. **Action:** Choose one of the following options: • Choose a constant $A \in \Sigma$ and replace $\mathfrak{s}$ by $X \sqsubseteq^? A$. If $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq A$, then *fail*. • Choose a role $r \in \Sigma$, introduce a new variable Z, replace $\mathfrak{s}$ by $X \sqsubseteq^? \exists r.Z$, $C_1 \not\sqsubseteq^? \exists r.Z$, $\ldots$, $C_n \not\sqsubseteq^? \exists r.Z$, and immediately apply Atomic Decomposition to each of these dissubsumptions.

Figure 2: Flattening and solving rules

- **Left Decomposition:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$ and, if $n = 0$, then $c(\Gamma_j) = c(\Gamma_{j+1}) + c(\mathfrak{s}) \geq c(\Gamma_{j+1}) + 1 > c(\Gamma_{j+1})$. Otherwise, we have $n > 1$, and thus

$$\begin{aligned} |C_1 \sqcap \cdots \sqcap C_n| \cdot |D| &= (|C_1| + \cdots + |C_n| + (n-1)) \cdot |D| \\ &> |C_1| \cdot |D| + \cdots + |C_n| \cdot |D|. \end{aligned}$$

- **Atomic Decomposition:** It suffices to consider Case e) since Case a) is impossible and the other cases are trivial. Then $\mathfrak{s} = \exists r.C' \not\sqsubseteq^? \exists r.D'$, and we get $|\exists r.C'| \cdot |\exists r.D'| = (|C'| + 1) \cdot (|D'| + 1) > |C'| \cdot |D'|$.

- **Flattening Right-Ground Dissubsumptions:** Then $\mathfrak{s} = X \not\sqsubseteq^? \exists r.D$ is replaced by $X \not\sqsubseteq^? \exists r.X_D$ and $D \sqsubseteq^? X_D$. To the dissubsumption, no further rule is applicable, and hence it does not count towards $c(\Gamma_j)$. Regarding the subsumption, we have $|X| \cdot |\exists r.D| = |D| + 1 > |D| = |D| \cdot |X_D|$.

- **Flattening Left-Ground Subsumptions:** Then the subsumption $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m \sqsubseteq^? X$ and only to the subsumptions $D_1 \sqsubseteq^? X_{D_1}$, $\ldots$, $D_m \sqsubseteq^? X_{D_m}$ this flattening rule may be applicable again. But we have

$$
\begin{aligned}
|C_1 \sqcap & \cdots \sqcap C_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m| \cdot |X| \\
&= |C_1| + \cdots + |C_n| + |\exists r_1.D_1| + \cdots + |\exists r_m.D_m| + (n + m - 1) \\
&\geq |\exists r_1.D_1| + \cdots + |\exists r_m.D_m| \\
&> |D_1| + \cdots + |D_m| \\
&= |D_1| \cdot |X_{D_1}| + \cdots + |D_m| \cdot |X_{D_m}|.
\end{aligned}
$$

- **Solving Left-Ground Dissubsumptions:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ and to a generated subsumption of the form $X \sqsubseteq^? A$ or $X \sqsubseteq^? \exists r.Z$ no further rule is applicable. If $n = 0$, then no further dissubsumptions are generated, and thus $c(\Gamma_j) > c(\Gamma_{j+1})$. Otherwise, we denote by $|\mathfrak{s}_i|$ the size of the dissubsumption resulting from applying Atomic Decomposition to $C_i \not\sqsubseteq^? \exists r.Z$, $1 \leq i \leq n$, where we consider this number to be 0 if the dissubsumption was simply discarded (cf. Cases b)–d) of Atomic Decomposition).

  If $|\mathfrak{s}_i| = 0$, we obtain $|C_i| \geq 1 > 0 = |\mathfrak{s}_i|$. But also in Case e), we have $C_i = \exists r.C_i'$, and thus $|C_i| = |C_i'| + 1 = |C_i'| \cdot |Z| + 1 > |\mathfrak{s}_i|$. Hence, we get

$$
\begin{aligned}
|C_1 \sqcap \cdots \sqcap C_n| \cdot |X| &= |C_1| + \cdots + |C_n| + (n - 1) \\
&\geq |C_1| + \cdots + |C_n| \\
&> |\mathfrak{s}_1| + \cdots + |\mathfrak{s}_n|.
\end{aligned}
\qquad \square
$$

Note that the Solving rule for left-ground dissubsumptions is not limited to non-flat dissubsumptions, and thus the algorithm completely eliminates all left-ground dissubsumptions from Γ. It is also easy to see that, if the algorithm is successful, then the resulting disunification problem Γ is flat. We now prove that this nondeterministic procedure is correct in the following sense.

Lemma 10. *The dismatching problem Γ_0 is solvable iff there is a successful run of Algorithm 8 such that the resulting flat disunification problem Γ has a local solution.*

Proof. For soundness (i.e. the if direction), let σ be the local solution of Γ and consider the run of Algorithm 8 that produced Γ. It is easy to show by induction on the reverse order in which the rules have been applied that σ solves all subsumptions that have been considered. Indeed, this follows from simple applications of Lemmata 1–3 and the properties of subsumption. This implies that σ is also a solution of Γ_0.

Showing completeness (i.e. the only-if direction) is a little more involved. Let γ be a solution of Γ_0. We guide the rule applications of Algorithm 8 and extend γ to the newly introduced variables in such a way to maintain the invariant that "*γ solves all (dis)subsumptions of* Γ". This obviously holds after the initialization $\Gamma := \Gamma_0$. Afterwards, we will use γ to define a local solution of Γ.

Consider a (dis)subsumption $\mathfrak{s} \in \Gamma$ that is solved by γ and to which one of the rules of Figures 1 and 2 is applicable. We make a case distinction on which rule is to be applied:

- **Right Decomposition:** Then $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \cdots \sqcap D_m$ for $m \neq 1$. Since $\gamma(C_1 \sqcap \cdots \sqcap C_n) \not\sqsubseteq \gamma(D_1 \sqcap \cdots \sqcap D_m)$, by applying Lemma 2 twice, we can find an index $i \in \{1, \ldots, m\}$ such that $\gamma(C_1 \sqcap \cdots \sqcap C_n) \not\sqsubseteq \gamma(D_i)$. Thus, we can choose this index in the rule application in order to satisfy the invariant.

- **Left Decomposition:** Then $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$, where $n \neq 1$ and D is a non-variable atom. This means that $\gamma(D)$ is also an atom, an thus by Lemma 2 we know that $\gamma(C_i) \not\sqsubseteq \gamma(D)$ holds for all $i \in \{1, \ldots, n\}$, as required.

- **Atomic Decomposition:** Then $\mathfrak{s}$ is of the form $C \not\sqsubseteq^? D$ for two non-variable atoms C and D. Since $\gamma(C) \not\sqsubseteq \gamma(D)$, Case a) cannot apply. If one of the Cases b)–d) applies, then $\mathfrak{s}$ is simply removed from Γ and there is nothing to show. Otherwise, we have $D = \exists r.D'$ and $C = \exists r.C'$, and the new dissubsumption $C' \not\sqsubseteq^? D'$ is added to Γ. Moreover, we have $\gamma(C) = \exists r.\gamma(C')$ and $\gamma(D) = \exists r.\gamma(D')$, and thus by Lemma 3 we know that $\gamma(C') \not\sqsubseteq \gamma(D')$.

- **Flattening Right-Ground Dissubsumptions:** Then $\mathfrak{s}$ is of the form $X \not\sqsubseteq^? \exists r.D$. By defining $\gamma(X_D) := D$, γ solves $X \not\sqsubseteq^? \exists r.X_D$ and $D \sqsubseteq^? X_D$.

