TECHNISCHE UNIVERSITÄT DRESDEN

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

LTCS-Report

Constructing SNOMED CT Concepts via Disunification

Franz Baader Stefan Borgwardt Barbara Morawska

LTCS-Report 17-07

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

1 Introduction

Description Logics (DLs) [BCM+07] are prominent modeling formalisms underlying the Web Ontology Language (OWL).[1] The lightweight DL $\mathcal{EL}$ in particular is used to formulate many biomedical ontologies.[2] DLs allow to represent subconcept-superconcept relationships between concepts, e.g., diseases, as well as more complex correspondences. Unification in DLs has been proposed as a non-standard reasoning task to detect redundant concepts in ontologies [BN01, BM10b]. Recently, disunification in $\mathcal{EL}$ has been investigated and several algorithms were proposed to solve disunification problems [BBM16].

We investigate the use of (local) disunification in $\mathcal{EL}$ for *ontology generation*, i.e., supporting an ontology engineer in the task of creating an ontology. Common approaches in this area focus on *ontology learning*, which tries to extract meaningful concepts and definitions from other kinds of data, e.g., facts about instances or textual descriptions of concepts [LV14]. Tools that implement such approaches are particularly important for maintaining and updating very large ontologies. For example, the biomedical ontology SNOMED CT published by SNOMED International[3] contains around 300.000 atomic concepts[4] describing many things from the *bones of the inner ear* to the *treatments of bacterial infections*. The contents of SNOMED CT can be formalized in $\mathcal{EL}$. By offering automated modeling tools for SNOMED CT, the quality of SNOMED CT can be increased, making it more useful for "post-coordination", i.e., reasoning [RI12].

This initial study on the benefit of disunification for ontology generation tackles the following concrete problem: Given an existing SNOMED CT concept, can we reconstruct its definition given only the definitions of its parents (i.e., direct superconcepts) and siblings (i.e., concepts having the same parents)? There are several obstacles that have to be overcome before we can give a positive answer. In particular, a straightforward application of the SAT-based algorithm from [BBM16] yields too many possible solutions. However, most of these solutions do not even represent legal SNOMED CT concepts as prescribed in the SNOMED CT Editorial Guide.[5] Hence, we augment the disunification algorithm by additional constraints in order to restrict the set of possible solutions to conform to the SNOMED CT specification. For some concepts of SNOMED CT, this results in the set of solutions actually containing the original definition, which was our goal.

2 Disunification in $\mathcal{EL}$

The syntax of $\mathcal{EL}$ is defined based on two disjoint 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 *(primitive) concept definition* is an expression of the form $A \equiv C_A$ ($A \sqsubseteq C_A$), where A is a concept name and C_A a concept term. A is called a *defined* concept name, and C_A its *definition*. An *acyclic ontology* $\mathcal{O}$ is a set of concept definitions such that each concept name has at most one definition, and the following *depends on* relation between concept names is acyclic: A concept name A *depends on* a concept name B if there is a concept definition $A \equiv C_A$ or $A \sqsubseteq C_A$ in $\mathcal{O}$

[1] `https://www.w3.org/TR/owl-overview/`
[2] `https://bioportal.bioontology.org/`
[3] Version: January 2017 v1.0, see `http://snomed.org`
[4] SNOMED CT browser: `http://browser.ihtsdotools.org`
[5] `http://snomed.org/eg`

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 restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} := \{x \mid \exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

such that B occurs in C_A. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{O}$ if for each $A \equiv C_A$ ($A \sqsubseteq C_A$) in $\mathcal{O}$, it holds that $A^{\mathcal{I}} = C_A^{\mathcal{I}}$ ($A^{\mathcal{I}} \subseteq C_A^{\mathcal{I}}$).

A concept term C is *subsumed* by a concept term D w.r.t. $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) if for every model $\mathcal{I}$ of $\mathcal{O}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We write a *dissubsumption* $C \not\sqsubseteq_{\mathcal{O}} D$ to abbreviate the fact that $C \sqsubseteq_{\mathcal{O}} D$ does not hold. The two concept terms C and D are *equivalent* w.r.t. $\mathcal{O}$ (written $C \equiv_{\mathcal{O}} D$) if $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$, i.e., they are interpreted as the same set.

Since conjunction is interpreted as set 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 (even w.r.t. the empty ontology) 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 $A \in \mathsf{N_C}$.

