What's in an Attribute? Consequences for the Least Common Subsumer[*]

Ralf Küsters
LuFG Theoretical Computer Science
RWTH Aachen Ahornstraße 55
52074 Aachen
Germany
kuesters@informatik.rwth-aachen.de

Alex Borgida
Department of Computer Science
Rutgers University
Piscataway, NJ 08855 USA
borgida@cs.rutgers.edu

Abstract

Functional relationships between objects, called "attributes", are of considerable importance in knowledge representation languages, including Description Logics (DLs). A study of the literature indicates that papers have made, often implicitly, different assumptions about the nature of attributes: whether they are always required to have a value, or whether they can be partial functions. The work presented here is the first explicit study of this difference for (sub-)classes of the Classic DL, involving the same-as concept constructor. It is shown that although determining subsumption between concept descriptions has the same complexity (though requiring different algorithms), the story is different in the case of determining the least common subsumer (lcs). For attributes interpreted as partial functions, the lcs exists and can be computed relatively easily; even in this case our results correct and extend three previous papers about the lcs of DLs. In the case where attributes must have a value, the lcs may not exist, and even if it exists it may be of exponential size. Interestingly, it is possible to decide in polynomial time if the lcs exists.

[*]An abridged version has been submitted to JAIR.

Contents

1 Introduction

Knowledge representation systems based on Description Logics (DL systems) have been the subject of continued attention in Artificial Intelligence, both as a subject of theoretical studies (e.g., [2, 11, 14]) and in applications (e.g., [1, 17]). More impressively, DLs have found applications in other areas involving information processing, such as databases [6, 16], semi-structured data [12, 13], information integration [15, 9], as well as more general problems such as configuration [28] and software engineering [7, 21].

In Description Logics, one takes an object-centered view, where the world is modeled as individuals, connected by binary relationships (here called *roles*), and grouped into classes (called *concepts*). In every DL system the concepts of the application domain are described by *concept descriptions* that are built from atomic concepts and roles using the constructors provided by the DL language. For example, using the atomic concepts $Model$, $Manufacturer$, and $INTEGER$ as well as the roles *model*, *seats*, and *madeBy* the concept Car can be described as follows:

$$\begin{array}{rcl} Car & := & madeBy \downarrow model\ madeBy \sqcap \\ & & seats \downarrow model\ seats \sqcap \forall seats.INTEGER \sqcap \\ & & \forall model.Model \sqcap \forall madeBy.Manufacturer \end{array}$$

By using same-as equalities (like $seats \downarrow model\ seats$), we ensure that the manufacturer (number of seats) of a car is the same as the manufacturer (number of seats) of the model of that car.

DLs support a variety of inferences, including deciding if a concept *subsumes* another one. Subsumption algorithms allow one to determine subconcept-superconcept relationships: C is subsumed by D ($C \sqsubseteq D$) if and only if all instances of C are also instances of D, i.e., the first description is always interpreted as a subset of the second description.

The traditional inference problems for DL systems, such as subsumption, inconsistency detection, membership checking, are by now well-investigated. Algorithms and detailed complexity results for realizing such inferences are available for a great variety of DL languages of differing expressive power (e.g., [23, 3, 29, 22, 25]).

In most knowledge representation systems, including DLs, functional relationships, here called *attributes* (in the literature also called features), are distinguished as a subclass of general relationships, at least in part because functional restrictions occur so frequently in practice. In the above example, *madeBy* and *seats* are meant to be attributes, thus making unnecessary number restrictions like $((\leq 1\, madeBy) \sqcap (\geq 1\, madeBy))$. In addition, distinguishing attributes helps identify tractable subsets of DL constructors: in CLASSIC, coreferences between attribute chains (as in the above examples) can be reasoned with efficiently [10], while if we changed to roles (e.g., allowed ($\mathsf{repairs} \downarrow \mathsf{ownedBy} \circ \mathsf{repairsPaidFor}$)), the subsumption problem becomes undecidable [30].

The distinction between roles and attributes in DLs is both theoretically and practically well understood. However, it turns out that there is a distinction to be made between attributes being interpreted as total (*total attributes*) or partial functions (*partial attributes*). This distinction is useful in practice, since there is a difference between a car not having a license plate, and having a license plate whose value is not currently known. In DLs, the latter is modeled by having the attribute hasLicensePlate, with restriction (≥ 1 hasLicensePlate), while the former is modeled with the restriction (≤ 0 hasLicensePlate). If attributes were total functions, the last assertion would immediately lead to a contradiction.

It turns out that the study of the theoretical implications of this distinction has "slipped through the cracks" of contemporary research. The purpose of this paper is to show explicitly and precisely the effect of allowing attributes to be total or partial. Specifically, we show that for one group of DLs related to the CLASSIC system, although this distinction does not affect the complexity of computing subsumption (the details of the algorithm do need to be changed), it does have a significant impact on the problem of computing the least common subsumer (lcs) of concepts, i.e., the most specific concept description subsuming a set of given concepts.

Since, as detailed below, a number of published papers on the above topic made differing assumptions about the nature of attributes, but did not highlite these differences, the more general lesson we want to impart is that in all knowledge representation schemes and investigations it is very important to be clear about whether attributes are total or partial.

Least common subsumer

The lcs was first introduced as a new inference problem in DLs in [20]. One motivation for considering the lcs is to use it as an alternative to disjunction. The idea is to replace disjunctions like $C_1 \sqcup \cdots \sqcup C_n$ by the lcs of $C_1, \dots, C_n$. In [8, 20], this operation is called *knowledge-base vivification.* Although, in general, the lcs is not equivalent to the corresponding disjunction, it is the best approximation of the disjunctive concept within the available language. Using such an approximation is motivated by the fact that, in many cases, adding disjunction would increase the complexity of reasoning.[1]

As proposed in [4, 5], the lcs operation can be used to support the "bottom-up" construction of DL knowledge bases, where, roughly speaking, starting from "typical" examples an lcs algorithm is used to compute a concept description that (i) contains all these examples, and (ii) is the most specific description satisfying property (i). In [4], such an algorithm has been presented for cyclic $\mathcal{ALN}$-concept descriptions; $\mathcal{ALN}$ is a sublanguage of Classic allowing for concept conjunction, primitive negation, value restrictions, and number restrictions. Baader et al. [5] have proposed an lcs algorithm for a DL allowing for existential restrictions instead of number restrictions.

Originally, the lcs was introduced as an operation in the context of inductive learning from examples [20], and several papers followed up this lead. The DLs considered were mostly sublanguages of Classic which allowed for same-as equalities, i.e., expressions like $madeBy \downarrow model \circ madeBy$. Cohen et al. [20] proposed an lcs algorithm for $\mathcal{ALN}$ and a language that allows for concept conjunction and same-as, which we will call $\mathcal{S}$. In [18], the algorithm for $\mathcal{S}$ was extended to CoreClassic, which additionally allows for value restrictions (see [19] for experimental results). Finally, Frazier and Pitt [24] presented an lcs algorithm for full Classic.

All these algorithms are based on a translation of concept descriptions into so-called description graphs, which had been used in [10] to decide subsumption. More precisely, the lcs is computed in three steps: First, the concept descriptions are turned into description graphs. Second, the lcs is computed as the product of the description graphs. Finally, the product graph thus obtained is turned back into a concept description, representing the lcs of the given concepts.

However, there is a mismatch between the semantics of attributes underlying the subsumption algorithm on the one hand and the lcs algorithms on the other hand. In particular, in the work of Borgida and Patel-Schneider [10], attributes are interpreted as total functions, whereas a careful examination of the lcs algorithms proposed, especially the ones involving same-as, reveals that the lcs is computed for DLs with partial attributes. Furthermore, it turns out that the lcs algorithm presented by Frazier and Pitt [24] does not handle properly inconsistency, which can be expressed in full Classic.

[1] Observe that if the language already allows for disjunction, we have $lcs(C_1, \ldots, C_n) \equiv C_1 \sqcup \cdots \sqcup C_n$. In particular, this means that, for such languages, the lcs is not really of interest.

New results

In Section 3, we provide a subsumption algorithm for full Classic with partial attributes, which essential extends $\mathcal{ALN}$ by same-as equalities, but also allows for the fills and one-of constructors on (host) individuals. This algorithm is a modification of the corresponding algorithm for the total attribute case presented in [10]. Then we present an lcs algorithm for this language along the lines of [18], and formally prove its correctness using the subsumption algorithm specified before. It turns out that, as in Core-Classic, the lcs always exists, and, for two concept descriptions, it can be computed in time polynomial in their size.

Finally, the central new results of this paper examines the problem of computing lcs when attributes are total (Section 5). The surprising result is that in this case the lcs does not exist in general for the language $\mathcal{S}$ — the construction in [20] being actually for partial attributes. We do however provide a polynomial-time algorithm for deciding when the lcs exists, and a (necessarily) worst-case exponential-time algorithm for computing the lcs in case it exists.

We start by introducing the basic notions necessary for our investigations.

2 Formal Preliminaries

Following [10], in this section we formally introduce the (full) Classic language except for the non-declarative *test-defined concepts* employed in the Classic system to algorithmically approximate representations for ideas that cannot be encoded using the constructors provided by Classic.

Concept descriptions in Classic are built up from a collection of concept names, role names, attribute names, and individuals. Roles, attributes, and individuals are always atomic but descriptions can be built up using constructors. Classic incorporates two different kinds of concept descriptions, namely, host concept descriptions and classic concept descriptions. Therefore, we distinguish host concept names and classic (or atomic) concept names as well as host individuals and classic individuals.

Host concept descriptions are used to describe objects in a concrete domain, e.g., a programming language. A general scheme for incorporating such host objects has been presented in [3].

Host concept descriptions in Classic are relatively simple. They are defined built up from host concepts names and host individuals. More precisely, such concepts have the following syntax:

Syntax	Constructor Name
$\top_H$	top concept of the host domain
E	host concept (name)
$\{I_1 \ldots I_n\}$	one of
$C \sqcap D$	concept conjunction

where $I_1, \ldots, I_n$ are host individuals and C and D are host concept descriptions.

The semantics of host concepts and host individuals is predefined and fixed. The extension of a host concept is a subset of Δ_H, the so-called *host realm.* A host individual is interpreted as an element in Δ_H where different individuals are assigned to different elements of the host realm (*unique name assumption*). The elements in the host realm have no role or attribute successors. However, they can be successors of elements in the classic realm (see below). Finally, we require that (i) all host concept names have an extension that is either of infinite size or is empty; and (ii) that if the extensions of two host concepts overlap, then one must be subsumed by the other (i.e., host concept names are disjoint, unless they are subconcepts of each other). and (iii) that the difference $(A - B)$ of the extensions of two host concept names is either infinite or empty. (These conditions are needed to avoid being able to infer conclusions from the size of host descriptions.) This, for example, allows for host concepts like $INTEGER$, $REAL$, $COMPLEX$, and $STRING$, but not $BOOLEAN$.