- **Flattening Left-Ground Subsumptions:** Then the subsumption $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m \sqsubseteq^? X$, where all $D_1, \ldots, D_m$ are ground. If we extend γ by defining $\gamma(X_{D_i}) := D_i$ for all $i \in \{1, \ldots, m\}$, then this obviously satisfies the new subsumptions $D_1 \sqsubseteq^? X_{D_1}, \ldots, D_m \sqsubseteq^? X_{D_m}$, and $C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.X_{D_1} \sqcap \cdots \sqcap \exists r_m.X_{D_m} \sqsubseteq^? X$ by our assumption that γ solves $\mathfrak{s}$.

- **Solving Left-Ground Dissubsumptions:** Then the dissubsumption $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$, where X is a variable and $C_1, \ldots, C_n$ are ground atoms. By Lemma 2, there must be a ground top-level atom D of $\gamma(X)$ such that $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq D$, i.e. $C_1 \not\sqsubseteq D, \ldots, C_n \not\sqsubseteq D$. If D is a concept constant, we can choose this in the rule application since we know that $\gamma(X) \sqsubseteq D$. Otherwise, we have $D = \exists r.D'$. By extending γ to $\gamma(Z) := D'$, we ensure that $X \sqsubseteq^? \exists r.Z$, $C_1 \not\sqsubseteq^? \exists r.Z, \ldots C_n \not\sqsubseteq^? \exists r.Z$ are solved by γ. The remaining claim follows as for the Atomic Decomposition rule above.

Once no more rules can be applied, we obtain a flat disunification problem Γ of which the extended substitution γ is a (possibly non-local) solution. To obtain a local solution, we denote by At, Var, and $\mathsf{At}_{\mathsf{nv}}$ the sets as defined in Section 3 and define the assignment S induced by γ as in [2]:

$$S_X := \{D \in \mathsf{At}_{\mathsf{nv}} \mid \gamma(X) \sqsubseteq \gamma(D)\},$$

for all (old and new) variables $X \in \mathsf{Var}$. It was shown in [2] that S is acyclic and the substitution σ_S solves all subsumptions in Γ.[4] Furthermore, it is easy to show that $\gamma(C) \sqsubseteq \sigma_S(C)$ holds for all concept terms C.

Since Γ contains no left-ground dissubsumptions anymore, it remains to show that σ_S solves all remaining right-ground dissubsumptions in Γ and all flat dissubsumptions created by an application of the Flattening rule for right-ground dissubsumptions. Consider first any flat right-ground dissubsumption $X \not\sqsubseteq^? D$ in Γ. We have already shown that $\gamma(X) \not\sqsubseteq D$ holds. Since $\gamma(X) \sqsubseteq \sigma_S(X)$, by the transitivity of subsumption $\sigma_S(X) \sqsubseteq D$ cannot hold, and thus also σ_S solves the dissubsumption.

Consider now a dissubsumption $X \not\sqsubseteq^? \exists r.X_D$ that was created by an application of the Flattening rule for right-ground dissubsumptions to $X \not\sqsubseteq^? \exists r.D$. By the same argument as above, from $\gamma(X) \not\sqsubseteq \exists r.D$ we can derive that $\sigma_S(X) \not\sqsubseteq \exists r.D$ holds. We now show that $\sigma_S(X_D) \sqsubseteq D$ holds, which implies that $\sigma_S(\exists r.X_D) \sqsubseteq \exists r.D$, and thus by the transitivity of subsumption it cannot be the case that $\sigma_S(X) \sqsubseteq \sigma_S(\exists r.X_D)$, which concludes the proof by showing that σ_S solves Γ.

We show that $\sigma_S(X_C) \sqsubseteq C$ holds for all variables X_C for which a subsumption $C \sqsubseteq^? X_C$ was introduced by a Flattening rule. We prove this claim by induction on the *role depth* of C, which is the maximum nesting depth of existential restrictions occurring in it. Let $C_1, \ldots, C_n$ be the top-level atoms of C. Then Γ contains a flat subsumption $C'_1 \sqcap \cdots \sqcap C'_n \sqsubseteq^? X_C$, where $C_i = C'_i$ if C_i is flat, and $C_i = \exists r.D_i$ and $C'_i = \exists r.X_{D_i}$ otherwise. Since the role depth of each such D_i is strictly smaller than that of C, by induction we know that $\sigma_S(X_{D_i}) \sqsubseteq D_i$, and thus $\sigma_S(C'_1 \sqcap \cdots \sqcap C'_n) \sqsubseteq C_1 \sqcap \cdots \sqcap C_n = C$ by Lemma 1. Furthermore, for all $i \in \{1, \ldots, n\}$ we have $\gamma(X_C) = C \sqsubseteq C_i = \gamma(C'_i)$ and $C'_i \in \mathsf{At}_{\mathsf{nv}}$. Thus, $C'_i \in S_{X_C}$ by the definition of S. The definition of σ_S now yields that $\sigma_S(X_C) \sqsubseteq \sigma_S(C'_1 \sqcap \cdots \sqcap C'_n) \sqsubseteq C$ (see Section 3). $\square$

The disunification problem of Example 7 is in fact a dismatching problem. The rule Solving Left-Ground Dissubsumptions can be used to replace $\top \not\sqsubseteq^? Y$ with $Y \sqsubseteq^? \exists r.Z$. The presence of the new atom $\exists r.Z$ makes the solution σ introduced in Example 7 local.

Together with Fact 6 and the NP-hardness of unification in $\mathcal{EL}$ [1], this shows the following complexity result.

Theorem 11. *Deciding solvability of dismatching problems in $\mathcal{EL}$ is* NP*-complete.*

[4] More precisely, it was shown that γ induces a satisfying valuation of a SAT problem, which in turn induces the solution σ_S above. For details, see [2] or Sections 6.1 and 6.2.

5 A goal-oriented algorithm for local disunification

In this section, we present an algorithm for local disunification that is based on transformation rules. Basically, to solve the subsumptions, this algorithm uses the rules of the goal-oriented algorithm for unification in $\mathcal{EL}$ [1, 13], which produces only local unifiers. Since any local solution of the disunification problem is a local unifier of the subsumptions in the problem, one might think that it is then sufficient to check whether any of the produced unifiers also solves the dissubsumptions. This would not be complete, however, since the goal-oriented algorithm for unification does *not* produce *all* local unifiers. For this reason, we have additional rules for solving the dissubsumptions. Both rule sets contain (deterministic) *eager* rules that are applied with the highest priority, and *nondeterministic* rules that are only applied if no eager rule is applicable. The goal of the eager rules is to enable the algorithm to detect obvious contradictions as early as possible in order to reduce the number of nondeterministic choices it has to make.

Let now Γ_0 be the flat disunification problem for which we want to decide local solvability, and let the sets At, Var, and $\mathsf{At}_{\mathsf{nv}}$ be defined as in Section 3. We assume without loss of generality that the dissubsumptions in Γ_0 have only a single atom on the right-hand side. If this is not the case, it can easily be achieved by exhaustive application of the nondeterministic rule Right Decomposition (see Figure 1) without affecting the complexity of the overall procedure.

Starting with Γ_0, the algorithm maintains a current disunification problem Γ and a current acyclic assignment S, which initially assigns the empty set to all variables. In addition, for each subsumption or dissubsumption in Γ, it maintains the information on whether it is *solved* or not. Initially, all subsumptions of Γ_0 are unsolved, except those with a variable on the right-hand side, and all dissubsumptions in Γ_0 are unsolved, except those with a variable on the left-hand side and a non-variable atom on the right-hand side.

Subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ and dissubsumptions of the form $X \not\sqsubseteq^? D$, for a non-variable atom D, are called *initially solved*. Intuitively, they only specify constraints on the assignment S_X. More formally, this intuition is captured by the process of *expanding* Γ w.r.t. the variable X, which performs the following actions:

- every initially solved subsumption $\mathfrak{s} \in \Gamma$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ is expanded by adding the subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? E$ to Γ for every $E \in S_X$, and
- every initially solved dissubsumption $X \not\sqsubseteq^? D \in \Gamma$ is expanded by adding $E \not\sqsubseteq^? D$ to Γ for every $E \in S_X$.