We now recall the relevant definitions from [BBM16]. For the purposes of disunification, we 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. In the following, let $\mathcal{O}$ be an acyclic ontology that contains only ground concept terms.

A *(flat) disunification problem* Γ (w.r.t. $\mathcal{O}$) is a set of (dis)subsumptions of the form

$$C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^?_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_m \text{ (or } C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^?_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_m),$$

where $C_1, \ldots, C_n, D_1, \ldots, D_m$ are flat atoms. We use an equation $C \equiv^?_{\mathcal{O}} D$ to abbreviate the two subsumptions $C \sqsubseteq^?_{\mathcal{O}} D$ and $D \sqsubseteq^?_{\mathcal{O}} C$. When $\mathcal{O}$ is empty, we drop the subscript $\mathcal{O}$. A *substitution* σ (w.r.t. $\mathcal{O}$) maps every variable to a ground concept term constructed from the concept and role names appearing in $\mathcal{O}$. This mapping can be extended to all concept terms in the usual way. A substitution σ *solves* a (dis)subsumption $C \sqsubseteq^?_{\mathcal{O}} D$ ($C \not\sqsubseteq^?_{\mathcal{O}} D$) if $\sigma(C) \sqsubseteq_{\mathcal{O}} \sigma(D)$ ($\sigma(C) \not\sqsubseteq_{\mathcal{O}} \sigma(D)$). A substitution that solves a given disunification problem (w.r.t. $\mathcal{O}$) is called a *solution* of this problem. A disunification problem is *solvable* if it has a solution. The restriction to *flat* disunification problems is without loss of generality [BBM16].

We denote by $\mathsf{At}(\Gamma)$ the set of (flat) atoms occurring as subterms in Γ, by $\mathsf{Ex}(\Gamma)$ the set of existential restrictions occurring in Γ, by $\mathsf{Rol}(\Gamma)$ the set of role names occurring in Γ, by $\mathsf{Var}(\Gamma)$ the set of variables occurring in Γ, and by $\mathsf{At_{nv}}(\Gamma) := \mathsf{At}(\Gamma) \setminus \mathsf{Var}(\Gamma)$ the set of *non-variable atoms* of Γ.

2.1 Incorporating Acyclic Definitions

In [BM10b], a procedure is described that can eliminate an acyclic ontology $\mathcal{O}$ from a disunification problem Γ. As a preliminary step, we transform every primitive definition $A \sqsubseteq C_A$

in $\mathcal{O}$ into a full definition $A \equiv A_{\mathsf{UNDEF}} \sqcap C_A$ by introducing a fresh concept name A_{UNDEF} that represents the "undefined part" of the concept name A. Then, every definition $A \equiv C_A$ from $\mathcal{O}$ is added as $A \equiv^? C_A$ to the disunification problem, each such A is changed into a variable, and the result is denoted by $\Gamma^{\mathcal{O}}$, which is a disunification problem w.r.t. the empty ontology. Then, it holds that Γ has a solution w.r.t. $\mathcal{O}$ iff $\Gamma^{\mathcal{O}}$ has a solution w.r.t. $\emptyset$.

If the ontology $\mathcal{O}$ is very large, this approach is not practical. Instead, in our system UEL[6] we implemented a goal-oriented approach that only incorporates those definitions $A \equiv C$ from $\mathcal{O}$ into Γ that are actually relevant for the disunification problem. This is done in a recursive fashion: first, all definitions $A \equiv C_A$ are imported for which A occurs in Γ, then all definitions of newly imported concept names (those occurring in the definitions C_A) are imported, and so on.

2.2 Local Disunification

We are interested in so-called *local* solutions [BM10b, BBM16], which are restricted to use only the atoms occurring in the input problem. Although this restriction is not without loss of generality [BBM16], it is useful to restrict the search space. Deciding the existence of local solutions for $\mathcal{EL}$-disunification problems is NP-complete, and the decidability of this problem is open for general solutions.

Let Γ be a (flat) disunification problem w.r.t. the empty ontology. Let $S\colon \mathsf{Var}(\Gamma) \to 2^{\mathsf{At}_{\mathsf{nv}}(\Gamma)}$ be an *assignment (for Γ)*, i.e., a function that assigns to each variable $X \in \mathsf{Var}(\Gamma)$ a set $S_X \subseteq \mathsf{At}_{\mathsf{nv}}(\Gamma)$ of non-variable atoms. The relation $>_S$ on $\mathsf{Var}(\Gamma)$ is defined as the transitive closure of

$$\{(X, Y) \in \mathsf{Var}(\Gamma) \times \mathsf{Var}(\Gamma) \mid Y \text{ occurs in an atom of } S_X\}.$$

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 w.r.t. $>_S$, then all elements of S_X are ground and we simply take

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

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

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

It is easy to see that the concept terms $\sigma_S(D)$ are ground, and hence σ_S is a valid candidate for a solution of Γ. A substitution σ is called *local* (w.r.t. Γ) if there exists an acyclic assignment S for Γ such that $\sigma = \sigma_S$.