Classic concept descriptions in CLASSIC allow for more complex concepts than host concept descriptions. They are formed according to the following syntax:

Syntax	Constructor Name
$\top_C$	top concept of the classic realm
E	classic/atomic concept name
$(\geq n\, R)$	at least restriction
$(\leq m\, R)$	at most restriction
$R : I$	fills (on a role)
$A : I$	fills (on an attribute)
$\{I_1 \ldots I_n\}$	one of
$A_1 \cdots A_k \downarrow B_1 \cdots B_h$	sams-as equality
$C \sqcap D$	concept conjunction
$\forall R.C$	(role) value restriction
$\forall A.C$	(attribute) value restriction

where E is an atomic concept name; R is a role; A, A_i, and B_j are attributes; $\mathcal{I}$ is the name of a classic or host individual; I_j are names of classic individuals; C and D are classic concept descriptions; F is a host or classic concept description; and k, h, m, n are non-negative integers.

A description which is either a host concept description or a classic concept description is called (Classic) *concept description.* In addition, a concept description might be the *top concept* $\top$, which will be interpreted as the whole domain.

The sublanguage $\mathcal{S}$ of Classic only allows for same-as and concept conjunction.

In the sequel, the set of concept names (host and classic) is called $\mathcal{C}$, the set of role names $\mathcal{R}$, the set of attributes $\mathcal{A}$, and the set of individuals $\mathcal{IND}$ (which consists of the set of host individuals $\mathcal{IND}_{\mathcal{H}}$ and classic individuals $\mathcal{IND}_{\mathcal{C}}$). All these sets are pairwise disjoint. Furthermore, we will use $\forall R_1 \cdots R_n.C$ as abbreviation of $\forall R_1.\forall R_2 \cdots \forall R_n.C$ where $\forall \varepsilon.C$ denotes C.

As argued in [10], for individuals it is reasonable to introduce a non-standard semantics. The most significant reason is to avoid intractability of subsumption. It has been shown in [10, 27] that subsumption in Classic is NP-complete when individuals are interpreted as single elements of the domain.

To avoid this problem the semantics of individuals are defined as follows: instead of mapping classic individuals onto single elements of the domain, as we have done it for host individuals or as it is done in standard semantics, classic individuals are mapped onto *disjoint subsets of the domain*, intuitively representing different possible realizations of that (Platonic) individual.

For a given interpretation $\mathcal{I}$ the classic individuals induce the following *congruence relation* over the domain $dom(\mathcal{I})$ of $\mathcal{I}$: two elements in $dom(\mathcal{I})$ are said to be congruent if and only if they belong to the same extension of the same individual $I \in \mathcal{IND}_{\mathcal{C}}$ or if they are identical. The *cardinality of a set of elements of the domain* is then the size of the set modulo this congruence relationship.

As usual, the denotational semantics for concept descriptions is recursively built on the extensions of atomic identifiers (i.e., of concept names, role names, attribute names, and individuals) by an interpretation:

Definition 1 *An interpretation $\mathcal{I}$ consists of a domain Δ and an interpretation function $\cdot^{\mathcal{I}}$. The domain is disjointly divided into a classic realm Δ_C and the host realm Δ_H which is fixed for all interpretations. The interpretation function assigns extensions to atomic identifiers as follows:*

- *The extension of an atomic concept name E is some subset $E^{\mathcal{I}}$ of the classic realm.*

- *The extension of a host concept name H is some predefined subset of Δ_H which is fixed for all interpretations and satisfies the conditions stated above.*

- *The extension of an atomic role name R is some subset $R^{\mathcal{I}}$ of $\Delta_C \times \Delta$.*

- *The extension of an atomic attribute name A is some partial function $A^{\mathcal{I}}$ from Δ_C to Δ, i.e., if $(x, y_1) \in A^{\mathcal{I}}$ and $(x, y_2) \in A^{\mathcal{I}}$ then $y_1 = y_2$.*

- *The extension of a classic individual I is some non-empty subset $I^{\mathcal{I}}$ of Δ_C where the interpretations of distinct identifiers must be disjoint as discussed above.*

- *Host individuals are interpreted according to the unique name assumption (see above). The extension $I^{\mathcal{I}}$ of a host individual I is an element in Δ_H. As already mentioned, the interpretation of the host individuals is fixed for all interpretations. Therefore, we occasionally refer to $I^{\mathcal{I}}$ as I.*

By $(R_1 \cdots R_n)^{\mathcal{I}}$ for role names or attributes R_i we denote the composite of the binary relations $R_i^{\mathcal{I}}$. If $n = 0$ then $\varepsilon^{\mathcal{I}}$ denotes the identical relation, i.e., $\varepsilon^{\mathcal{I}} := \{(d, d) \mid d \in \Delta_C\}$. For an individual d, we define $R^{\mathcal{I}}(d) := \{e \mid (d, e) \in R^{\mathcal{I}}\}$. If the R_i are attributes, we say that $(R_1 \cdots R_n)^{\mathcal{I}}$ is defined *for d iff $(R_1 \cdots R_n)^{\mathcal{I}}(d) \neq \emptyset$; occasionally, we will refer to the image of d by $(R_1 \cdots R_n)^{\mathcal{I}}(d)$.*

The extension $C^{\mathcal{I}}$ of a concept description C is inductively defined as follows:

- $\top^{\mathcal{I}} = \Delta$; $\top_C^{\mathcal{I}} = \Delta_C$; $\top_H^{\mathcal{I}} = \Delta_H$;

- $p : I^{\mathcal{I}} = \{d \in \Delta_C \mid \exists x \ (d, x) \in p^{\mathcal{I}} \ \wedge \ x \in I^{\mathcal{I}}\}$ *where p is a role or an attribute;*

- $\{I_1 \, \ldots \, I_n\}^{\mathcal{I}} = \bigcup_{k=1}^{n} I_k^{\mathcal{I}}$. *If the $I_k's$ are all host individuals then this means* $\{I_1 \, \ldots \, I_n\}^{\mathcal{I}} = \{I_1 \, \ldots \, I_n\}$;

- $(\geq n\, R)^{\mathcal{I}}$ *(resp. $(\leq n\, R)^{\mathcal{I}}$) is those objects in Δ_C with at least (resp. at most) n* non-congruent *successors for the role R;*

- $(A_1 \cdots A_k \downarrow B_1 \cdots B_h)^{\mathcal{I}} = \{d \in \Delta_C \mid (A_1 \cdots A_k)^{\mathcal{I}}$ *and* $(B_1 \cdots B_h)^{\mathcal{I}}$ *are defined for d and* $A_1 \cdots A_k{}^{\mathcal{I}}(d) \ = \ B_1 \cdots B_h{}^{\mathcal{I}}(d)\}$;

- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$*;*
- $(\forall p.C)^{\mathcal{I}} = \{d \in \Delta_C \mid p^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$ *where p is a role or an attribute.*

If we had not restricted the interpretation of the constructors like value-restriction and number restriction to subsets of Δ_C then subsumption relationships as $INTEGER \sqsubseteq \forall R.C$ or $INTEGER \sqsubseteq (\leq n\, R)$ would hold.

Note that the above definition supports partial attributes. Since the main point of this paper is to demonstrate the impact of different semantics for attributes, we occasionally restrict the set of interpretations to those that map attributes to *total* functions. Such interpretations are called *t-interpretations.*

The main inference service provided by a system based on description logics is to compute subconcept-superconcept relationships, which are also called subsumption relationships.

Definition 2 *A concept description C is* subsumed *by the concept description D ($C \sqsubseteq D$ for short) if and only if for all interpretations $\mathcal{I}$ it is $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. If we consider only total interpretations, we get* t-subsumption*: $C \sqsubseteq_t D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all t-interpretations $\mathcal{I}$.*

Having defined subsumption, equivalence of concept descriptions is defined in the usual way: $C \equiv D$ if and only if $C \sqsubseteq D$ and $D \sqsubseteq C$. Analogously, t-equivalence $C \equiv_t D$ is specified.

Although, Classic as introduced here does not contain the bottom concept $\perp$ explicitly, it can be expressed by, e.g., $(\geq 1\, R) \sqcap (\leq 0\, R)$. We will use $\perp$ as abbreviation of some inconsistent concept description. Furthermore, primitive negation can be expressed by number restrictions. For an atomic concept E one can replace every occurrence of E by $\geq 1 R_E$ and the negation $\neg E$ of E by $\leq 0 R_E$ where R_E is a new role name. We also do not allow for number restrictions on attributes. However, $(\geq n\, A) \equiv \perp$ and $(\leq n\, A) \equiv \top_C$ for every $n \geq 2$. Moreover, $(\leq 0 A) \equiv \forall A.\perp$, $(\leq 1 A) \equiv \top_C$, $(\geq 0 A) \equiv \top_C$, and $(\geq 1 A) \equiv A \downarrow A$. Usually, Classic also allows for *min* $n := \{m \mid\ m$ is a nonnegative integer greater or equal $n\}$ and *max* $n := \{m \mid\ m$ is a nonnegative integer less or equal $n\}$ where n is a nonnegative integer. This constructor can be simulated by introducing a host concept name NON-NEGATIVE-INTEGER which is interpreted as the nonnegative integers and an infinite set of host individuals $0, 1, 2, \ldots$ Then, for example, *min* n is equivalent to the concept description $\{n, n+1, n+2, \ldots\}$ (infinite one-of) which can be represented finitely. To sum up, the results presented in this paper, can easily be extended to a language that in addition to the constructors introduced for CLASSIC also allows for the bottom concept, primitive negation, number restrictions on attributes, and *min n*, *max n*.

The least common subsumer of a set of concept descriptions is the most specific concept subsuming all concept descriptions of the set:

Definition 3 *The concept description D is the* least common subsumer (lcs) *of the concept descriptions $C_1, \dots, C_n$ ($lcs(C_1, \dots, C_n)$ for short) iff i) $C_i \sqsubseteq D$ for all $i = 1, \dots, n$ and ii) for every D' with that property $D \sqsubseteq D'$. Analogously, we define $lcs_t(C_1, \dots, C_n)$ using $\sqsubseteq_t$ instead of $\sqsubseteq$.*

Note that the lcs of concept descriptions may not exist, but if it does, by definition it is uniquely determined up to equivalence. In this sense, we may refer to *the* lcs.