A (non-failing) application of a rule of our algorithm does the following:

- it solves exactly one unsolved subsumption or dissubsumption,
- it may extend the current assignment S by adding elements of $\mathsf{At}_{\mathsf{nv}}$ to some set S_X,
- it may introduce new flat subsumptions or dissubsumptions built from elements of At, and
- it keeps Γ expanded w.r.t. all variables X.

Subsumptions and dissubsumptions are only added by a rule application or by expansion if they are not already present in Γ. If a new subsumption or dissubsumption is added to Γ, it is marked as unsolved, unless it is initially solved (because of its form). Solving subsumptions and dissubsumptions is mostly independent, except for expanding Γ, which can add new unsolved subsumptions and dissubsumptions at the same time, and may be triggered by solving a subsumption or a dissubsumption.

The rules dealing with subsumptions are depicted in Figures 3 and 4. Note that several rules may be applicable to the same subsumption.

The rules for solving dissubsumptions are presented in Figures 5 and 6. In the rule Local Extension, the left-hand side of $\mathfrak{s}$ may be a variable, and then $\mathfrak{s}$ is of the form $Y \not\sqsubseteq^? X$. This dissubsumption is not initially solved, because X is not a non-variable atom.

Algorithm 12. Let Γ_0 be a flat disunification problem. We initialize $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all variables $X \in \mathsf{Var}$. While Γ contains an unsolved subsumption or dissubsumption, do the following:

1. **Eager rule application:** If eager rules are applicable to some unsolved subsumption or dissubsumption $\mathfrak{s}$ in Γ, apply an arbitrarily chosen one to $\mathfrak{s}$. If the rule application fails, return "failure".
2. **Nondeterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption or dissubsumption in Γ. If one of the nondeterministic rules (Figures 4 and 6) applies to $\mathfrak{s}$, choose one and apply it. If none of these rules apply to $\mathfrak{s}$ or the rule application fails, then return "failure".

Once all (dis)subsumptions in Γ are solved, return the substitution σ_S that is induced by the current assignment.

Eager Ground Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if $\mathfrak{s}$ is ground. **Action:** The rule application fails if $\mathfrak{s}$ does not hold. Otherwise, $\mathfrak{s}$ is marked as *solved.*

Eager Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is an index $i \in \{1, \ldots, n\}$, such that $C_i = D$ or $C_i = X \in \mathsf{Var}$ and $D \in S_X$. **Action:** The application of the rule marks $\mathfrak{s}$ as *solved.*

Eager Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is an index $i \in \{1, \ldots, n\}$, such that $C_i = X \in \mathsf{Var}$ and $\{C_1, \ldots, C_n\} \setminus \{X\} \subseteq S_X$. **Action:** The application of the rule adds D to S_X. It this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. X and $\mathfrak{s}$ is marked as *solved.*

Figure 3: Eager rules for subsumptions

As with Algorithm 8, the choice which (dis)subsumption to consider next and which eager rule to apply is don't care nondeterministic, while the choice of which nondeterministic rule to apply and the choices inside the rules are don't know nondeterministic. Each of these latter choices may result in a different solution σ_S.

5.1 Termination

Lemma 13. *Every run of Algorithm 12 terminates in time polynomial in the size of* Γ_0.

Proof. Each rule application solves one subsumption or dissubsumption. We show that there are only polynomially many subsumptions and dissubsumptions produced during a run of the algorithm, and thus there can be only polynomially many rule applications during one run of the algorithm.

A new subsumption or dissubsumption may be created only by an application of Decomposition, Eager Left Decomposition, or Eager Atomic Decomposition, and then it is of the form $C \sqsubseteq^? D$ or $C \not\sqsubseteq^? D$, where $C, D \in \mathsf{At}$. Obviously, there are only polynomially many such (dis)subsumptions.

Now, we consider (dis)subsumptions created by expanding Γ. These subsumptions or dissubsumptions have the following forms, where $D, E \in \mathsf{At}_{\mathsf{nv}}$:

1. $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? E$, for $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ in Γ,

2. $E \not\sqsubseteq^? D$, for $X \not\sqsubseteq^? D$ in Γ.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D \in \Gamma$, if there is at least one index $i \in \{1, \ldots, n\}$ with $C_i = \exists s.C$.
Action: The application of the rule chooses such an index i, adds $C \sqsubseteq^? D$ to Γ, expands Γ w.r.t. D if D is a variable, and marks $\mathfrak{s}$ as *solved*.

Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is at least one index $i \in \{1, \ldots, n\}$ with $C_i \in \mathsf{Var}$.
Action: The application of the rule chooses such an index i and adds D to S_{C_i}. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and $\mathfrak{s}$ is marked as *solved*.

Figure 4: Nondeterministic rules for subsumptions

Dissubsumptions of the second type are also of the form described above. For the subsumptions of the first type, note that $C_1 \sqcap \cdots \sqcap C_n$ is either the left-hand side of a subsumption from the original problem Γ_0, or it was created by a Decomposition rule, in which case we have $n = 1$. Thus, there can also be at most polynomially many subsumptions of the first type.

Finally, each rule application takes at most polynomial time. □

5.2 Soundness

Assume that a run of the algorithm terminates with success, i.e. all subsumptions and dissubsumptions are solved. Let $\hat{\Gamma}$ be the set of all subsumptions and dissubsumptions produced by this run, S be the final assignment, and σ_S the induced substitution (see Section 3). To show that σ_S solves $\hat{\Gamma}$, and hence also Γ_0, we use induction on the following order on (dis)subsumptions.

Definition 14. Consider any (dis)subsumption $\mathfrak{s}$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? C_{n+1}$ or $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? C_{n+1}$ in $\hat{\Gamma}$.

- We define $m(\mathfrak{s}) := (m_1(\mathfrak{s}), m_2(\mathfrak{s}))$, where
  - $m_1(\mathfrak{s}) := \emptyset$ if $\mathfrak{s}$ is ground; otherwise, $m_1(\mathfrak{s}) := \{X_1, \ldots, X_m\}$, where $\{X_1, \ldots, X_m\}$ is the multiset of all variables occurring in the concept terms $C_1, \ldots, C_n, C_{n+1}$.
  - $m_2(\mathfrak{s}) := |\mathfrak{s}|$, where $|\mathfrak{s}|$ is the size of $\mathfrak{s}$, i.e. the number of symbols in $\mathfrak{s}$.
- The strict partial order $\succ$ on such pairs is the lexicographic order, where the second components are compared w.r.t. the usual order on natural numbers, and the first components are compared w.r.t. the multiset extension of $>_S$ [5].

Eager Top Solving:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? \top \in \Gamma$. **Action:** The rule application fails.

Eager Left Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D \in \Gamma$ if $n = 0$ or $n > 1$, and $D \in \mathsf{At}_{\mathsf{nv}}$. **Action:** The application of the rule marks $\mathfrak{s}$ as *solved* and, for each $i \in \{1, \ldots, n\}$, adds $C_i \not\sqsubseteq^? D$ to Γ and expands Γ w.r.t. C_i if C_i is a variable.

Eager Atomic Decomposition:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? D \in \Gamma$ if $C, D \in \mathsf{At}_{\mathsf{nv}}$. **Action:** The application of the rule applies the first case that matches $\mathfrak{s}$: a) if C and D are ground and $C \sqsubseteq D$, then the rule application fails; b) if C and D are ground and $C \not\sqsubseteq D$, then $\mathfrak{s}$ is marked as *solved*; c) if C or D is a concept name, then $\mathfrak{s}$ is marked as *solved*; d) if $C = \exists r.C'$ and $D = \exists s.D'$ with $r \neq s$, then $\mathfrak{s}$ is marked as *solved*; e) if $C = \exists r.C'$ and $D = \exists r.D'$, then $C' \not\sqsubseteq^? D'$ is added to Γ, Γ is expanded w.r.t. C' if C' is a variable and D' is not a variable, and $\mathfrak{s}$ is marked as *solved*.