2.3 The SAT Reduction

One of the algorithms presented in [BM10a, BBM16] and implemented in UEL is based on a representation of the disunification problem as a propositional satisfiability problem. Since our approach is based on this reduction, we present its main ideas here. We again consider a flat disunification problem Γ w.r.t. the empty ontology.

The translation uses the propositional variables $[C \sqsubseteq D]$ for all $C, D \in \mathsf{At}(\Gamma)$. 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

[6] `http://julianmendez.github.io/uel/`

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, where $D \in S_X$ whenever $[X \sqsubseteq D]$ is evaluated to true, and then computing $\sigma := \sigma_S$. The additional variables $[X > Y]$ for all $X, Y \in \mathsf{N_v}$ 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.

The size of $\mathsf{Cl}(\Gamma)$ is polynomial in the size of Γ, and Γ has a local solution iff $\mathsf{Cl}(\Gamma)$ is satisfiable. Moreover, this reduction preserves *all* solutions (modulo equivalence), i.e., for every local solution σ of Γ there is a satisfying valuation τ of $\mathsf{Cl}(\Gamma)$ that yields a local solution σ_τ (as described above) that is *equivalent* to σ, i.e., we have $\sigma(X) \equiv \sigma_\tau(X)$ for all $X \in \mathsf{Var}(\Gamma)$.

3 Constructing SNOMED CT Concepts

Our approach can be roughly described as follows: Given any concept name A that has a full definition $A \equiv C_A$ in SNOMED CT, we want to construct a disunification problem Γ_A (without using this definition) that describes some constraints on a variable X_A intended to represent A. The procedure is *successful* if our disunification algorithm outputs at least one solution that represents the original definition of A in the sense that X_A is mapped to C_A. It has *failed* if no solutions for Γ_A can be found or none of the solutions represents C_A in this way (more precisely, none of the solutions found within a reasonable time limit represent C_A).

As a running example, consider the following definition for $A =$ Difficulty_writing[7] from SNOMED CT (in $\mathcal{EL}$ syntax):

$$
\begin{aligned}
\textsc{Difficulty\_writing} \equiv\ & \textsc{Language\_finding} \sqcap \\
& \textsc{Finding\_related\_to\_ability\_to\_write} \sqcap \\
& \exists \textsc{RoleGroup}.(\\
& \quad \exists \textsc{Has\_interpretation}.\textsc{Able\_with\_difficulty} \sqcap \\
& \quad \exists \textsc{Interprets}.\textsc{Ability\_to\_write} \\
&)
\end{aligned}
$$

In contrast to the other roles, the special role RoleGroup does not carry any medical semantics; it is used to group different existential restrictions together. In the above definition, it signals that the qualifier Able_with_difficulty refers to the Ability_to_write. This is useful in more complex concepts, e.g., when several abilities are described.

3.1 The Basic Problem

We start to construct the disunification problem Γ_A as follows. Let $\mathsf{parents}(A)$ denote the *parents* of A, which are those concept names that occur in the top-level of its definition C_A. In our example, these are Language_finding and Finding_related_to_ability_to_write. Further, for an element $B \in \mathsf{parents}(A)$, the set $\mathsf{siblings}_B(A)$ contains the *siblings* of A w.r.t. B, i.e., those concept names C that also have B as a parent. Difficulty_writing has many siblings w.r.t. Language_finding, but only 2 w.r.t. Finding_related_to_ability_to_write, namely, Able_to_write and Unable_to_write. While the former has a definition that is very similar to the one of Difficulty_writing, curiously Unable_to_write is not defined to be a subconcept of Language_finding.

[7] In SNOMED CT, this is represented by the IRI `http://snomed.info/id/102938007`. For clarity, we here use the (abbreviated) textual labels of the concept names.