In the following two sections, attributes are always interpreted as partial functions; only in Section 5 we consider total attributes.

3 Subsumption

As proposed in [10] subsumption is decided by a multi-part process. First, descriptions are turned into description graphs. Next, description graphs are put into canonical form, where certain inferences are explicated and other redundancies are reduced by combining nodes and edges in the graph. Finally, subsumption is determined between a description and a canonical description graph.

Since in [10] attributes are interpreted as total functions, we need to adjust the steps listed above to decide subsumption in the case of partial attributes. However, we have tried to reduce the changes necessary. For this reason, roughly speaking, attributes are treated as roles unless they form part of a same-as equality. (Note that attributes participating in a same-as construct must have values!) To some extent, this will allow us to adopt the semantics of the original description graphs, which is crucial for proofs. However, the two different occurrences of attributes, namely, in a same-as equality vs. a role in a value-restriction, require us to modify and extend the definition of description graphs, the normalization rules, and the subsumption algorithm itself. Furthermore, the subsumption algorithm proposed in [10] was still incomplete because some normalization rules were missing, e.g., for dealing with singleton sets of host individuals.

In the following, we will present the steps of the subsumption algorithm in detail. We start with the definition of description graphs.

3.1 Description Graphs

Intuitively, description graphs reflect the syntactic structure of concept descriptions. A description graph is a labeled, directed multigraph, with a distinguished node. Roughly speaking, the edges (a-edges) of the graph capture the constraints expressed by same-as equalities. The nodes are labeled with a set of so-called r-edges which correspond to value restrictions. These r-edges lead to description graphs again which represent the concept descriptions of the corresponding value restrictions. Unlike the graphs proposed in [10], the value restrictions represented by nodes not only contain restrictions on roles, as in [10], but also on attributes. We shall comment on the advantage of this modification in order to deal with attributes as partial functions instead of total functions.

Before defining description graphs formally, we will look at our example concept Car. The description graph depicted in Figure 1 corresponds to C. We use $G(Manufacturer)$, $G(Model)$, $G(INTEGER)$ to denote the descriptions graphs for the atomic concept names $Manufacturer$ and $Model$ as well as the host concept name $INTEGER$. In this case, these description graphs are very simple; they merely consist of one node labeled with the corresponding concept name. In general, the description graphs in r-edges are more complicated since in a value restriction like $\forall R.C$, C is an arbitrary concept description.

For the sake of simplicity, we have omitted the components of the nodes and edges corresponding to the one of constructor and fills.

Although, the concept Car does not have number restrictions, the corresponding graph has the restriction $[0, 1]$ on the r-edges since in our example these edges are restrictions on attributes, which have at most one direct successor. By adding these additional informations for attributes it is possible to treat attributes like roles, unless no successors by fills or a same-as equality are required. In our example, there are same-as restrictions on the attributes. As we will see later, it is necessary to "lift" the r-edges to a-edges in order to get a complete subsumption algorithm. This normalization operation was not necessary for the graphs defined in [10] since attributes were not allowed in r-edges.

Formally, description graphs are defined as follows:

Definition 4 *A* description graph *G is a tuple (N, E, r, l), consisting of a finite set N of* nodes*; a finite set E of edges (*a-edges*); a* distinguished node *r in N (*root of the graph*); and a function l from N into the set of labels of nodes. We will occasionally use the notation $G.Nodes$, $G.Edges$, and $G.root$ to access the components N, E and r of the graph G.*

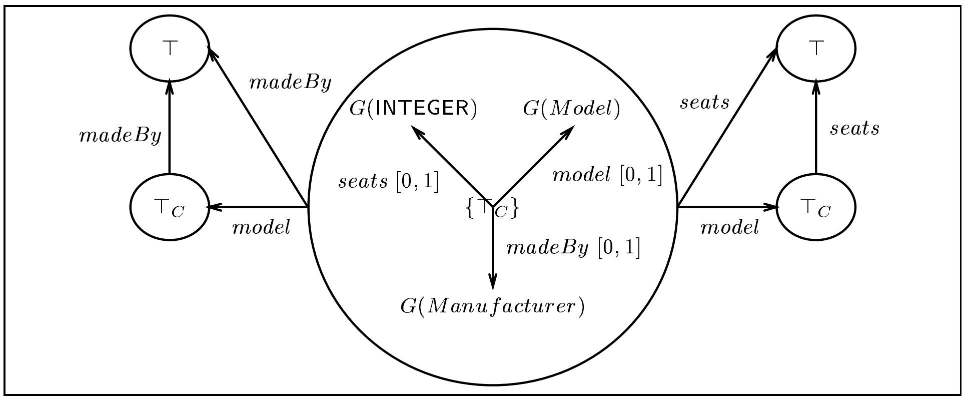

Figure 1: The description graph for Car where the node in the middle is the root of the graph

An a-edge *is a tuple of the form* (n_1, n_2, A, F) *where* n_1*,* n_2 *are nodes,* A *is an attribute name, and* F *is a set of individuals (the* fillers *of the node).*

A label of a node *is defined to be* $\perp$ *or a tuple of the form* (C, D, H)*, consisting of a finite set* C *of concept names (the* atoms *of the node), a finite set* D *of classic individuals or* $\top$ *(the* dom *of the node), and a finite set* H *of tuples (the r-edges of the node).* Concept names *in a description graph are atomic concept names, host concept names,* $\top$*,* $\top_C$*, or* $\top_H$*. We will occasionally use the notation* $n.Atoms$*,* $n.Dom$*, and* $n.REdges$ *to access the components* C*,* D*, and* H *of the node* n*.*

An r-edge *is a tuple,* (R, m, M, F, G')*, of a role or an attribute name,* R*; a min,* m*, which is a non-negative integer; a max,* M*, which is a non-negative integer or* ∞*; a finite set* F *of individuals (the* fillers *of the r-edge); and a (recursively nested) description graph* G'*. The graph* G' *will often be called the* restriction graph *of the node for the role* R*. We assume that the nodes of* G' *are distinct from all nodes of* G *and from all other nested description graphs of* G*. If* R *is an attribute then we require* $m = 0$*,* $M \in \{0, 1\}$*, and* $F = \emptyset$*.*

A path p in G from the node n_0 to n_m is a sequence of a-edges of the form $(n_0, n_1, A_1, F_1), (n_1, n_2, A_2, F_2) \ldots, (n_{m-1}, n_m, A_m, F_m)$ where $m \geq 0$ (for $m = 0$ the *path* p *is empty*); $w = A_1 \cdots A_m$ is called *attribute-label* of p (the empty path has attribute-label ε).

For $n \in N$ we define $G_{|n}$ to be the description graph (N, E, n, l).

In order to merge description graphs we need the notion of "recursive set of nodes" of a description graph G: The *recursive set of nodes of* G is the union of the nodes of G and the recursive set of nodes of all description graphs in the r-edges of nodes in G.

Just as for concept descriptions, the semantics of description graphs is defined by means of an interpretation $\mathcal{I}$. We introduce a function Υ which assigns an individual of the domain of $\mathcal{I}$ to every node of the graph. This ensures that all same-as equalities are satisfied.

Definition 5 *Let* $G = (N, E, r, l)$ *be a description graph and let* $\mathcal{I}$ *be an interpretation.*

An element, d*, of* Δ *is in* $G^{\mathcal{I}}$*, iff there is some function,* Υ*, from* N *into* Δ *such that*

1. $d = \Upsilon(r)$;
2. *for all* $n \in N$ *it is* $\Upsilon(n) \in n^{\mathcal{I}}$;
3. *for all* $(n_1, n_2, A, F) \in E$ *we have* $(\Upsilon(n_1), \Upsilon(n_2)) \in A^{\mathcal{I}}$*, and for all* $f \in F$, $\Upsilon(n_2) \in f^{\mathcal{I}}$.

The extension of a node n *with label* $\perp$ *is the empty set. An element,* d*, of* Δ *is in* $n^{\mathcal{I}}$*, where* $n = (C, D, H)$*, iff*

1. *for all* $B \in C$*, we have* $d \in B^{\mathcal{I}}$;
2. *If* D *is not* $\top$ *then there exists* $f \in D$ *such that* $d \in f^{\mathcal{I}}$.
3. *for all* $(R, m, M, F, G') \in H$,
   (a) *there are between* m *and* M *elements,* d'*, of the domain such that* $(d, d') \in R^{\mathcal{I}}$;
   (b) *for all* $f \in F$ *there is a domain element,* d'*, such that* $(d, d') \in R^{\mathcal{I}}$ *and* $d' \in f^{\mathcal{I}}$*; and*
   (c) $d' \in G'^{\mathcal{I}}$ *for all* d' *such that* $(d, d') \in R^{\mathcal{I}}$

The semantics of the graphs in [10] has been defined in the same way. However, in their paper not only same-as equalities have been expressed by a-edges but also value restrictions on attributes. But then, in the context of partial functions, we could not define the semantics of description graphs by means of the function Υ since individuals need not to have successors for attributes. For that purpose, value restrictions of attributes are always translated into r-edges. The next section will present the translation of concept description into description graphs in detail.

Having defined the semantics of description graphs, subsumption and equivalence between description graphs (e.g., $H \sqsubseteq G$) as well as concept descriptions and description graphs (e.g., $C \sqsubseteq G$) is defined in the same way as subsumption and equivalence between concept descriptions.

3.2 Translating Concept Descriptions to Description Graphs

A Classic description is turned into a description graph by a recursive process, working from the "inside out". In this process, nodes and description graphs are often merged.

Definition 6 *The* merge of two nodes*, $n_1 \oplus n_2$, is a new node n with the following label: if n_1 or n_2 has label $\bot$ then the label of n is $\bot$. Otherwise if both labels are not equal to $\bot$ then the atoms of n are the union of the atoms of n_1 and n_2; the* dom *is the* dom *of n_2, if the* dom *of n_1 is $\top$ and vice versa; otherwise, if both the* dom *of n_1 and n_2 are not equal to $\top$ then* dom *of n is the intersection of the* dom *of n_1 and n_2; the set of r-edges is the union of the r-edges of the two nodes.*

Definition 7 *Let G_1 and G_2 be two description graphs with disjoint recursive sets of nodes (see above). Then the merge of G_1 and G_2, $G := G_1 \oplus G_2$, is defined as follows: The nodes of G are the union of the nodes of G_1 and G_2 without the roots of G_1 and G_2 but with an additional node r. The a-edges of the merged graphs are the union of the a-edges of G_1 and G_2, except that edges touching on the roots of G_1 and G_2 are modified to touch r, i.e., in all a-edges of G_1 and G_2 the roots are replaced by r. The new node r is the root of G and its label is defined by the merge of the two root nodes of G_1 and G_2.*

The rules for translating a description C in Classic into a description graph $G(C)$ are as follows:

1. $\top$ ($\top_C$ or $\top_H$) is turned into a description graphs with one node r and no a-edges. The only atom of r is $\top$ ($\top_C$ or $\top_H$); the *dom* of r is $\top$; and the set of r-edges is empty.