Figure 5: Eager rules for dissubsumptions

- We extend $\succ$ to $\hat{\Gamma}$ by setting $\mathfrak{s}_1 \succ \mathfrak{s}_2$ iff $m(\mathfrak{s}_1) \succ m(\mathfrak{s}_2)$.

Since multiset extensions and lexicographic products of well-founded strict partial orders are again well-founded [5], $\succ$ is a well-founded strict partial order on $\hat{\Gamma}$.

Lemma 15. *σ_S is a solution of $\hat{\Gamma}$, and thus also of its subset Γ_0.*

Proof. Consider a (dis)subsumption $\mathfrak{s} \in \hat{\Gamma}$ and assume that σ_S solves all $\mathfrak{s}' \in \hat{\Gamma}$ with $\mathfrak{s}' \prec \mathfrak{s}$. Since $\mathfrak{s}$ is solved, either it has been solved by a rule application or it was initially solved.

- If $\mathfrak{s}$ was solved by a rule application, we consider which rule was applied.
  - **Eager Ground Solving:** Then $\mathfrak{s}$ is ground and holds under any substitution.
  - **Eager Solving:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ and $\sigma_S(D)$ occurs on the top-level of $\sigma_S(C_1) \sqcap \cdots \sqcap \sigma_S(C_n)$, hence σ_S solves the subsumption.
  - **(Eager) Extension:** Then $\mathfrak{s} = X \sqcap C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ for a variable X and $D \in S_X$. By the definition of σ_S, we have $\sigma_S(X) \sqsubseteq \sigma_S(D)$ and thus σ_S solves $\mathfrak{s}$.

Local Extension:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? X \in \Gamma$ if $X \in \mathsf{Var}$. **Action:** The application of the rule chooses $D \in \mathsf{At}_{\mathsf{nv}}$ and adds D to S_X. If this makes S cyclic, the rule application fails. Otherwise, the new dissubsumption $C \not\sqsubseteq^? D$ is added to Γ, Γ is expanded w.r.t. X, Γ is expanded w.r.t. C if C is a variable, and $\mathfrak{s}$ is marked as *solved*.

Figure 6: Nondeterministic rule for dissubsumptions

- **Decomposition:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D$ with $C_i = \exists s.C$ for some $i \in \{1, \ldots, n\}$ and we have $\mathfrak{s}' = C \sqsubseteq^? D \in \hat{\Gamma}$. We know that $\mathfrak{s}' \prec \mathfrak{s}$, because $m_1(\mathfrak{s}') \leq m_1(\mathfrak{s})$ and $m_2(\mathfrak{s}') < m_2(\mathfrak{s})$. By induction, we get $\sigma_S(C) \sqsubseteq \sigma_S(D)$, and hence σ_S solves $\mathfrak{s}$.
- **Eager Top Solving:** This rule cannot have been applied since we assumed the run to be successful.
- **Eager Left Decomposition:** Then either $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$ with $n > 1$, or $\mathfrak{s} = \top \not\sqsubseteq^? D$, for a non-variable atom D. In the former case, σ_S solves $\mathfrak{s}$ by Lemma 2. In the latter case, for each $i \in \{1, \ldots, n\}$ we have $\mathfrak{s}_i := C_i \not\sqsubseteq^? D \in \hat{\Gamma}$. Notice that $m_1(\mathfrak{s}) \geq m_1(\mathfrak{s}_i)$ and $m_2(\mathfrak{s}) > m_2(\mathfrak{s}_i)$ and hence $\mathfrak{s} \succ \mathfrak{s}_i$. Thus, by induction we have that $\sigma_S(C_i) \not\sqsubseteq \sigma_S(D)$. By Lemma 2, $\sigma_S(C_1) \sqcap \cdots \sqcap \sigma_S(C_n) \not\sqsubseteq \sigma_S(D)$.
- **Eager Atomic Decomposition:** Then $\mathfrak{s} = C \not\sqsubseteq^? D$, where C and D are non-variable atoms. Since we assume that the run was successful, Case a) cannot apply. In Cases b)–d), σ_S must solve $\mathfrak{s}$ by Lemma 3. Finally, in Case e), we have $C = \exists r.C'$, $D = \exists r.D'$, and $\mathfrak{s}' = C' \not\sqsubseteq^? D' \in \hat{\Gamma}$. Notice that $\mathfrak{s} \succ \mathfrak{s}'$, because $m_1(\mathfrak{s}) = m_1(\mathfrak{s}')$ and $m_2(\mathfrak{s}) > m_2(\mathfrak{s}')$. Hence, by induction we get $\sigma_S(C') \not\sqsubseteq \sigma_S(D')$ and thus $\sigma_S(C) \not\sqsubseteq \sigma_S(D)$ by Lemma 3.
- **Local Extension:** Then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ and there is a non-variable atom $D \in S_X$ such that $\mathfrak{s}' = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D \in \hat{\Gamma}$. We have $\mathfrak{s} \succ \mathfrak{s}'$, because D may only contain a variable strictly smaller than X, and thus $m_1(\mathfrak{s}) > m_1(\mathfrak{s}')$. Hence by induction, σ solves $\mathfrak{s}'$. Since $\sigma_S(D)$ is a top-level atom of $\sigma_S(X)$, σ_S solves $\mathfrak{s}$ by Lemma 2.

- If $\mathfrak{s}$ is a subsumption that is initially solved, then $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ with $X \in \mathsf{Var}$. By expansion, for every $E \in S_X$, there is a subsumption $\mathfrak{s}_E = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? E$ in $\hat{\Gamma}$. We have $\mathfrak{s}_E \prec \mathfrak{s}$ since $m_1(\mathfrak{s}_E) < m_1(\mathfrak{s})$, for every $E \in S_X$. Hence, by induction all subsumptions $\mathfrak{s}_E$ are solved by σ_S. Since the top-level atoms of $\sigma_S(X)$ are exactly those of the form $\sigma_S(E)$ for $E \in S_X$, σ_S solves $\mathfrak{s}$ by Lemma 1.

- If $\mathfrak{s}$ is a dissubsumption that is initially solved, then $\mathfrak{s} = X \not\sqsubseteq^? D$ for $X \in \mathsf{Var}$ and $D \in \mathsf{At}_{\mathsf{nv}}$. By expansion, for every $E \in S_X$, we have $\mathfrak{s}_E = E \not\sqsubseteq^? D \in \hat{\Gamma}$.

We know that $\mathfrak{s} \succ \mathfrak{s}_E$, because E may only contain a variable strictly smaller than X, and thus $m_1(\mathfrak{s}) > m_1(\mathfrak{s}_E)$. Hence by induction, σ_S solves all dissubsumptions $\mathfrak{s}_E$ with $E \in S_X$. By the definition of $\sigma_S(X)$ and Lemma 2, σ_S also solves $\mathfrak{s}$. □

5.3 Completeness

Let σ be a local solution of Γ_0. We show that σ can guide the choices of Algorithm 12 to obtain a local solution σ' of Γ_0 such that, for every variable X, we have $\sigma(X) \sqsubseteq \sigma'(X)$. The following invariants will be maintained throughout the run of the algorithm for the current set of (dis)subsumptions Γ and the current assignment S:

I. σ is a solution of Γ.

II. For each $D \in S_X$, we have that $\sigma(X) \sqsubseteq \sigma(D)$.

By Lemma 1, chains of the form $\sigma(X_1) \sqsubseteq \sigma(\exists r_1.X_2), \dots \sigma(X_{n-1}) \sqsubseteq \sigma(\exists r_{n-1}.X_n)$ with $X_1 = X_n$ are impossible, and thus invariant II implies that S is acyclic. Hence, if extending S during a rule application preserves this invariant, this extension will not cause the algorithm to fail.

Lemma 16. *The invariants are maintained by the operation of expanding* Γ.

Proof. Since expansion does not affect the assignment S, we have to check only invariant I. Consider a subsumption $\mathfrak{s} = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X$ in Γ, for which a new subsumption $\mathfrak{s}_E = C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? E$ is created because $E \in S_X$. By the invariants, σ solves $\mathfrak{s}$ and $\sigma(X) \sqsubseteq \sigma(E)$. Hence by transitivity, σ also solves $\mathfrak{s}'$, i.e. invariant I is satisfied after adding $\mathfrak{s}_E$ to Γ.