We now construct the subsumptions and dissubsumptions

$$X_A \sqsubseteq^? B,\ B \not\sqsubseteq^? X_A \quad \text{for all } B \in \mathsf{parents}(A), \tag{1}$$

which express that X_A should be a proper subconcept of the parents of A, as well as

$$X_A \not\sqsubseteq^? C,\ C \not\sqsubseteq^? X_A \quad \text{for all } C \in \bigcap_{B \in \mathsf{parents}(A)} \mathsf{siblings}_B(A) \text{ that are fully defined}, \tag{2}$$

i.e., X_A should be incomparable to the siblings of A that share all its parents. The initial disunification problem Γ_A^0 is constructed from the subsumptions and dissubsumptions in (1) and (2) by recursively including all definitions of concept names occurring in them, as described in Section 2.1.

Together, the basic constraints in Γ_A^0 encode the relative position of A in the concept hierarchy of SNOMED CT. Additionally, the inclusion of the siblings of A in Γ_A^0 causes their definitions to be included in the final disunification problem. The idea is that definitions of siblings should be similar, and hence the definitions of A's siblings are one of the sources from which A's definition can be reconstructed. Note that A itself does not occur in Γ_A^0, and hence the disunification problem does not contain any knowledge about its true definition.

3.2 Additional Siblings

However, the definition of Able_to_write by itself is not enough to reconstruct the definition of Difficulty_writing—we also need to include the siblings of the concept names occurring in this definition *inside existential restrictions*. For example, without including the sibling Able_with_difficulty of Able it is impossible to reconstruct the definition of Difficulty_writing from the one of Able_to_write.

Including all possible siblings of all concept names in Γ_A^0 would lead to a disunification problem that is too large to handle with the SAT algorithm of UEL in its current form. Hence, we restrict ourselves here to the siblings of the most specific concept names in Γ_0, i.e., those that are not used in other definitions. However, in contrast to (2), we include *all* siblings of those concept names, and not only the ones that share all their parents. Formally, Γ_A^1 is obtained from Γ_A^0 by including the definitions of all $C \in \mathsf{siblings}_B(D)$, where D does not occur in any definition in Γ_A^0 except its own and B is a parent of D (and, as usual, recursively adding all definitions of concept names that are used in the definition of C).

3.3 Additional Atoms

To give the disunification algorithm additional flexibility in the construction of solutions, we further introduce, for each role name $r \in \mathsf{Rol}(\Gamma_A^1)$, an auxiliary atom $\exists r.X_r$ with a fresh variable X_r. This allows to construct new existential restrictions over the existing role names in order to find a solution. For example, the definition of Difficulty_writing contains the atom $\exists$Has_interpretation.Able_with_difficulty, which is not an atom of the disunification problem we have constructed so far, and hence cannot appear in any local solution. But now we can use $\exists$Has_interpretation.$X_{\text{Has\_interpretation}}$ instead, since we can map the new variable $X_{\text{Has\_interpretation}}$ to Able_with_difficulty. Formally, Γ_A^2 is obtained by adding the above atoms $\exists r.X_r$, $r \in \mathsf{Rol}(\Gamma_A^1)$, to $\mathsf{At}(\Gamma_A^1)$, e.g., by introducing trivial subsumptions of the form $\exists r.X_r \sqsubseteq^? \exists r.X_r$.

This finally allows us to obtain the original definition of Difficulty_writing as a solution of Γ_A^2. However, it turns out that this problem in general has too many solutions to feasibly

enumerate, in particular since the wanted solution is often not among the first ones computed by UEL. Hence, we need to include additional constraints on the form of the solutions, which are obtained by taking into account the structure of SNOMED CT concept definitions and their design guidelines. In the following sections, we directly present the constraints as they appear in the SAT encoding used by UEL. Hence, we consider the set of propositional clauses $\mathsf{Cl}(\Gamma_A^2)$ encoding Γ_A^2 as described in [BBM16]. We extend this set by additional constraints on the existing propositional variables of the form $[C \sqsubseteq D]$ with $C, D \in \mathsf{At}(\Gamma_A^2)$, in order to prune the set of solutions. Since we do not introduce any more atoms, for brevity we use the notations At, Ex, Var, Rol in the following without explicitly referring to Γ_A^2.

3.4 Domains and Ranges of Roles

The SNOMED CT Editorial Guide specifies so-called *hierarchies*, which group the concept names according to common characteristics. Each hierarchy can be identified by the most general concept name contained in it, which subsumes all other concept names in that hierachy. For example, all subconcepts of Clinical_finding belong to the "finding" hierarchy, which encompasses all kinds of diagnoses. The "disorder" hierarchy is characterized by the direct subconcept Disease of Clinical_finding.