2. $\bot$ is turned into a description graph with one node r and no a-edges. The label of r is $\bot$.

3. A concept name is turned into a description graph with one node and no a-edges. The atoms of the node contain only the concept name; *dom* is $\top$; and the node has no r-edges.

4. A description of the form $(\geq n\ R)$ is turned into a description graph with one node and no a-edges. The node has as its atoms $\top_C$; *dom* is $\top$; and it has a single r-edge $(R, n, \infty, \emptyset, G(\top))$. s

5. A description of the form $(\leq n\ R)$ is turned into a description graph with one node and no a-edges. The node has as its atom $\top_C$; *dom* is $\top$, and it has a single r-edge $(R, 0, n, \emptyset, G(\top))$.

6. A description of the form $R : I$ is turned into a description graph with one node and no a-edges. The node has as its atom $\top_C$, as *dom* $\top$, and it has a single r-edge $(R, 0, \infty, \{I\}, G(\top_C))$ if I is a classic individual, and $(R, 0, \infty, \{I\}, G(\top_H))$ otherwise.

7. A description of the form $A : I$ is turned into a description graph with two nodes r, n, and the a-edges $(r, n, A, \{I\})$ where r denotes the root of the graph. The atom of r is $\top_C$; *dom* is $\top$; and r has no r-edges. The atom of n is $\top_C$ if I is a classic individual and $\top_H$ otherwise; *dom* is $\top$; and n has no r-edges.

8. A description of the form $\{I_1\ \ldots\ I_n\}$ is turned into a description graph with one node. The node has as *dom* the set containing I_1 through I_n, and no r-edges. The only atom of the node is $\top_H$ if all of the individuals are host values, and $\top_C$ if all of the individuals are classic individual names. (Note that classic and host individuals can not be together in one set.)

9. A description of the form $A_1 \cdots A_n \downarrow B_1 \cdots B_m$ is turned into a graph with the pairwise distinct nodes $a_1, \ldots, a_{n-1}, b_1, \ldots, b_{m-1}$, the root $a_0 = b_0 = r$, and an additional node $a_n = b_m = e$; the set of a-edges consists of $(a_0, a_1, A_1, \emptyset)$, $(a_1, a_2, A_2, \emptyset)$,. . . ,$(, a_{n-1}, a_n, A_n, \emptyset)$ and $(b_0, b_1, B_1, \emptyset)$, $(b_1, b_2, B_2, \emptyset)$, $\ldots$, $(b_{m-1}, b_m, B_m, \emptyset)$, i.e., two disjoint paths from r to e. (Note that for $n = 0$ the first path is the empty path from r to r and for $m = 0$ the second path is the empty path from r to r.) All nodes except e have $\top_C$ as there only atom. If $r \neq e$ then the atom of e is $\top$. Finally, the domain of all nodes is $\top$, and there are no r-edges.

10. A description of the form $\forall R.C$, where R is a role, is turned into a description graph with one node and no a-edges. The node has the atom $\{\top_C\}$, its *dom* is $\top$, and it has a single r-edge $(R, 0, \infty, \emptyset, G(C))$.

11. A description of the form $\forall A.C$, where A is an attribute, is turned into a description graph with one node and no a-edges. The node has the atom $\{\top_C\}$, its *dom* is $\top$, and it has a single r-edge $(A, 0, 1, \emptyset, G(C))$.[2]

12. To turn a description of the form $C \sqcap D$ into a description graph, construct $G(C)$ and $G(D)$ and merge them.

Note that this translation is well-defined since it ensured that for every r-edge containing an attribute *min* is 0 and $max = 1 \in \{0, 1\}$.

Figure 1 shows the description graph for the concept description *Car* of our example.

Now we want to show that this process preserves extensions. As we use the merge operations we first show that they work correctly.

Lemma 1 *If n_1 and n_2 are nodes then $(n_1 \oplus n_2)^{\mathcal{I}} = n_1^{\mathcal{I}} \cap n_2^{\mathcal{I}}$. If D_1 and D_2 are description graphs then $(D_1 \oplus D_2)^{\mathcal{I}} = D_1^{\mathcal{I}} \cap D_2^{\mathcal{I}}$.*

Proof: It is easy to see that the claim is true for nodes if one of them has label $\perp$. Otherwise, if both labels are not $\perp$ then atoms and r-edges of the merged node are obtained by unioning the components of the respective nodes; the *dom* of the new node is the intersection of *dom* of n_1 and n_2 if both are not $\top$, otherwise the *dom* of the node which is not $\top$ or it is $\top$ if both nodes have *dom* $\top$. This implies that the interpretation of each component of the new node is the intersection of the interpretation of the corresponding component of n_1 and n_2. Furthermore, since the interpretation of a node is the intersection of the interpretations of its components, the result is obviously true for nodes.

For merging graphs, the only difference is that the root nodes are replaced by their merger in all edges and that the new root is the merger of the roots of the merged graphs. But then an element of $(D_1 \oplus D_2)^{\mathcal{I}}$ is clearly an element of both $D_1^{\mathcal{I}}$ and $D_2^{\mathcal{I}}$. Conversely, since we take the *disjoint* union of the other nodes in the two graphs, the mapping functions Υ_1 and Υ_2 in Definition 5 can simply be unioned, so that an element of both $D_1^{\mathcal{I}}$ and $D_2^{\mathcal{I}}$ is an element of the merged root node, and hence of $(D_1 \oplus D_2)^{\mathcal{I}}$. ■

Theorem 1 *A concept description C and its corresponding description graph $G(C)$ are equivalent, i.e., $C \equiv G(C)$.*

Proof: The proof is by structural induction on concept descriptions.

[2]In [10], the concept description $\forall A.C$ would be turned into an a-edge. As already mentioned, this would cause problems for attributes interpreted as partial functions when defining the semantics by means of Υ as specified in Definition 5.

The extension of concept names, fills, one of, number restrictions, and $\forall$-restrictions on roles and attributes can be easily seen to agree with the extension of description graphs formed from them. Lemma 1 shows that conjunction is properly handled.

For same-as equalities $A_1 \cdots A_n \downarrow B_1 \cdots B_m$ the construction forms a description graph with two disjoint paths from the distinguished node r to an end node e, one labeled by the A_i's, through nodes a_i, and the other labeled by the B_j's, through nodes b_j. If $d \in (A_1 \cdots A_n \downarrow B_1 \cdots B_m)^{\mathcal{I}}$ then defining $\Upsilon(a_i) := (A_1 \cdots A_i)^{\mathcal{I}}(d)$, and $\Upsilon(b_j) := (B_1 \cdots B_j)^{\mathcal{I}}(d)$, yields the mapping required by Definition 5. The converse is satisfied by the requirement in Definition 5 that for each a-edge $(n_1, n_2, A, F) \in E$, we have $(\Upsilon(n_1), \Upsilon(n_2)) \in A^{\mathcal{I}}$. ∎

3.3 Translating a Description Graph to a Concept Description

Although, we do not need the converse translation from description graphs to concept descriptions for characterizing subsumption, the translation is presented here already in order to show that concept descriptions and description graphs are equivalent representations. Later on, when discussing the lcs, we will actually need to translate description graphs into concept descriptions.

In the sequel, let $G = (N, E, r, l)$ be a description graph. W.l.o.g. we assume that G and (recursively) all description graphs nested in G are connected. A description graph is said to be *connected* if all nodes of the graph can be reached from the root of the graph by a directed path and if all nested graphs are connected. Note that the semantics of a graph is not changed, if nodes that are not connected via a path with the root are deleted.

We now (recursively) specify C_G which corresponds to the description graph G using the concept description C_n corresponding to the label of the node n in G.

Let n be a node in G. If the label of n is $\perp$, then $C_n := \perp$. Now, let (S, D, H) be the label of n. Then, C_n is a conjunction consisting of the following conjuncts:

1. atomic concepts: If $S = \emptyset$, then $\top$; otherwise
$$\bigsqcap_{E \in S} E.$$
2. one-of: If $D = \top$, then $\top$; if $D = \emptyset$, then $\perp$; otherwise D.

3. r-edges: If $H = \emptyset$, then $\top$; otherwise

$$\bigsqcap_{(R,l,k,F,G')\in H} \left[\forall R.C_{G'} \sqcap (\geq l\ R) \sqcap (\leq k\ R) \sqcap \bigsqcap_{f\in F}(R : f) \right].$$

If R is an attribute, then we know $l = 0$, $k \in \{0,1\}$, and $F = \emptyset$. In this case, we define $(\geq l\ R)$ to be $\top$. Furthermore, we define $(\leq k\ R)$ to be $\top$ if $k = 1$ and $\forall R.\bot$ otherwise.

Using C_n, C_G is defined as follows:

1. Same-as: Let T be a spanning tree of G. (Note that because G is connected, T contains all nodes of G.) For every leave of T, C_G has a same-as equality $v \downarrow v$ where v is the attribute-label of the rooted path in T to the leave. Furthermore, for every a-edge (n_1, n_2, A, F) not contained in T we have $v \downarrow w$ in the conjunction where v and w are defined as follows: w is the attribute-label of the rooted path in T to n_2; v is the attribute-label of the path in T to n_1 concatenated with the a-edge (n_1, n_2, A, F) from n_1 to n_2.

2. nodes and a-edges: For every node n in T we have

$$\forall al(n). \left[C_n \sqcap \bigsqcap_{(n,m,A,F)\in E} \left(\bigsqcap_{f\in F}(A : f) \right) \right]$$

where $al(n)$ denotes the attribute-label of the rooted path in T.

We can show:

Lemma 2 $G \equiv C_G$.

Proof idea: Let n be a node in G. Then, it is easy to see that if C_n is translated back into a description graph G'' where according to part 3 of the definition of C_n the r-edges for $\forall R.C_{G'} \sqcap (\geq l\ R) \sqcap (\leq k\ R) \sqcap \bigsqcap_{f\in F}(R : f)$ are merged to one r-edge, then G'' is isomorphic to the subgraph of G only consisting of the node n.