For a dissubsumption $\mathfrak{s} = X \not\sqsubseteq^? D \in \Gamma$ and $E \in S_X$, a new dissubsumption $\mathfrak{s}_E = E \not\sqsubseteq^? D$ is created. Since σ solves $\mathfrak{s}$ and $\sigma(X) \sqsubseteq \sigma(E)$ by invariant II, we have $\sigma(E) \not\sqsubseteq \sigma(D)$ by transitivity of subsumption, i.e. σ solves $\mathfrak{s}_E$. □

Now we show that if the invariants are satisfied, the eager rules maintain the invariants and do not lead to failure.

Lemma 17. *The application of an eager rule never fails and maintains the invariants.*

Proof. There are six eager rules to consider:

- **Eager Ground Solving:** By invariant I, σ solves all ground subsumptions in Γ, and thus they must be valid subsumptions. Therefore the rule cannot fail, and obviously it preserves the invariants.

- **Eager Solving:** The rule cannot fail and does not affect the invariants.
- **Eager Extension:** Consider any $C_1 \sqcap \cdots \sqcap C_m \sqsubseteq^? D \in \Gamma$ such that there is an index $i \in \{1, \ldots, n\}$ with $C_i = X \in \mathsf{Var}$ and $\{C_1, \ldots, C_m\} \setminus \{X\} \subseteq S_X$. By the invariants and Lemma 1, we have $\sigma(X) \sqsubseteq \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_m) \sqsubseteq \sigma(D)$, and thus adding D to S_X maintains invariant II. Therefore, the application of the rule does not cause S to be cyclic, and does not fail. Invariant I is not affected by this rule.
- **Eager Top Solving:** By invariant I, this rule will never be applied since $\sigma(C) \not\sqsubseteq^? \top$ is impossible by Lemma 2.
- **Eager Left Decomposition:** Notice that this rule never fails. Furthermore, S is not affected by the rule, and hence invariant I is preserved. Finally, if σ solves $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$, then it must also solve $C_i \not\sqsubseteq^? D$ for each $i \in \{1, \ldots, n\}$ by Lemma 2.
- **Eager Atomic Decomposition:** Case a) cannot apply since σ is a solution of Γ. Invariant II is not affected, because S is not changed by these rules. The fact that invariant I is maintained in Case e) follows from Lemma 3. $\square$

Now we show that the non-deterministic rules can be applied in such a way that the invariants are maintained and the application does not lead to failure.

Lemma 18. *If $\mathfrak{s}$ is an unsolved (dis)subsumption of Γ to which no eager rule applies, then there is a nondeterministic rule that can be successfully applied to $\mathfrak{s}$ while maintaining the invariants.*

Proof. If $\mathfrak{s}$ is an unsolved subsumption, then it is of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where D is a non-variable atom. By invariant I, we have $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq \sigma(D)$. By Lemma 1, there is an index $i \in \{1, \ldots, n\}$ and a top-level atom E of $\sigma(C_i)$ such that $E \sqsubseteq \sigma(D)$.

- If C_i is a constant, then by Lemma 1 we have $C_i = E = D$, and thus Eager Solving is applicable, which contradicts the assumption.
- If $C_i = \exists r.C'$, then $\sigma(C_i) = \exists r.\sigma(C') = E$ and by Lemma 1 we must have $D = \exists r.D'$ and $\sigma(C') \sqsubseteq \sigma(D')$. Thus, the Decomposition rule can be successfully applied to $\mathfrak{s}$ and results in a new subsumption $C' \sqsubseteq^? D'$ that is solved by σ.
- If C_i is a variable, then invariant II is preserved by adding D to S_{C_i} since $\sigma(C_i) \sqsubseteq E\sigma(D)$. Thus, we can successfully apply the Extension rule to $\mathfrak{s}$.

If $\mathfrak{s}$ is an unsolved dissubsumption, then it must be of the form $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ since otherwise one of the eager rules in Figure 5 would be applicable to it. We have $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \not\sqsubseteq \sigma(X)$ by invariant I. By Lemma 2, there is a top-level atom E of $\sigma(X)$ such that $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \not\sqsubseteq E$. Since σ is local, we must have $E = \sigma(D)$ for some $D \in \mathsf{At}_{\mathsf{nv}}$. Hence, adding D to S_X maintains invariant II, and adding $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$ to Γ maintains invariant I. Thus, we can successfully apply the Local Extension rule to $\mathfrak{s}$. □

This concludes the proof of correctness of Algorithm 12, which provides a more goal-directed way to solve local disunification problems than blindly guessing an assignment as described in Section 4.

Theorem 19. *The flat disunification problem Γ_0 has a local solution iff there is a successful run of Algorithm 12 on Γ_0.*

6 Encoding local disunification into SAT

The following reduction to SAT is a generalization of the one for unification problems in [2]. We again consider a flat disunification problem Γ and the sets At, Var, and $\mathsf{At}_{\mathsf{nv}}$ as in Section 3. Since we are restricting our considerations to *local* solutions, we can without loss of generality assume that the sets $\mathsf{N_v}$, $\mathsf{N_c}$, and $\mathsf{N_R}$ contain exactly the variables, constants, and role names occurring in Γ. To further simplify the reduction, we assume in the following that all flat dissubsumptions in Γ are of the form $X \not\sqsubseteq^? Y$ for variables X, Y. This is without loss of generality, which can be shown using a transformation similar to the flattening rules from Section 4.

The translation into SAT uses the propositional variables $[C \sqsubseteq D]$ for all $C, D \in \mathsf{At}$. The SAT problem consists of a set of clauses $\mathsf{Cl}(\Gamma)$ over these variables that express properties of (dis)subsumption in $\mathcal{EL}$ and encode the elements of Γ. The intuition is that a satisfying valuation of $\mathsf{Cl}(\Gamma)$ induces a local solution σ of Γ such that $\sigma(C) \sqsubseteq \sigma(D)$ holds whenever $[C \sqsubseteq D]$ is true under the valuation. The solution σ is constructed by first extracting an acyclic assignment S out of the satisfying valuation and then computing $\sigma := \sigma_S$. We additionally introduce the variables $[X > Y]$ for all $X, Y \in \mathsf{N_v}$ to ensure that the generated assignment S is indeed acyclic. This is achieved by adding clauses to $\mathsf{Cl}(\Gamma)$ that express that $>_S$ is a strict partial order, i.e. irreflexive and transitive.

Finally, we use the auxiliary variables $p_{C,X,D}$ for all $X \in \mathsf{N_v}$, $C \in \mathsf{At}$, and $D \in \mathsf{At}_{\mathsf{nv}}$ to express the restrictions imposed by dissubsumptions of the form $C \not\sqsubseteq^? X$ in clausal form. More precisely, whenever $[C \sqsubseteq X]$ is false for some $X \in \mathsf{N_v}$ and $C \in \mathsf{At}$, then the dissubsumption $\sigma(C) \not\sqsubseteq \sigma(X)$ should hold. By Lemma 2, this means that we need to find an atom $D \in \mathsf{At}_{\mathsf{nv}}$ that is a top-level atom of $\sigma(X)$ and satisfies $\sigma(C) \not\sqsubseteq \sigma(D)$. This is enforced by making the auxiliary variable $p_{C,X,D}$ true, which makes $[X \sqsubseteq D]$ true and $[C \sqsubseteq D]$ false (see Definition 20(IV) and Lemma 23 for details).