The importance of these hierarchies lies in the specification of contexts in which roles are allowed to occur. For example, the role Interprets can only be used in the definition of subconcepts of Clinical_finding, and its fillers must be subconcepts of either Observable_entity, Laboratory_procedure, or Evaluation_procedure. Note that these are purely syntactic restrictions, and do not correspond to the common notions of *domain and range restrictions* in DLs [BBL08]. Since the input for UEL is given in the form of OWL ontologies, we nevertheless specify these restrictions as OWL *ObjectPropertyDomain* and *ObjectPropertyRange* axioms. For SNOMED CT, we manually extracted them from the SNOMED CT Editorial Guide.

In this way, each role is assigned a domain and a range, which are disjunctions of concept names, and we denote the sets of these concept names by $\mathsf{domain}(r)$ and $\mathsf{range}(r)$, respectively. The only exception is the special role RoleGroup, which has neither domain nor range—it can be used in any subhierarchy of SNOMED CT. We call the concept names occurring in a domain or range restriction *types*, and collect them in the set $\mathcal{T}$. We assume that all types already occur in At; if this is not the case, we drop the superfluous types from all domains and ranges, since they are apparently not relevant for A. We now add the following restrictions:

$$[X \sqsubseteq \exists r.C] \to \bigvee_{D \in \mathsf{domain}(r)} [X \sqsubseteq D] \quad \text{for all } X \in \mathsf{Var} \text{ and } \exists r.C \in \mathsf{Ex}, \tag{3}$$

$$\top \to \bigvee_{D \in \mathsf{range}(r)} [C \sqsubseteq D] \quad \text{for all } \exists r.C \in \mathsf{Ex}, \tag{4}$$

Intuitively, (3) says that, whenever the substitution of a variable contains an existential restriction over a role r, this variable must be of a type compatible with r, i.e., it must belong to one of the subhierarchies of SNOMED CT specified as the domain for r. Similarly, (4) expresses that the filler of an existential restriction over r must belong to a subhierarchy that is in the range of r.

3.5 The Special Case of RoleGroup

Although RoleGroup is not restricted like the other roles, there is a strong relation between the domains of the existential restrictions occurring within a role group and the hierarchy of the concept in which the role group occurs. Since RoleGroup only serves to group together other existential restrictions, the intuition is that it does not affect their domains. Moreover,

certain roles occur within SNOMED CT only inside of RoleGroup restrictions. To enforce this syntactic restriction also in our solutions, we introduce artificial "role group types" (denoted, e.g., Clinical_finding$_{\text{RoleGroup}}$) that denote the scope of a RoleGroup restriction. The domains of all roles (except for Laterality, Has_dose_form and Has_active_ingredient, which never occur in RoleGroup restrictions) are hence modified by replacing all types by the corresponding role group types. We denote by $\mathcal{R}$ the set of all role group types that occur in the domain of any role after this modification.

We construct new propositional variables of the form $\mathsf{type}(C, D_{\text{RoleGroup}})$ to encode that a concept name C has the role group type $D_{\text{RoleGroup}}$. In a first step, we modify the encoding (3) of the domain restrictions for all roles r that can only occur with a RoleGroup-restriction as follows:

$$[X \sqsubseteq \exists r.C] \rightarrow \bigvee_{D_{\text{RoleGroup}} \in \mathsf{domain}(r)} \mathsf{type}(X, D_{\text{RoleGroup}}) \quad \text{for all } X \in \mathsf{Var} \text{ and } \exists r.C \in \mathsf{Ex}. \tag{3'}$$

We then add constraints that "translate" between normal concepts and role group types inside RoleGroup restrictions:

$$[X \sqsubseteq \exists \text{RoleGroup}.C] \wedge \mathsf{type}(C, D_{\text{RoleGroup}}) \rightarrow [X \sqsubseteq D] \quad \text{for all } X \in \mathsf{Var},\ \exists \text{RoleGroup}.C \in \mathsf{Ex}, \text{ and } D_{\text{RoleGroup}} \in \mathcal{R}, \tag{5}$$

$$\top \rightarrow \bigvee_{D_{\text{RoleGroup}} \in \mathcal{R}} \mathsf{type}(C, D_{\text{RoleGroup}}) \quad \text{for all } \exists \text{RoleGroup}.C \in \mathsf{Ex}, \tag{6}$$

$$[X \sqsubseteq C] \wedge \mathsf{type}(X, D_{\text{RoleGroup}}) \rightarrow \bot \quad \text{for all } X \in \mathsf{Var},\ C \in \mathcal{T}, \text{ and } D_{\text{RoleGroup}} \in \mathcal{R} \tag{7}$$