The conjunction consisting of the conjuncts $\forall al(n).C_n$ where n is a node in T can be translated back into a description graph where certain a-edges are merged such that the resulting graph is a tree isomorphic to T except for the fillers of the a-edges. Now adding the remaing same-as equalities yields a description graph G' which is isomorphic to G except for fillers on a-edges.

Finally, it is not hard to verify that the description graph for the conjunction consisting of $\forall al(n).\left[\bigsqcap_{(n,m,A,F)\in E}\left(\bigsqcap_{f\in F}(A:f)\right)\right]$ on the one hand, where, again, n is a node in T, and the graph G' on the other hand can be merged in such a way that the resulting description graph (which by construction is equivalent to C_G) is isomorphic to G. ■

3.4 Canonical Description Graphs

In the following we occasionally refer to "*marking a node incoherent*"; this means that the label of this node is changed to $\bot$. "*Marking a description graph as incoherent*" means that the description graph is replaced by the graph $G(\bot)$ corresponding to $\bot$.

One important property of canonical description graphs is that they are deterministic, i.e., every node has at most one outgoing edge (r-edge or a-edge) with the same attribute or role name. Therefore, as in [10], to turn a description graph into a canonical graph we need to merge a-edges and r-edges. In addition, because in the graphs defined here, attributes can occur in r-edges and in a-edges it might be necessary to "lift" r-edges to a-edges. Finally, because of the one of constructor on host values, it might be necessary to merge nodes.

To *merge two a-edges* (n, n_1, A, F_1) and (n, n_2, A, F_2) in a description graph G, replace them with a single new edge $(n, n', A, F_1 \cup F_2)$ where n' is the result of merging n_1 and n_2. (If $n_1 = n_2$ then $n' = n_1$.) In addition, replace n_1 and n_2 by n' in all other a-edges of G.

To *merge two r-edges* (R, l_1, r_1, F_1, G_1), (R, l_2, r_2, F_2, G_2) replace them by $(R, max(l_1, l_2), min(r_1, r_2), F_1 \cup F_2, G_1 \oplus G_2)$.

To *lift up* an r-edge (A, l, r, F_A, G_A) of a node n in concept graph G when G has edge (n, n_1, A, F), remove it from $n.REdges$, and augment G by adding $G_A.Nodes$ to $G.Nodes$, $G_A.Edges$ to $G.Edges$, as well as adding $(n, G_A.Root, A, F_A)$ to $G.Edges$.

To *merge two nodes* n_1, n_2 *in a concept graph* G let n' be the result of merging n_1, n_2 and replace n_1 and n_2 by n' in all a-edges of G.

Description graphs are transformed into *canonical form* by repeating the following *normalization rules* whenever possible for the description graph and all its descendants. In the sequel, the *cardinality of the dom* of a node is defined as follows: $\top$ has cardinality ∞ and if the domain is a subset of $\mathcal{IND}$ the cardinality is the number of elements in this subset.

1. If any node in a description graph is marked incoherent, mark the description graph as incoherent. *(Reason: Even if the node is not a root, attributes corresponding to a-edges must always have a value (since they participate in same-as equalities), and this value cannot belong to the empty set.)*

2. If an a-edge of a node has more than one filler, then mark the node incoherent. *(Reason: Attributes can only have at most one filler.)*

3. If an a-edge of a node n points to a node n' where the *dom* is not $\top$ and the filler of the a-edge is not included the *dom* of n', mark n incoherent. *(Reason: Same as 2.)*

4. If the *dom* of a node is empty, mark the node incoherent. *(Reason:* $\{\} \equiv \bot$*)*

5. If some r-edge in a node has its min greater than its max, mark the node incoherent. *(Reason:* $(\geq 2\, R) \sqcap (\leq 1\, R) \equiv \bot$*)*

6. If for an r-edge *dom* of the distinguished node of the restriction graph[3] is not $\top$ and the fillers of this r-edge are not a subset of the *dom*, mark the node of the r-edge incoherent. *(Reason:* $R : I \sqcap \forall R.\{I_1, \ldots, I_n\} \equiv \bot$ *if* $I \notin \{I_1, \ldots, I_n\}$*)*

7. If some node has both $\top_H$ and $\top_C$ in its atoms, mark the node incoherent. If some node has in its atoms a pair of host concepts that are not related by the pre-defined subsumption relationship, mark the node incoherent. *(Reason: The intersection of the atoms will be empty.)*

8. If the fillers of an r-edge are host individuals and they are not contained in the extension of all atoms of the root of the restriction graph for the r-edge then mark the node as incoherent. *(Reason:* $R : I \sqcap \forall R.C_H \equiv \bot$ *if* I *is not an element of the extension of the host concept* C_H*.)*

9. If the fillers of an a-edge from n to n' are host individuals and they are not contained in the extension of all atoms of the node the a-edge is pointing to then mark n as incoherent. *(Reason: see 8.)*

10. If some node has in its atoms a pre-defined host concept, add $\top_H$ to its atoms. If some node has an atomic concept name in its atoms, add $\top_C$ to its atoms. For each pre-defined host concept in the atoms of the node, add all the more-general pre-defined host concepts to its atoms.

[3]Recall that this is the graph contained in an r-edge.

11. If a host individual in the *dom* of a node is not in all the atoms of the node, remove it from the *dom*. *(Reason:* $\{I_1, I_2\} \sqcap INTEGER \equiv \{I_1\} \sqcap INTEGER$ *if* I_1 *is an integer but* I_2 *is not an integer.)*

12. If *dom* of a node is not $\top$ and *dom* contains host individuals then add to the atoms of the node all host concepts which extensions contain all the host individuals in *dom*. *(Reason:* $\{I_1, I_2\} \sqcap INTEGER \equiv \{I_1, I_2\} \sqcap INTEGER \sqcap \mathsf{REAL}$ *if* I_1 *and* I_2 *both belong to* $INTEGER$ *since* $INTEGER \sqsubseteq REAL$*.)*

13. If some r-edge in a node has its description graph marked incoherent, change its max to 0. *(Reason:* $(\leq 0\, R) \equiv \forall R.\bot$*.)*

14. If some r-edge in a node has a max of 0, mark its restriction graph as incoherent. *(Reason: See 13)*

15. If the min on an r-edge is less than the cardinality of fillers on it, let the min be this cardinality. *(Reason:* $R : I_1 \sqcap R : I_2 \sqcap R : I_3 \sqsubseteq (\geq 3\, R)$*)*

16. If the max on an r-edge is greater than the cardinality of the *dom* on the distinguished node of the restriction graph, make the max of this edge be the cardinality of the *dom*. *(Reason:* $\forall R.\{I_1, I_2, I_3\} \sqsubseteq (\leq 3\, R)$*)*

17. If the min on an r-edge is greater than or equal to the cardinality of the *dom* on the distinguished node of the restriction graph, let the fillers of the edge be the union of its fillers and the *dom* above. (If min is greater than the cardinality, then steps 5 and 16 detect the inconsistency.) *(Reason:* $\forall R.\{I_1, I_2, I_3\} \sqcap (\geq 3\, R) \sqsubseteq R : I_1 \sqcap R : I_2 \sqcap R : I_3$; $\forall R.\{I_1, I_2, I_3\} \sqcap (\geq 4\, R) \equiv \bot$*)*

18. If the max on an r-edge is less than or equal to the cardinality of fillers on the edge, let the *dom* on the distinguished node of the restriction graph be the intersection of the *dom* and the fillers. (If max is less than the cardinality, steps 5 and 15 detect the inconsistency.) *(Reason:* $R : I_1 \sqcap R : I_2 \sqcap R : I_3 \sqcap (\leq 3\, R) \sqsubseteq \forall R.\{I_1, I_2, I_3\}$; $R : I_1 \sqcap R : I_2 \sqcap R : I_3 \sqcap (\leq 2\, R) \equiv \bot$*)*

19. If an a-edge has a filler and the node at its end has $\top$ as its *dom* or the filler is an element of the *dom*, make the *dom* be the filler. *(Reason:* $A : I \equiv A : I \sqcap \{I\}$*)*

20. If a node has only one element in its *dom*, add this element to the fillers for all the a-edges pointing to it. *(Reason:* $\{I\} \equiv A : I \sqcap \{I\}$*)*

21. If there are two nodes n_1, n_2 where the *dom* in each of these nodes is $\{I\}$ for some host individual I, merge these nodes in G.[4] *(Reason: $A \downarrow B \sqcap C \downarrow D \sqcap \forall A.\{I\} \sqcap \forall C.\{I\} \sqsubseteq A \downarrow C$ if I is a host individual)*

22. If some node has two r-edges labeled with the same role, merge the two edges, as described above. *(Reason: $\forall R.C \sqcap \forall R.D \equiv \forall R.(C \sqcap D)$)*

23. If some description graph has two a-edges from the same node labeled with the same attribute, merge the two edges, as described above. *(Reason: $\forall A.C \sqcap \forall A.D \equiv \forall A.(C \sqcap D)$)*

24. If some node in a graph has both an a-edge and an r-edge for the same attribute, then "lift up the r-edge", as described above. *(Reason: r-edges participating in same as restrictions need to be a-edges.)*

These normalization rules are well-defined: Starting with a description graph the outcome of applying a normalization rule is always a description graph. In particular, *min* of an r-edge with attribute is always 0 and $max \in \{0, 1\}$. Furthermore, no rules can force such an r-edge to have fillers.

We need to show that the transformations to canonical form do not change the extension of the graph. The main difficulty is in showing that the merging processes and the lifting do not change the extension.

Lemma 3 *Let $G = (N, E, r, l)$ be a description graph with two mergeable a-edges and let $G' = (N', E', r', l')$ be the result of merging these two a-edges. Then, $G \equiv G'$.*

Proof: Let the two edges be (n, n_1, A, F_1) and (n, n_2, A, F_2) and the new node n' be $n_1 \oplus n_2$.

Choose $d \in G^{\mathcal{I}}$, and let Υ be a function from N into the domain of $\mathcal{I}$ satisfying the conditions for extensions (Definition 5) such that $\Upsilon(r) = d$. Then $\Upsilon(n_1) = \Upsilon(n_2)$ because both are equal to $A^{\mathcal{I}}(\Upsilon(n))$. Let Υ' be the same as Υ except that $\Upsilon'(n') = \Upsilon(n_1) = \Upsilon(n_2)$. Then Υ' satisfies Definition 5, part 3, for G', because we replace n_1 and n_2 by n' everywhere, and the conditions for fillers are satisfied for Υ. Moreover, $\Upsilon'(n') = \Upsilon(n_1) \in n_1^{\mathcal{I}} \cap n_2^{\mathcal{I}}$, which by Lemma 1 equals $(n_1 \oplus n_2)^{\mathcal{I}}$; so part 2 is satisfied too, since $n' = n_1 \oplus n_2$. Finally, if the root is modified by the merger, i.e., n_1 or n_2 is r, say n_1, then $d = \Upsilon(n_1) = \Upsilon'(n')$, so part 1 of the definition is also satisfied.