Definition 20. The set $\mathsf{Cl}(\Gamma)$ contains the following propositional clauses:

(I) *Translation of* Γ.

a. For every subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ with $D \in \mathsf{At}_{\mathsf{nv}}$:

$$\to [C_1 \sqsubseteq D] \vee \cdots \vee [C_n \sqsubseteq D]$$

b. For every subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ in Γ with $X \in \mathsf{N}_{\mathsf{v}}$, and every $E \in \mathsf{At}_{\mathsf{nv}}$:

$$[X \sqsubseteq E] \to [C_1 \sqsubseteq E] \vee \cdots \vee [C_n \sqsubseteq E]$$

c. For every dissubsumption $X \not\sqsubseteq^? Y$ in Γ: $[X \sqsubseteq Y] \to$

(II) *Properties of subsumptions between non-variable atoms.*

a. For every $A \in \mathsf{N}_{\mathsf{c}}$: $\to [A \sqsubseteq A]$

b. For every $A, B \in \mathsf{N}_{\mathsf{c}}$ with $A \neq B$: $[A \sqsubseteq B] \to$

c. For every $\exists r.A, \exists s.B \in \mathsf{At}_{\mathsf{nv}}$ with $r \neq s$: $[\exists r.A \sqsubseteq \exists s.B] \to$

d. For every $A \in \mathsf{N}_{\mathsf{c}}$ and $\exists r.B \in \mathsf{At}_{\mathsf{nv}}$:

$$[A \sqsubseteq \exists r.B] \to \quad \text{and} \quad [\exists r.B \sqsubseteq A] \to$$

e. For every $\exists r.A, \exists r.B \in \mathsf{At}_{\mathsf{nv}}$:

$$[\exists r.A \sqsubseteq \exists r.B] \to [A \sqsubseteq B] \quad \text{and} \quad [A \sqsubseteq B] \to [\exists r.A \sqsubseteq \exists r.B]$$

(III) *Transitivity of subsumption.*
For every $C_1, C_2, C_3 \in \mathsf{At}$: $[C_1 \sqsubseteq C_2] \wedge [C_2 \sqsubseteq C_3] \to [C_1 \sqsubseteq C_3]$

(IV) *Dissubsumptions of the form $C \not\sqsubseteq^? X$ with a variable X.*
For every $C \in \mathsf{At}$, $X \in \mathsf{N}_{\mathsf{v}}$:

$$\to [C \sqsubseteq X] \vee \bigvee_{D \in \mathsf{At}_{\mathsf{nv}}} p_{C,X,D},$$

and additionally for every $D \in \mathsf{At}_{\mathsf{nv}}$:

$$p_{C,X,D} \to [X \sqsubseteq D] \quad \text{and} \quad p_{C,X,D} \wedge [C \sqsubseteq D] \to$$

(V) *Properties of* $>$.

a. For every $X \in \mathsf{N}_{\mathsf{v}}$: $[X > X] \to$

b. For every $X, Y, Z \in \mathsf{N}_{\mathsf{v}}$: $[X > Y] \wedge [Y > Z] \to [X > Z]$

c. For every $X, Y \in \mathsf{N}_{\mathsf{v}}$ and $\exists r.Y \in \mathsf{At}$: $[X \sqsubseteq \exists r.Y] \to [X > Y]$

The main difference to the encoding in [2] (apart from the fact that we consider (dis)subsumptions here instead of equations) lies in the clauses (IV) that ensure the presence of a non-variable atom D that solves the dissubsumption $C \not\sqsubseteq^? X$ (cf. Lemma 2). We also need some additional clauses in (II) to deal with dissubsumptions. It is easy to see that $\mathsf{Cl}(\Gamma)$ can be constructed in time cubic in the size of Γ (due to the clauses in (III) and (V)b). We prove the correctness of this reduction in the following two sections.

6.1 Soundness

Let τ be a valuation of the propositional variables that satisfies $\mathsf{Cl}(\Gamma)$. We define the assignment S^τ as follows:

$$S_X^\tau := \{D \in \mathsf{At}_{\mathsf{nv}} \mid \tau([X \sqsubseteq D]) = 1\}.$$

We show the following connection between $>_{S^\tau}$ and the order relation encoded by the propositional variables $[X > Y]$. The proof is exactly the same as in [2], but uses a different notation.

Lemma 21. *The relation $>_{S^\tau}$ is irreflexive.*

Proof. We first show that $X >_{S^\tau} Y$ implies $\tau([X > Y]) = 1$ for all $X, Y \in \mathsf{N_v}$. If Y occurs in an atom of S_X^τ, then this atom must be of the form $\exists r.Y$ with $r \in \mathsf{N_R}$. By construction of S^τ, this implies that $\tau([X \sqsubseteq \exists r.Y]) = 1$. Since τ satisfies the clauses in (V)c, we have $\tau([X > Y]) = 1$. By definition of $>_{S^\tau}$ and the transitivity clauses in (V)b, we conclude that $\tau([X > Y]) = 1$ whenever $X >_{S^\tau} Y$.

Assume now that $X >_{S^\tau} X$ holds for some $X \in \mathsf{N_v}$. By the claim above, this implies that $\tau([X > X]) = 1$. But this is impossible since τ satisfies the clauses in (V)a. $\square$

This in particular shows that S^τ is acyclic. In the following, let σ_τ denote the substitution σ_{S^τ} induced by S^τ. We show that σ_τ is a solution of Γ.

Lemma 22. *If $C, D \in \mathsf{At}$ such that $\tau([C \sqsubseteq D]) = 1$, then $\sigma_\tau(C) \sqsubseteq \sigma_\tau(D)$.*

Proof. We show this by induction on the pairs $(\mathsf{rd}(\sigma_\tau(D)), \mathsf{Var}(D))$, where $\mathsf{Var}(D)$ is either the variable that occurs in D, or $\bot$ if D is ground. These pairs are compared by the lexicographic extension of the order $>$ on natural numbers for the first component and the order $>_{S^\tau}$ for the second component, which is extended by $Y >_{S^\tau} \bot$ for all $Y \in \mathsf{N_v}$.

We make a case distinction on the form of C and D and consider first the case that D is a variable. Let $\sigma_\tau(E)$ be any top-level atom of $\sigma_\tau(D)$, which means that $\tau([D \sqsubseteq E]) = 1$. By the clauses in (III), we also have $\tau([C \sqsubseteq E]) = 1$.

Since $\mathsf{rd}(\sigma_\tau(D)) \geq \mathsf{rd}(\sigma_\tau(E))$ and $\mathsf{Var}(D) = D >_{S^\tau} \mathsf{Var}(E)$, by induction we get $\sigma_\tau(C) \sqsubseteq \sigma_\tau(E)$. Since $\sigma_\tau(D)$ is equivalent to the conjunction of all its top-level atoms, by Lemma 1 we obtain $\sigma_\tau(C) \sqsubseteq \sigma_\tau(D)$.

If D is a non-variable atom and C is a variable, then $\sigma_\tau(C) \sqsubseteq \sigma_\tau(D)$ holds by construction of S^τ and Lemma 1.

If C, D are both non-variable atoms, then by the clauses in (II) they must either be the same concept constant, or be existential restrictions using the same role name. In the first case, the claim follows immediately. In the latter case, let $C = \exists r.C'$ and $D = \exists r.D'$. By the clauses in (II)e, we have $\tau([C' \sqsubseteq D']) = 1$. Since $\mathsf{rd}(\sigma_\tau(D)) > \mathsf{rd}(\sigma_\tau(D'))$, by induction we get $\sigma_\tau(C') \sqsubseteq \sigma_\tau(D')$, and thus $\sigma_\tau(C) \sqsubseteq \sigma_\tau(D)$ by Lemma 1. □

We now show that the converse of this lemma also holds.

Lemma 23. *If $C, D \in \mathsf{At}$ such that $\tau([C \sqsubseteq D]) = 0$, then $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D)$.*

Proof. We show this by induction on the tuples $(\mathsf{rd}(\sigma_\tau(C)), \mathsf{Var}(C), \mathsf{Var}(D))$ and make a case distinction on the form of C and D. If D is a variable, then by the clauses in (IV) there must be a $D' \in \mathsf{At}_{\mathsf{nv}}$ such that $\tau(p_{C,D,D'}) = 1$. This implies that $\tau([D \sqsubseteq D']) = 1$ and $\tau([C \sqsubseteq D']) = 0$. By construction of S^τ, $\sigma_\tau(D')$ is a top-level atom of $\sigma_\tau(D)$ and $\mathsf{Var}(D) >_{S^\tau} \mathsf{Var}(D')$. Since $\mathsf{rd}(\sigma_\tau(C)) = \mathsf{rd}(\sigma_\tau(C))$ and $\mathsf{Var}(C) = \mathsf{Var}(C)$, by induction we get $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D')$, and thus $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D)$ by Lemma 2.