The implication (6) requires that each concept name C occurring within a RoleGroup restriction has some role group type $D_{\text{RoleGroup}}$, and (5) says that $D_{\text{RoleGroup}}$ then determines the position in the subsumption hierarchy of any variable X that contains $\exists$RoleGroup.C in its substitution, i.e., X must lie below D. The converse implication that $[X \sqsubseteq D]$ and $[X \sqsubseteq \exists \text{RoleGroup}.C]$ imply $\mathsf{type}(C, D_{\text{RoleGroup}})$ would not be correct—it is perfectly fine if X has a more specific type D', as long as $D' \sqsubseteq D$ holds. Finally, (7) says that a variable cannot have a role group type and at the same time be part of the ordinary SNOMED CT hierarchy.

3.6 Compatibility of Classes and Types

Additional syntactic restrictions concern the kinds of concept names that can co-occur in the same conjunction. For example, it would not make sense to write the conjunction Language_finding$\sqcap$Able, as this goes against the intuitive interpretation of the conjuncts. A naive idea is to restrict such concept names to belong to the same subhierarchy of SNOMED CT (in the above example, we would allow concepts of the "finding" or the "qualifier value" hierarchy, but not from both). However, this is already violated by many definitions occurring within SNOMED CT itself; for example, the high-level concept Post-surgical_anatomy is declared to be a subconcept of both Effect_of_surgery (from the "morphologic abnormality" hierarchy) and Acquired_body_structure (a "body structure"). Similarly a Drug-device_combination_product belongs to both the "physical object" and the "(pharmaceutical/biologic) product" hierarchies.

To allow such combinations to occur in a solution of our problem, we first extract a binary *compatibility relation* $\mathsf{compatible}$ between all SNOMED CT concept names occurring in Var. The idea is that all concept names C, D that are superconcepts of a common subconcept are marked as compatible. More formally, we define the set $\mathsf{ancestors}(C)$ of all *ancestors* of a concept name C as the closure of $\{C\}$ under applications of $\mathsf{parents}$: $\mathsf{ancestors}(C) := \bigcup_{i \geq 0} \mathsf{parents}^i(\{C\})$, where $\mathsf{parents}(\mathbb{C})$ for a set of concept names $\mathbb{C}$ is defined as the union of all sets $\mathsf{parents}(C)$,

$C \in \mathbb{C}$. Since SNOMED CT is finite, $\mathsf{ancestors}(C)$ can be computed in finite time, and is usually not very large. We now set $(C, D) \in \mathsf{compatible}$ for two concept names $C, D \in \mathsf{Var}$ iff there is a concept name E such that $C, D \in \mathsf{ancestors}(E)$ (this concept name must also occur within Var since we imported all necessary definitions from SNOMED CT; see Section 2.1). The relation $\mathsf{compatible}$ in particular includes pairs of sub- and superconcepts, as well as pairs (C, D) that co-occur in (the top-level conjunction of) the same definition.

We can now use the additional information contained in $\mathsf{compatible}$ to formulate new constraints reflecting the intuition described above, for all $X, C, D \in \mathsf{Var}$ with $(C, D) \notin \mathsf{compatible}$:

$$[X \sqsubseteq C] \wedge [X \sqsubseteq D] \rightarrow \bot \tag{8}$$

$$\mathsf{type}(X, C_{\textsc{RoleGroup}}) \wedge \mathsf{type}(X, D_{\textsc{RoleGroup}}) \rightarrow \bot \quad \text{if } C_{\textsc{RoleGroup}}, D_{\textsc{RoleGroup}} \in \mathcal{R} \tag{9}$$

These two statements enforce that incompatible concepts C, D cannot both occur on the top-level conjunction of the substitution of a variable X (since X is supposed to represent a SNOMED CT concept). Similarly, incompatible concept names should not appear as role group types within role group restrictions (recall that, if X has a role group type, then there must be an existential restriction of the form $\exists \textsc{RoleGroup}.X \in \mathsf{Ex}$).