Conversely, given $d \in G'^{\mathcal{I}}$, let Υ' be the function stipulated by Definition 5 such that $\Upsilon'(r') = d$. Let Υ be the same as Υ' except that

[4]This rule was missing in [10].

$\Upsilon(n_1) = \Upsilon(n')$ and $\Upsilon(n_2) = \Upsilon'(n')$. Then the above argument can be traversed in reverse to verify that Υ satisfies Definition 5, such that $d \in G^{\mathcal{I}}$. ■

Lemma 4 *Let n be a node with two mergeable r-edges and let n' be the node with these edges merged. Then $n^{\mathcal{I}} = n'^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.*
Proof: Let the two r-edges be $(R, m_1, M_1, F_1, G_1,)$ and (R, m_2, M_2, F_2, G_2).

Let $d \in n^{\mathcal{I}}$. Then there are between m_1 (m_2) and M_1 (M_2) elements d' of the domain such that $(d, d') \in R^{\mathcal{I}}$. Therefore there are between the maximum of m_1 and m_2 and the minimum of M_1 and M_2 elements d' of the domain such that $(d, d') \in R^{\mathcal{I}}$. Furthermore, for all $f \in F_1$ ($f \in F_2$) there is a d' such that $(d, d') \in R^{\mathcal{I}}$ and $d' \in f^{\mathcal{I}}$. Thus, there are fillers of d for all $f \in F_1 \cup F_2$. Also, all d' such that $(d, d') \in R^{\mathcal{I}}$ are in $G_1^{\mathcal{I}}$ ($G_2^{\mathcal{I}}$). Therefore, all d' such that $(d, d') \in R^{\mathcal{I}}$ are in $G_1^{\mathcal{I}} \cap G_2^{\mathcal{I}}$, which equals $(G_1 \oplus G_2)^{\mathcal{I}}$ by Lemma 1. Thus, $d \in n'^{\mathcal{I}}$.

Let $d \in n'^{\mathcal{I}}$. Then there are between the maximum of m_1 and m_2 and the minimum of M_1 and M_2 elements d' of the domain such that $(d, d') \in R^{\mathcal{I}}$. Therefore there are between m_1 (m_2) and M_1 (M_2) elements d' of the domain such that $(d, d') \in R^{\mathcal{I}}$. Furthermore, for all $f \in F_1 \cup F_2$ there is a d' such that $(d, d') \in R^{\mathcal{I}}$ and $d' \in f^{\mathcal{I}}$. Thus, for all $f \in F_1$ ($f \in F_2$) there is a d' such that $(d, d') \in R^{\mathcal{I}}$ and $d' \in f^{\mathcal{I}}$. Also, all d' such that $(d, d') \in R^{\mathcal{I}}$ are in $(G_1 \oplus G_2)^{\mathcal{I}} = G_1^{\mathcal{I}} \cap G_2^{\mathcal{I}}$. Therefore, all d' such that $(d, d') \in R^{\mathcal{I}}$ are in $G_1^{\mathcal{I}}$ ($G_2^{\mathcal{I}}$). Hence, $d \in n^{\mathcal{I}}$. ■

Lemma 5 *Let $G = (N, E, r, l)$ be a description graph with node n and a-edge (n, n', A, F). Suppose n has an associated r-edge (A, l, r, F_A, G_A) and let $G' = (N', E', r', l')$ be the result of lifting up the r-edge. Then, $G \equiv G'$.*
Proof: Obviously, it is sufficient to show that $G^{\mathcal{I}}_{|n} = G'^{\mathcal{I}}_{|n}$ since only the label of n is changed in G' and only n obtains an additional a-edge which point to the graph G_A not connected to the rest of G'. W.l.o.g. we therefore can assume that n is the root of G, i.e., $n = r$. Let $d \in G^{\mathcal{I}}$. Thus, there is a function Υ from N into Δ as specified in Definition 5 and an individual e such that $d = \Upsilon(n)$, $e = \Upsilon(n')$, and $(d, e) \in A^{\mathcal{I}}$. This implies $e \in G_A^{\mathcal{I}}$. Hence, there exists a function Υ' from $G_A.Nodes$ into Δ for G_A and e satisfying the conditions in Definition 5. Since the sets of nodes of G and G_A are disjoint, we can define Υ'' to be the union of Υ and Υ', i.e., $\Upsilon''(m) := \Upsilon(m)$ for all nodes m in G and $\Upsilon''(m) := \Upsilon'(m)$ for all nodes m in G_A. Since by construction for the additional a-edge $(n, G_A.Root, A, F_A) \in E'$ we have $(\Upsilon''(n), \Upsilon''(G_A.Root)) \in A^{\mathcal{I}}$ and the conditions on the fillers F_A are satisfied, it follows that all conditions in Definition 5 are satisfied for d and G', and thus, $d \in G'^{\mathcal{I}}$.

Now let $d \in G'^{\mathcal{I}}$. Thus, there is a function Υ'' from N' into Δ according to Definition 5. Let $e := \Upsilon''(G_A.Root) = \Upsilon''(n')$. Let G'' be the description graph we obtain from G' by deleting the nodes corresponding to G_A, which is the same graph as G without the r-edge (A, l, r, F_A, G_A). If we restrict Υ'' to the nodes of G'', then it follows $d \in G''^{\mathcal{I}}$. Furthermore, restricting Υ'' to the nodes of G_A yields $e \in G_A^{\mathcal{I}}$. Since e is the only A-successor of d, we can conclude $d \in G^{\mathcal{I}}$. ∎

Lemma 6 *Let $G = (N, E, r, l)$ be a description graph with nodes n_1 and n_2 such that the* dom *of these nodes is $\{I\}$ where I is a host individual. Let $G' = (N', E', r', l')$ be the result of merging n_1 and n_2 in G. Then, $G \equiv G'$.*
Proof: Let n' be the result of merging n_1 and n_2, i.e., $n' = n_1 \oplus n_2$.

Choose $d \in G^{\mathcal{I}}$, and let Υ be a function from N into the domain satisfying the conditions for extensions (Definition 5) such that $\Upsilon(r) = d$. Then, $\Upsilon(n_1) = \Upsilon(n_2) = I$ because the *dom* of n_1 and n_2 is $\{I\}$. Let Υ' be the same as Υ except that $\Upsilon'(n') = \Upsilon(n_1) = \Upsilon(n_2)$. Then Υ' satisfies Definition 5, part 3, for G', because we replace n_1 and n_2 by n' everywhere, and the conditions for fillers are satisfied for Υ. Moreover, $\Upsilon'(n') = \Upsilon(n_1) \in n_1^{\mathcal{I}} \cap n_2^{\mathcal{I}}$, which, by Lemma 1, equals $(n_1 \oplus n_2)^{\mathcal{I}}$; so part 2 is satisfied too, since $n' = n_1 \oplus n_2$. Finally, if the root is modified by the merger, i.e., n_1 or n_2 is r, say n_1, then $d = \Upsilon(n_1) = \Upsilon'(n')$, so part 1 of the definition is also satisfied.

Conversely, given arbitrary $d \in G'^{\mathcal{I}}$, let Υ' be the function stipulated by Definition 5 such that $\Upsilon'(r') = d$. Let Υ be the same as Υ' except that $\Upsilon(n_1) = \Upsilon(n')$ and $\Upsilon(n_2) = \Upsilon'(n')$. Then the above argument can be traversed in reverse to verify that Υ satisfies Definition 5, so that $d \in G^{\mathcal{I}}$. ∎

Having dealt with the issue of merging and lifting, it is now easy to verify that "normalization" does not affect the meaning of description graphs.

Theorem 2 *If G is a description graph and G' is the corresponding canonical description graph, then G and G' are equivalent, i.e., $G \equiv G'$.*

An example of the canonical description graph shown in Figure 1 is given in Figure 2. In this example all r-edges are lifted since there are corresponding a-edges. Thus, all nodes contain only atoms and no r-edges anymore.

3.5 Subsumption Algorithm

The final part of the subsumption process is checking to see if a canonical description graph is subsumed by a concept description. It turns out that it

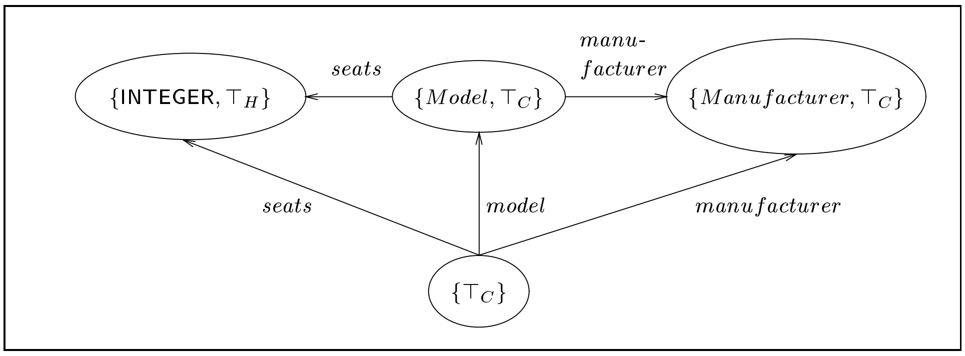

Figure 2: The canonical description graph for *Car* where the components for individuals as well as $\top$ are omitted

is not necessary to turn the potential subsumer into a canonical description graph.