If D is a non-variable atom and C is a variable, then consider any top-level atom $\sigma_\tau(E)$ of $\sigma_\tau(C)$, which means that we have $\tau([C \sqsubseteq E]) = 1$. By the clauses in (III) this implies that $\tau([E \sqsubseteq D]) = 0$. Since we have $\mathsf{rd}(\sigma_\tau(C)) \geq \mathsf{rd}(\sigma_\tau(E))$ and $\mathsf{Var}(C) = C >_{S^\tau} \mathsf{Var}(E)$, by induction we get $\sigma_\tau(E) \not\sqsubseteq \sigma_\tau(D)$. Since $\sigma_\tau(C)$ is equivalent to the conjunction of all its top-level atoms, by Lemma 2 we get $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D)$.

If C, D are both non-variable atoms, then by the clauses in (II), they are either different constants, a constant and an existential restriction, or two existential restrictions. In the first two cases, $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D)$ holds by Lemma 1. In the last case, they can either contain two different roles or the same role. Again, the former case is covered by Lemma 1, while in the latter case we have $C = \exists r.C'$, $D = \exists r.D'$, and $\tau([C' \sqsubseteq D']) = 0$ by the clauses in (II)e. Since $\mathsf{rd}(\sigma_\tau(C)) > \mathsf{rd}(\sigma_\tau(C'))$, by induction we get $\sigma_\tau(C') \not\sqsubseteq \sigma_\tau(D')$, and thus $\sigma_\tau(C) \not\sqsubseteq \sigma_\tau(D)$ by Lemma 2. □

This suffices to show soundness of the reduction.

Lemma 24. *If $\mathsf{Cl}(\Gamma)$ is solvable, then Γ has a local solution.*

Proof. Since σ_τ is obviously local, it suffices to show that it solves Γ.

Consider any flat subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ. If $D \in \mathsf{At}_{\mathsf{nv}}$, then we have $\sigma_\tau(C_i) \sqsubseteq \sigma_\tau(D)$ for some i, $1 \leq i \leq n$, by the clauses in (I) and Lemma 22. By Lemma 1, σ_τ solves the subsumption.

If D is a variable, then consider any top-level atom $\sigma_\tau(E)$ of $\sigma_\tau(D)$, for which we must have $\tau([D \sqsubseteq E]) = 1$. By the clauses in (I), there must be an i, $1 \leq i \leq n$, such that $\tau([C_i \sqsubseteq E]) = 1$, and thus $\sigma_\tau(C_i) \sqsubseteq \sigma_\tau(E)$ by Lemma 22. Again, by Lemma 1 this implies that σ_τ solves the subsumption.

Finally, consider a dissubsumption $X \not\sqsubseteq^? Y$ in Γ. Then by the clauses in (I) and Lemma 23 we have $\sigma_\tau(X) \not\sqsubseteq \sigma_\tau(Y)$ i.e. σ_τ solves the dissubsumption. ☐

6.2 Completeness

Let now σ be a ground local solution of Γ and $>_\sigma$ the resulting partial order on $\mathsf{N_v}$, defined as follows for all $X, Y \in \mathsf{N_v}$:

$$X >_\sigma Y \text{ iff } \sigma(X) \sqsubseteq \exists r_1. \ldots \exists r_n.\sigma(Y) \text{ for some } r_1, \ldots, r_n \in \mathsf{N_R} \text{ with } n \geq 1.$$

Note that $>_\sigma$ is irreflexive since $X >_\sigma X$ is impossible by Lemma 1, and it is transitive since $\sqsubseteq$ is transitive and closed under applying existential restrictions on both sides. Thus, $>_\sigma$ is a strict partial order. We define a valuation τ_σ as follows for all $C, D \in \mathsf{At}$, $E \in \mathsf{At}_{\mathsf{nv}}$, and $X, Y \in \mathsf{N_v}$:

$$\tau_\sigma([C \sqsubseteq D]) := \begin{cases} 1 & \text{if } \sigma(C) \sqsubseteq \sigma(D) \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_\sigma(p_{C,X,E}) := \begin{cases} 1 & \text{if } \sigma(X) \sqsubseteq \sigma(E) \text{ and } \sigma(C) \not\sqsubseteq \sigma(E) \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_\sigma([X > Y]) := \begin{cases} 1 & \text{if } X >_\sigma Y \\ 0 & \text{otherwise} \end{cases}$$

Lemma 25. *If Γ has a local solution, then $\mathsf{Cl}(\Gamma)$ is solvable.*

Proof. We verify that τ_σ satisfies all clauses of Definition 20.

For (I)a, consider any flat subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ with $D \in \mathsf{At}_{\mathsf{nv}}$. Since σ solves Γ, we have $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq \sigma(D)$. Since $\sigma(D)$ is an atom, by Lemma 1 there must be an i, $1 \leq i \leq n$, and a top-level atom E of $\sigma(C_i)$ such that $\sigma(C_i) \sqsubseteq E \sqsubseteq \sigma(D)$. By the definition of τ_σ, this shows that $\tau_\sigma([C_i \sqsubseteq D]) = 1$, and thus the clause is satisfied.

Consider now an arbitrary flat subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ from Γ where X is a variable, and any $E \in \mathsf{At}_{\mathsf{nv}}$ such that $\tau_\sigma([X \sqsubseteq E]) = 1$. This implies that we have $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq \sigma(X) \sqsubseteq \sigma(E)$, and thus as above there is a top-level atom F of some $\sigma(C_i)$ such that $\sigma(C_i) \sqsubseteq F \sqsubseteq \sigma(E)$, which shows that $\tau_\sigma([C_i \sqsubseteq E]) = 1$, as required for the clause in (I)b.

For every dissubsumption $X \not\sqsubseteq^? Y$ in Γ, we must have $\sigma(X) \not\sqsubseteq \sigma(Y)$, and thus $\tau_\sigma([X \sqsubseteq Y]) = 0$, satisfying the clause in (I)c.

For $A \in \mathsf{N_c}$, we have $\sigma(A) \sqsubseteq \sigma(A)$, and thus $\tau_\sigma([A \sqsubseteq A]) = 1$. Similar arguments show that the remaining clauses in (II) are also satisfied (see Lemma 1). For (III), consider $C_1, C_2, C_3 \in \mathsf{At}$ with $\tau_\sigma([C_1 \sqsubseteq C_2]) = \tau_\sigma([C_2 \sqsubseteq C_3]) = 1$, and thus $\sigma(C_1) \sqsubseteq \sigma(C_2) \sqsubseteq \sigma(C_3)$. By transitivity of $\sqsubseteq$, we infer $\tau_\sigma([C_1 \sqsubseteq C_3]) = 1$.

For all $C \in \mathsf{At}$, $X \in \mathsf{N_v}$, and $D \in \mathsf{At_{nv}}$ with $\tau_\sigma(p_{C,X,D}) = 1$, we must have $\tau_\sigma([X \sqsubseteq D]) = 1$ and $\tau_\sigma([C \sqsubseteq D]) = 0$ by the definition of τ_σ. Furthermore, whenever $\tau_\sigma([C \sqsubseteq X]) = 0$, we have $\sigma(C) \not\sqsubseteq \sigma(X)$, and thus by Lemma 2 there must be a top-level atom E of $\sigma(X)$ such that $\sigma(C) \not\sqsubseteq E$. Since σ is a local solution, E must be of the form $\sigma(F)$ for some $F \in \mathsf{At_{nv}}$, and thus we obtain $\sigma(X) \sqsubseteq \sigma(F)$ and $\sigma(C) \not\sqsubseteq \sigma(F)$, and hence $\tau_\sigma(p_{C,X,F}) = 1$. This shows that all clauses in (IV) are satisfied by τ_σ.