3.7 UNDEF concept names

The concept names of the form X_{UNDEF} that are introduced for the primitive definitions in SNOMED CT fulfill a special function in our encoding. If we would treat them as variables, they could be substituted by $\top$, and hence we would lose the ability to distinguish sub- and superconcepts; e.g., Clinical_finding and Disease could become equivalent under some solutions. Since we do not want to change the meaning of SNOMED CT, we instead designate them as constants, which allows us to faithfully preserve the subsumption hierarchy of SNOMED CT.

Moreover, we require these concept names to appear only in the context they were originally introduced for, i.e., if $\textsc{Disease}_{\mathsf{UNDEF}}$ appears in the substitution of a variable X, then X must be subsumed by the whole concept Disease (including it superconcepts Clinical_finding, $\textsc{Clinical\_finding}_{\mathsf{UNDEF}}$, etc.):

$$[X \sqsubseteq C_{\mathsf{UNDEF}}] \rightarrow [X \sqsubseteq C] \quad \text{for all } X \in \mathsf{Var},\ C_{\mathsf{UNDEF}} \in \mathsf{At}. \tag{10}$$

3.8 Restricting the Number of Existential Restrictions

As a final restriction on the solutions, we investigated how existential restrictions over the same role usually co-occur in definitions of SNOMED CT. For example, inside of a RoleGroup restriction there may occur several restrictions over Finding_site that denote possible places of a finding within the body. In contrast, for Interprets there is always at most one existential restriction inside a role group. Hence, we restrict our solutions to follow the same behavior.

Additionally, we introduce a parameter k that bounds the number of existential restrictions over RoleGroup that can co-occur in the same conjunction. Sometimes it may be sufficient to look for solutions with at most one role group in them, but for other concepts we may need to allow more. Increasing this parameter increases the number of possible solutions that have to be investigated, and hence we could start with $k = 1$ and successively increase k until we obtain the desired solution.

In the following, we denote by $\mathsf{num}(r)$ the number of existential restrictions over a role r that are allowed to co-occur in the same conjunction, where 0 represents no limit; that is,

$\mathsf{num}(\textsc{Finding\_site}) = 0$, $\mathsf{num}(\textsc{Interprets}) = 1$, $\mathsf{num}(\textsc{RoleGroup}) = k$, etc. We enforce these constraints as follows:

$$[X \sqsubseteq \exists r.C_1] \wedge \cdots \wedge [X \sqsubseteq \exists r.C_n] \rightarrow \bigvee_{\substack{1 \leq i < j \leq n \\ C_i, C_j \in \mathsf{Var}}} [C_i \sqsubseteq C_j] \vee [C_j \sqsubseteq C_i] \vee \bigvee_{\exists r.C' \in \mathsf{Ex}} [X \sqsubseteq \exists r.C'] \wedge [C' \sqsubseteq C_i] \wedge [C' \sqsubseteq C_j] \quad (11)$$

for all $X \in \mathsf{Var}$, $r \in \mathsf{Rol}$ with $n = \mathsf{num}(r) + 1$, and all $\exists r.C_1, \ldots, \exists r.C_n \in \mathsf{Ex}$. That is, whenever there are $\mathsf{num}(r) + 1$ existential restrictions in the top-level conjunction of the substitution of the same variable X, then one of the following cases must hold:

- One existential restriction is redundant, i.e., subsumes a more specific one.
- Two existential restrictions are redundant, i.e., subsume another existential restriction that is also present in the substitution of X.

In both cases, we can obtain an equivalent solution that contains less existential restrictions over r.

4 Evaluation

We implemented the encoding described above in our tool UEL, and executed a preliminary evaluation on SNOMED CT. For the parameter $\mathsf{num}(\textsc{RoleGroup})$, we observed that with only one role group we could miss some solutions that were possible if we allowed two role groups instead, but larger numbers did not yield any improvement. Therefore, our experiments were executed with $\mathsf{num}(\textsc{RoleGroup}) = 2$.

Our first experiments on the full SNOMED CT ontology were not very successful, which is why we also looked at some *submodules*. For this, we chose several concepts C at the root of important subhierarchies in SNOMED CT, such as Body_structure and Clinical_finding (see the left half of Table 2). For each C, we first extracted all definitions of concept names that are subsumed by C, as well as all dependent definitions, i.e., those of concepts used in the definitions of subconcepts of C. The number of concept names in the resulting modules of SNOMED CT are reported in Table 2. Since we try to reconstruct only *full* definitions $A \equiv C_A$ from SNOMED CT, the third column lists the number of all subconcepts of C that have a full definition. Of those concept names, we randomly chose 100 to run UEL with the encoding described in Section 3.