Algorithm 1 (Subsumption Algorithm) *Given a description D and description graph $G = (N, E, r, l)$,* subsumes?(D, G) *is defined to be true if and only if any of the following conditions hold:*

1. *The description graph G is marked incoherent.*
2. *D is $\top$.*
3. *D is a concept name, $\top_C$, or $\top_H$ and is an element of the atoms of r.*
4. *D is $(\geq n\ R)$ and i) some r-edge of r has R as its role and min greater than or equal to n; or ii) $n = 0$ and r has $\top_C$ in its atoms.*
5. *D is $(\leq n\ R)$ and some r-edge of r has R as its role and max less than or equal to n.*
6. *D is $R : I$, for a role R, and some r-edge of r has role R and fillers including I.*
7. *D is $A : I$, for an attribute A, and some a-edge from r has attribute A and fillers including I.*
8. *D is $\{I_1 \ldots I_n\}$ and the* dom *of r is a subset of $\{I_1 \ldots I_n\}$.*
9. *D is $A_1 \cdots A_n \downarrow B_1 \cdots B_m$ and r has $\top_C$ in its atoms, and there are paths with attribute-label $A_1 \cdots A_n$ and $B_1 \cdots B_m$ in G starting from r and ending at the same node.*

10. *D is $\forall R.C$, for a role R, and i) some r-edge of r has R as its role and G' as its restriction graph and, further,* subsumes?(C, G')*; or ii) subsumes?$(C, G(\top))$ and r has $\top_C$ in its atoms.* (Reason: $\forall R.\top$ only requires the possibility that R be applicable to an object, which is absent for host values.)

11. *D is $\forall A.C$, for an attribute A, and i) some a-edge of G is of the form (r, r', A, F), and* subsumes?$(C, (N, E, r'))$*; or ii) some r-edge of r has A as its attribute and G' as its restriction graph and, further,* subsumes?(C, G')*; or iii) subsumes?$(C, G(\top))$ and r has $\top_C$ in its atoms.*

12. *D is $C \sqcap E$ and both subsumes?(C, G) and subsumes?(E, G) are true.*

In the next section we will proof soundness and completeness of the subsumption algorithm.

In [10], it has been shown that the canonical description graph G of a concept description C can be constructed in time polynomial in the size of C. Furthermore, Algorithm 1 runs in time polynomial in the size of G and D. It is not hard to see that the changes presented here do not increase the complexity. Thus, soundness and completeness of the subsumption algorithm provides us with the following corollary.

Corollary 1 *Subsumption for* CLASSIC *concept descriptions C, D, where attributes are interpreted as partial functions, can be decided in time polynomial in the size of C and D.*

3.6 Soundness and Completeness of the Subsumption Algorithm

The soundness of this algorithm is fairly obvious, so we shall not dwell on it. The structure of the proof of completeness is similar to that in [10]. We show that if Algorithm 1 returns false then there is an interpretation $\mathcal{I}$ and an element of the domain of $\mathcal{I}$ such that this element is in the extension of D but not in the one for G. To be more precise, we define a set of interpretations, so-called graphical worlds, for the canonical description graph G (not marked incoherent) such that the root of the graphical world (the distinguished element) is in the extension of G. In order to show completeness, we then prove that one can pick a graphical world for G such that the distinguished element of this world does not belong to the extension of D.

A common operation is to merge two interpretations.

Definition 8 *Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two interpretations. The* merge *of $\mathcal{I}_1$ and $\mathcal{I}_2$, $\mathcal{I}_1 \oplus \mathcal{I}_2$, is an interpretation with classic realm the disjoint union of the classic realm of $\mathcal{I}_1$ and the classic realm of $\mathcal{I}_2$. The extension of atomic names, roles, attributes, and classic individuals in $\mathcal{I}_1 \oplus \mathcal{I}_2$ is the disjoint union of their extensions in $\mathcal{I}_1$ and $\mathcal{I}_2$. Note that the host realm and the interpretation of host concept names and host individuals is the same for all interpretations.*

It is easy to show that the extension of a concept description, a description graph, or a node in $\mathcal{I}_1 \oplus \mathcal{I}_2$ is the union (disjoint union for the classic realm) of its extensions in $\mathcal{I}_1$ and $\mathcal{I}_2$.

Another operation is to add new domain elements to an interpretation. These new domain elements must be in the classic realm. The extension of all atomic concept names, roles, attributes, and classic individuals remain the same except that the new domain elements belong to some arbitrary set of atomic concept names (classic individuals) and have some arbitrary set of successors for each role (attribute). Again, for CLASSIC it is easy to show that a domain element of the original world is in an extension in the original world iff it is in the extension in the augmented world.

A final operation is to add to some individual role successors for a role or attribute which is not restricted in any way by the concepts to which the individual belongs. We now define the set of graphical worlds for canonical description graphs which are not marked incoherent as well as nodes of such graphs.

Given a node, n, that is not marked as incoherent, we construct the *graphical worlds for n* as follows:

1. If the atoms of n are precisely $\top$, then n can have no r-edges, because the only constructs that cause r-edges to be created also add $\top_C$ to the atoms. Furthermore, *dom* must be $\top$ because otherwise the atoms of n would contain $\top_C$ or $\top_H$. Any interpretation, with any domain element the distinguished domain element, is a graphical world for n.

2. If the atoms of n include $\top_H$, then n can have no r-edges. If *dom* is $\top$ then any interpretation, with distinguished element any domain element exactly in the extension of all the atoms of n, is a graphical world for n. This is is possible because of the normalization rule 10 for canonical description graphs. If *dom* is not $\top$ then *dom* can only contain host individuals that are elements of any host concept in the set of atoms of n. In this case, all interpretations with distinguished element one of the elements in *dom* are graphical worlds for n.

3. If the atoms of n include $\top_C$, then for each r-edge, (R, m, M, F, G), in n, construct between m and M pairwise disjoint graphical worlds for G. This can be done for any number between m and M because $m \leq M$ and if $m > 0$ then G is not incoherent, and if G is marked incoherent then $M = 0$. Let S denote the set of distinguished elements of the constructed worlds. For each filler f in F we require that there is an element in S that belongs to the extension of f. In the following, we verify that such a set S can be constructed:

   (a) If G is incoherent, then F is the empty set and $S := \emptyset$ satisfies the conditions.

   (b) Assume the root of G has $\top_H$ in its atoms. By assumption, the intersection of the extensions of the atoms is infinite.

   i) Thus, if *dom* of the root of G is $\top$, then pairwise distinct element can be chosen. Furthermore, since n is coherent F is a subset of each extension of an atom of the root of G and the cardinality of F is less or equal m. Thus, we can construct a set S satisfying the required conditions.

   ii) If the *dom* of G is not $\top$, then *dom* consists of host individuals. Since n is not marked incoherent the cardinality of the *dom* is at least M. Therefore, again, the distinguished elements of the graphical worlds can be chosen pairwise distinct. Furthermore, since F is a subset of *dom* of the root of G and m is greater or equal to the cardinality of F, the set S of distinguished elements can be chosen such that it contains F.

   (c) Assume, G has $\top_C$ in its root. Then, the distinguished elements for the graphical worlds of G can be defined to be pairwise distinct. If F is not empty, then for each filler f in F one can choose one distinguished element to be in the extension of f. Since, if *dom* is not $\top$, the fillers F are a subset of *dom* and we define the distinguished elements to be contained in one of the extensions of the classic individuals in the fillers. All other distinguished elements can be defined to be in the extensions of one classic individual in *dom* if *dom* is not $\top$ and otherwise, if *dom* is $\top$, there are defined to be not contained in any extension of an individual.

   Now merge all the graphical worlds for each r-edge into one interpretation. Add some new domain elements such that one of them satisfies the following conditions:

(a) It is in exactly the extensions of the atoms of n;

(b) if *dom* is not $\top$, then it is in one of the extensions of the classic individuals in *dom*; and

(c) it has as successors for each R exactly the distinguished elements S of the appropriate graphical worlds (see above).

It is easy to see that the resulting world is a graphical world for n. Although, we have shown that R-successors can be constructed for any number between m and M, in graphical worlds the number of such successors of distinguished elements of nodes should only be m. However, in some cases in the proof of completeness we will need to extend the number of R-successors for particular elements.

Given a canonical description graph, $G = (N, E, r, l)$, that is not marked incoherent, we construct the *graphical worlds for G* as follows:

1. For each node $n \in N$ construct a graphical world for n. This can be done because none of them are marked incoherent. Note that in an canonical description graph all a-edges pointing to a single node have the same value for their fillers, and that if this is not the empty set, then the node has this as the value for its *dom*. Thus, the distinguished elements of the graphical worlds corresponding to a node not only satisfy the restrictions on the *dom* on the node but also the restrictions on the fillers in a-edges pointing to this node.

2. Merge these graphical worlds.

3. Modify the resulting world $\mathcal{I}$ so that for each $(n_1, n_2, A, F) \in E$ the A-successor for the distinguished node of the graphical world from n_1 is the distinguished node of the graphical world from n_2. Make the distinguished element of r be the distinguished element of $\mathcal{I}$. As mentioned above, the distinguished element in n_2 is an element of the extension of F (since n_1 is consistent F contains at most one individual). Furthermore, note that the distinguished element in the graphical world of n_1 has no A-successor since n_1 has no r-edge with label A (normalization rule 24).

It is easy to show that the distinguished node of $\mathcal{I}$ is in the extension of G.

Now we can show the final part of the result.

Theorem 3 *Let D be a* Classic *description and G a canonical description graph. Then, if subsumes?(D, G) yields false then G is not subsumed by D.*
Proof: Assume subsumes?(D, G) indicates that G is not subsumed by D. We show that there are some graphical worlds for G such that their distinguished domain elements are not in the extension of D.

Remember that if the subsumption algorithm indicates that G is not subsumed by D, G must not be marked as incoherent and thus there are graphical worlds for G.

The proof proceeds by structural induction on D. Let $G := (N, E, r, l)$.

- If D is $\top_C$ or $\top_H$, then the distinguished domain elements will be in the wrong realm. If D is $\top$, then it is not possible for the subsumption algorithm to indicate a non-subsumption. In each case any graphical world for G has the property that its distinguished domain element is not in the extension of D.

- If D is an atomic concept name then D does not occur in the atoms of r. By construction, in any graphical world for G the distinguished domain element will not be in the extension of D.

- If D is a pre-defined host concept name then again D does not occur in the atoms of r. We distinguish two cases:

  1. If *dom* of r is $\top$ then the distinguished elements of all graphical worlds of G are exactly in the atoms of r by construction.
  2. If *dom* of r is not $\top$ then *dom* contains only host individuals. Thus, all distinguished elements of graphical worlds for G are some of the host individuals in *dom*. Since D is not in the atoms of r the normalization rule 12 ensures that not all of these host individuals are in the extension of the atoms of r. Thus, there is at least one graphical world for G such that its distinguished element is not in the extension of D.

- If D is of the form $(\geq n\, R)$ then $n > 0$ and either the r-edge from r labeled with R has min less than n or there is no such r-edge.

  In the former case all distinguished elements of graphical worlds for G have less then n R-successors. Thus, the distinguished elements of these worlds are not in the extension of D.