For (V)a, recall that $>_\sigma$ is irreflexive. Transitivity of $>_\sigma$ yields satisfaction of the clauses in (V)b. Finally, if $\sigma(X) \sqsubseteq \sigma(\exists r.Y) = \exists r.\sigma(Y)$ for some $X, Y \in \mathsf{N_v}$ with $\exists r.Y \in \mathsf{At}$, we have $X >_\sigma Y$ by definition, and thus the clauses in (V)c are satisfied by τ_σ. □

This completes the proof of the correctness of the translation presented in Definition 20, which provides us with a reduction of local disunification (and thus also of dismatching) to SAT. This SAT reduction has been implemented in our prototype system UEL,[5] which uses SAT4J[6] as external SAT solver. First experiments show that dismatching is indeed helpful for reducing the number and the size of unifiers. The runtime performance of the solver for dismatching problems is comparable to the one for pure unification problems.

7 Related and future work

Since Description Logics and Modal Logics are closely related [25], results on unification in one of these two areas carry over to the other one. In Modal Logics, unification has mostly been considered for expressive logics with all Boolean operators [18, 19, 24]. An important open problem in the area is the question whether unification in the basic modal logic K, which corresponds to the DL $\mathcal{ALC}$, is decidable. It is only known that relatively minor extensions of K have an undecidable unification problem [26].

[5] version 1.3.0, available at `http://uel.sourceforge.net/`

[6] `http://www.sat4j.org/`

Disunification also plays an important role in Modal Logics since it is basically the same as the admissibility problem for inference rules [14, 21, 23]. To be more precise, a given a normal modal logic L induces an equational theory E_L that axiomatizes equivalence in this logic, where the formulas are viewed as terms. Validity is then just equivalence to $\top$ and inconsistency is equivalence to $\bot$. An *inference rule* is of the form

$$\frac{A_1, \ldots, A_m}{B_1, \ldots, B_n} \tag{4}$$

where $A_1, \ldots, B_n$ are formulas (terms) that may contain variables. More precisely, it is not a single rule but a rule schema that stands for all its instances

$$\frac{\sigma(A_1), \ldots, \sigma(A_m)}{\sigma(B_1), \ldots, \sigma(B_n)} \tag{5}$$

where σ is a substitution. The semantics of such a rule (5) is the following: whenever all of its premises are valid, then one of the consequences must be valid as well. We only admit the inference rule (4) for the logic L if all its instances (5) satisfy this requirement. Thus, we say that the inference rule (4) is *admissible for L* if

$$\sigma(A_1) =_{E_L} \top \wedge \ldots \wedge \sigma(A_m) =_{E_L} \top \;\text{ implies }\; \sigma(B_1) =_{E_L} \top \vee \ldots \vee \sigma(B_n) =_{E_L} \top$$

for all substitutions σ. Obviously, this is the case iff the disunification problem

$$\{A_1 \equiv^? \top, \ldots, A_m \equiv^? \top, B_1 \not\equiv^? \top, \ldots, B_n \not\equiv^? \top\}$$

does *not* have a solution.

Regarding future work, we want to investigate the decidability and complexity of general disunification in $\mathcal{EL}$, and consider also the case where non-ground solutions are allowed. From a more practical point of view, we plan to implement also the goal-oriented algorithm for local disunification, and to evaluate the performance of both presented algorithms on real-world problems.

References

[1] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010. doi: 10.2168/LMCS-6(3:17)2010.

[2] Franz Baader and Barbara Morawska. SAT encoding of unification in $\mathcal{EL}$. In Christian G. Fermüller and Andrei Voronkov, editors, *Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10)*, volume 6397 of *Lecture Notes in Computer Science*, pages 97–111. Springer-Verlag, 2010. doi: 10.1007/978-3-642-16242-8_8.

[3] Franz Baader and Barbara Morawska. Matching with respect to general concept inclusions in the description logic $\mathcal{EL}$. In Carsten Lutz and Michael Thielscher, editors, *Proc. of the 37th German Conf. on Artificial Intelligence (KI'14)*, volume 8736 of *Lecture Notes in Computer Science*, pages 135–146. Springer-Verlag, 2014. doi: 10.1007/978-3-319-11206-0_14.

[4] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.

[5] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

[6] Franz Baader and Alexander Okhotin. Solving language equations and disequations with applications to disunification in description logics and monadic set constraints. In Nikolaj Bjørner and Andrei Voronkov, editors, *Proceedings of the 18th International Conference on Logic for Programming, Artifical Intelligence, and Reasoning (LPAR-12)*, volume 7180 of *Lecture Notes in Computer Science*, pages 107–121, Mérida, Venezuela, 2012. Springer-Verlag.

[7] Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3): 411–447, 1999. doi: 10.1093/logcom/9.3.411.

[8] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[9] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[10] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the Fifth International Workshop on OWL: Experiences and Directions (OWLED'08)*, Karlsruhe, Germany, 2008.

[11] Franz Baader, Stefan Borgwardt, Julian Alfredo Mendez, and Barbara Morawska. UEL: Unification solver for $\mathcal{EL}$. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proc. of the 25th Int. Workshop on Description Logics (DL'12)*, volume 846 of *CEUR Workshop Proceedings*, pages 26–36, 2012. URL `http://ceur-ws.org/Vol-846/paper_8.pdf`.

[12] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Computing minimal $\mathcal{EL}$-unifiers is hard. In Silvio Ghilardi and Lawrence Moss, editors, *Proceedings of the 9th International Conference on Advances in Modal Logic (AiML'12)*, 2012.

[13] Franz Baader, Stefan Borgwardt, and Barbara Morawska. A goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proc. of the 25th Int. Workshop on Description Logics (DL'12)*, volume 846 of *CEUR Workshop Proceedings*, pages 37–47, 2012. URL `http://ceur-ws.org/Vol-846/paper_1.pdf`.

[14] Sergey Babenyshev, Vladimir V. Rybakov, Renate Schmidt, and Dmitry Tishkovsky. A tableau method for checking rule admissibility in $S4$. In *Proc. of the 6th Workshop on Methods for Modalities (M4M-6)*, Copenhagen, 2009.

[15] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.

[16] W. L. Buntine and H.-J. Bürckert. On solving equations and disequations. *J. of the ACM*, 41(4):591–629, 1994.

[17] H. Comon. Disunification: A survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 322–359. MIT Press, Cambridge, MA, 1991.

[18] Silvio Ghilardi. Unification through projectivity. *Journal of Logic and Computation*, 7(6):733–752, 1997.

[19] Silvio Ghilardi. Unification in intuitionistic logic. *Journal of Logic and Computation*, 64(2):859–880, 1999.

[20] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[21] Rosalie Iemhoff and George Metcalfe. Proof theory for admissible rules. *Annals of Pure and Applied Logic*, 159(1-2):171–186, 2009.

[22] Ralf Küsters. Chapter 6: Matching. In *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Computer Science*, pages 153–227. Springer-Verlag, 2001. doi: 10.1007/3-540-44613-3_6.

[23] Vladimir V. Rybakov. *Admissibility of logical inference rules*, volume 136 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1997.

[24] Vladimir V. Rybakov. Multi-modal and temporal logics with universal formula - reduction of admissibility to validity and unification. *Journal of Logic and Computation*, 18(4):509–519, 2008.

[25] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In John Mylopoulos and Raymond Reiter, editors, *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471. Morgan Kaufmann, 1991. URL `http://ijcai.org/Past%20Proceedings/IJCAI-91-VOL1/PDF/072.pdf`.

[26] Frank Wolter and Michael Zakharyaschev. Undecidability of the unification and admissibility problems for modal and description logics. *ACM Transactions on Computational Logic*, 9(4):25:1–25:20, 2008. doi: 10.1145/1380572.1380574.