We ran the experiments on a MacBook Pro with 2,2 GHz Intel Core i7 CPU, 16 GB DDR3 RAM at 1600 MHz, and a Java heap limit of 14 GB. We set a timeout of 5 minutes per concept name, and report in Table 2 the number of timeouts, the number of successes (i.e., where we were able to reconstruct C_A) and failures (where no solutions were equivalent to C_A). For successes and failures, we indicate the average time spent until detecting success or failure, and the number of solutions computed until then. In case of success, this number must be at least 1, but does not need to be the total number of solutions to the given problem, whereas failure is also possible with 0 solutions, but cannot be declared until all solutions have been computed. Since the SAT encoding can yield solutions that are equivalent w.r.t. SNOMED CT, we checked for every new solution if it was equivalent to a previously computed one, and did not count such duplicates.

From the results, we can see that only in 6 of 100 cases our approach could reconstruct the original definition of a concept name from SNOMED CT. However, in specific subhierarchies, such as those

Table 2: Evaluation results on SNOMED CT modules. "Size" denotes the number of concept names in the module, "Def." the number of fully defined concept names below the root concept name, "#" is the number of timed-out/successful/failed test cases, "Time" is the average time (in seconds) to completion, and "Sol." is the average number of solutions that were computed.

			T/O	Success			Failure		
Root concept name	Size	Def.	#	#	Time	Sol.	#	Time	Sol.
SNOMED_CT_Concept	325 143	79 468	92	6	19.0	4.7	2	5.0	0.0
└ Body_structure	31 540	1 086	0	99	2.8	2.3	1	22.0	1.0
└ Clinical_finding	126 993	47 225	86	9	18.6	1.2	5	69.8	47.4
│ └ Clin._hist._&_obs._find.	23 446	5 297	27	60	17.1	3.7	13	21.8	1.4
│ │ └ Functional_finding	6 741	3 325	7	85	17.8	4.6	8	44.1	3.1
│ └ Disease	88 594	38 056	89	9	4.2	1.1	2	30.5	0.0
│ │ └ Poisoning	6 738	3 171	40	53	5.2	1.7	7	2.6	11.0
│ └ Finding_by_site	15 149	2 725	87	2	7.0	1.0	11	26.4	21.2
└ Observable_entity	9 243	136	64	0	–	–	36	13.8	0.0
└ Pharm./biol._product	23 678	2 678	82	9	13.9	2.7	9	5.3	0.0
└ Procedure	70 608	23 907	89	2	18.0	4.5	9	34.8	2.7

below Body_structure, Clinical_history_&_observation_finding, or Poisoning, the success rates were much higher. The reason for this is that these hierarchies are well-structured, with a smaller number of children per parent concept name, and a larger number of levels in the subsumption hierarchy. Since we add the definitions of all siblings to the SAT encoding, such a structure prevents the search space from getting too large. In contrast, the hierarchy below Observable_entity is relatively flat and contains mostly primitive definitions, which do not allow to distinguish siblings from each other. For example, the concept names Ability_to_use_non-verbal_communication and Ability_to_use_verbal_communication are both defined simply as subconcepts of Ability_to_communicate, with no existential restrictions to distinguish them. Another problem occurred when siblings were not sufficiently similar to allow to deduce the original definition of a concept name from the definitions of its siblings.

5 Conclusion

We demonstrated that disunification can find definitions of SNOMED CT concept names when given only a general description of their position in the concept hierarchy. An ontology engineer could, for example, specify only the direct superconcepts and some incomparable concepts, in order to obtain the definition of a new concept for SNOMED CT.

References

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proc. of the 4th Workshop on OWL: Experiences and Directions*, pages 1–10, 2008.

[BBM16] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in $\mathcal{EL}$ to disunification: The case of dismatching and local disunification. *Logical Methods in Computer Science*, 12(4:1):1–28, 2016.

[BCM+07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F.

Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2 edition, 2007.

[BM10a] 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.

[BM10b] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3:17):1–31, 2010.

[BN01] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *Journal of Symbolic Computation*, 31(3):277–305, 2001.

[LV14] Jens Lehmann and Johanna Völker. *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*. 2014.

[RI12] Alan Rector and Luigi Iannone. Lexically suggest, logically define: Quality assurance of the use of qualifiers and expected results of post-coordination in SNOMED CT. *Journal of Biomedical Informatics*, 45(2):199–209, 2012.