  In the latter case, again, all distinguished elements of graphical worlds for G have less then n R-successors, since the number of R-successors is $0 < n$. Thus, the distinguished elements of these worlds are not in the extension of D.

- If D is of the form $(\leq n\, R)$ then either the r-edge from r labeled with R has max greater than n (including ∞) or there is no such r-edge.

  In the former case one can extend graphical worlds for G such that the distinguished nodes have at least $n + 1$ successors for R, because n is less than the max on the r-edge for R, and thus the distinguished node is not in the extension of D.

  In the latter case, one can extend graphical worlds for G such that the distinguished nodes have any number of successors for R. Those with $n+1$ successors have the property that their distinguished node is not in the extension of D.

- If D is of the form $R : I$ then either the r-edge from r labeled with R does not have filler I or there is no such r-edge.

  In the former case the cardinality of the *dom* of the distinguished node of the description graph of this r-edge is greater than the min, m, of the r-edge, or the *dom* does not include I.

  If the *dom* does not include I, then all graphical worlds for the node have their distinguished element not in the extension of I, as required. If the *dom* does include I, there are at least m elements of the *dom* besides I, and the successors of the r-edge are a subset of the set of these elements. There are thus graphical worlds for G that use only these elements, as required.

  In the latter case, all distinguished elements of graphical worlds for G have no R-successors. Obviously, these distinguished elements are not in the extension of D.

- If D is of the form $A : I$ then we distinguish two cases:

  1. The root of G has no a-edge with label A. If G has an r-edge with label A then we know that the *min* component of this r-edge is 0. Thus, in any case, all distinguished elements of graphical worlds for G have no A-successors. Obviously, the distinguished elements for these graphical worlds for G are not in the extension of D.
  2. If the root r has an a-edge labeled with A then the node pointed to by the a-edge cannot have as its *dom* the singleton consisting of I. Therefore there are graphical worlds for G that have

their distinguished node A-successor not in the extension of I, as required.

3. If the root has an r-edge labeled with A then the graphical worlds for G can be defined as in the case of $R : I$.

- If D is of the form $A_1 \cdots A_n \downarrow B_1 \dots B_m$ again several cases arise.

  1. If the paths $A_1, \dots, A_n$ and $B_1, \dots, B_m$ exist in G starting from r but end at different nodes n_1 and n_2, then use graphical worlds in which the domain elements for these two nodes are different. This is obviously possible if the distinguished elements of n_1, n_2 can be chosen in different realms, if the atoms of n_1, n_2 contain $\top_C$, or if *dom* of at least one of these nodes is $\top$. Otherwise, the atoms of r contain $\top_H$ and *dom* is a set of host individuals. Because of normalization rule 21 and 19 we know that *dom* of both nodes are not the same singletons. Consequently, if the *dom* of both nodes is a singleton then the distinguished elements of graphical worlds for these nodes are distinct. Otherwise, at least one *dom* contains more than one host individual, say the *dom* for n_1. This implies that there is no a-edge to n_1 with a filler. Therefore, there are graphical worlds for G such that the distinguished elements corresponding to n_1 and n_2 are distinct.

     Thus, in all cases the distinguished elements of the constructed worlds are not in the extension of D.

  2. If one of the paths $A_1, \dots, A_n$ and $B_1, \dots, B_m$ does not exist in G starting from r then in all graphical worlds the image of one of the chains $A_1 \cdots A_n$ or $B_1 \cdots B_m$ is not defined for the distinguished element. Thus, the distinguished element is not in the extension of D. Note that successors of attributes are not required by r-edges since the *min* component is always 0.

- If D is of the form $D_1 \sqcap D_2$ then the subsumption algorithm must indicate that G is not subsumed by at least one of D_1 or D_2. By the inductive hypothesis, we get some graphical worlds of G where the distinguished domain elements are not in the extension of D_1 or not in the extension of D_2, and thus are not in the extension of D.

- If D is of the form $\forall R.C$, where R is a role, then two cases arise.

  1. If subsumes?$(C, G(\top))$ then $\top_C$ is not in the atoms of r. Thus,

there are some graphical worlds for G whose distinguished element is in the host realm, and thus not in the extension of D.

2. Otherwise, either there is an r-edge from r with role R and description graph H such that subsumes?(C, H) is false or there is no r-edge from r with role R. Note that the extension of C is not the entire domain, and thus must be a subset of either the host realm or the classic realm.

   In the former case H is not marked incoherent (or else the subsumption could not be false) and the max on the r-edge cannot be 0. Thus, there is graphical world for H whose distinguished element is not in the extension of C and one can extend a graphical world for G such that the distinguished element uses this distinguished element for H as distinguished domain element R-successors. In these graphical worlds for G the distinguished element is not in the extension of D.

   In the latter case, extend graphical worlds for G such that they have some distinguished node R-successor in the wrong realm. In these graphical worlds for G the distinguished element is not in the extension of D.

- If D is of the form $\forall A.C$, where A is an attribute, then four cases arise.

  1. If subsumes?$(C, G(\top))$ then $\top_C$ is not in the atoms of r. Then there are some graphical worlds for G whose distinguished element is in the host realm, and thus, not in the extension of D.
  2. There is no a-edge and r-edge from r with attribute A. Extend graphical worlds for G such that they have some distinguished node A-successor in the wrong realm. In these graphical worlds for G the distinguished element is not in the extension of D.
  3. There is an a-edge from r with attribute A to some other node r' such that subsumes?(C, H) is false, where $H = (N, E, r')$. Note that the extension of C is not the entire domain, and thus, must be a subset of either the host realm or the classic realm.

     We know H is not marked incoherent, because G is not marked incoherent. Thus, there are graphical worlds for H whose distinguished element is not in the extension of C. Given any graphical world for H, a graphical world for G can be formed simply by changing the distinguished domain element. If the original graphical world's distinguished element is not in the extension of

C, then the new graphical world's distinguished element will not be in the extension of D, as required.

4. If r has an r-edge with attribute A then graphical worlds can be constructed as in the case of $\forall R.C$.

This complete the proof of Theorem 3. ∎

4 Computing the lcs in Classic

In this section, we will show that the lcs of two CLASSIC concept descriptions can be stated in terms of a product of canonical description graphs.

A similar result has been proved in [18] for a sublanguage of CLASSIC, which only allows for atomic concept names, concept conjunction, value restrictions, and same-as equalities. In particular, this sublanguage does not allow for inconsistent concept descriptions (which, for example, can be expressed by conflicting number-restrictions). Furthermore, the semantics of the description graphs defined in [18] is not well-defined: their graphs do not have a recursive structure such that cycles in a graph lead to cyclic statements concerning the extension of the graph.

In the following, we first define the product of description graphs. Then, we show that for given concept descriptions C and D the lcs is equivalent to a description graph obtained as product of G_C and G_D.

4.1 The Product of Description Graphs

A description graph represents the constraints that must be satisfied by all individuals in the extension of the graph. Intuitively, the product of two description graphs is the intersection of these constraints—as the product of finite automata corresponds to the intersection of the words accepted by the automata. Moreover, the intersection of constraints, intuitively, describes the lcs of concepts.[5]

Definition 9 *Let $G_1 = (N_1, E_1, r_1, l_1)$ and $G_2 = (N_2, E_2, r_2, l_2)$ be two description graphs where the recursive set of nodes of these graphs are disjoint. Then, the* product $G = (N, E, r, l)$ of the two graphs *($G_1 \times G_2$ for short) is recursively defined as follows:*

[5]In [4], the lcs for cyclic $\mathcal{ALN}$-concept description is defined in terms of the intersection of regular languages, which corresponds to the product of finite automata. In [20, 18], the correspondence between the lcs and the product of finite automata has been pointed out as well.

1. $N := N_1 \times N_2$;

2. $r := (r_1, r_2)$;

3. $E :=\{((n_1,n_2),(m_1,m_2),A,min(l_1,l_2),max(r_1,r_2),F_1 \cap F_2) \mid$
$(n_1,m_1,A,l_1,r_1,F_1) \in E_1$ *and* $(n_2,m_2,A,l_2,r_2,F_2) \in E_2\}$;

4. *Let* $n_1 \in N_1$ *and* $n_2 \in N_2$. *If* $l_1(n_1) = \bot$, *then let* $l((n_1,n_2)) := l_2(n_2)$ *and, analogously, if* $l_2(n_2) = \bot$ *then* $l((n_1,n_2)) := l_1(n_1)$. *Otherwise if* $l_1(n_1) = (S_1,D_1,H_1)$ *and* $l_2(n_2) = (S_2,D_2,H_2)$ *then* $l((n_1,n_2)) := (S,D,H)$ *where*

(a) $S := S_1 \cap S_2$;

(b) *If* $D_1 = \top$, *then* $D := D_2$; *if* $D_2 = \top$ *then* $D := D_1$; *otherwise* $D := D_1 \cup D_2$.

(c) $H := \{(R,min(l_1,l_2),max(r_1,r_2),F_1 \cap F_2,G_1' \times G_2') \mid$
$(R,l_1,r_1,F_1,G_1') \in H_1,\ (R,l_2,r_2,F_2,G_2') \in H_2\} \cup$
$\{(A,0,1,F_1 \cap F_2,G_{1|n_1'} \times G_2') \mid (n_1,n_1',A,F_1) \in E_1,\ (A,l_2,r_2,F_2,G_2') \in H_2\} \cup$
$\{(A,0,1,F_1 \cap F_2,G_1' \times G_{2|n_2'}) \mid (A,l_1,r_1,F_1,G_1') \in H_1,\ (n_2,n_2',A,F_2) \in E_2\}$.

Note that since attributes can occur both in r-edges and a-edges we have to consider not only the product of restriction graphs but also the product of a description graph and G_1, G_2 rooted at certain nodes.

4.2 Computing the lcs

In this section, we will show how the lcs can be computed using the product of description graphs. W.l.o.g. we can assume that the product of two description graphs is connected (see Section 3.3). The main theorem is:

Theorem 4 *Let* C_1 *and* C_2 *be two concept descriptions, and let* G_1 *and* G_2 *be corresponding* canonical *description graphs. Then,* $C_{G_1 \times G_2} \equiv lcs(C_1,C_2)$.

Proof. First, we show that $C := C_{G_1 \times G_2}$ subsumes C_1 (by symmetry this also holds for C_2). For this purpose, we show by induction over the size of G_1 and G_2 that *subsumes?*(C', G_1) for every conjunct C' in C. Let $G_1 = (N_1,E_1,r_1,l_1)$, $G_2 = (N_2,E_2,r_2,l_2)$, and let $G = (N,E,r,l)$ be $G_1 \times G_2$. According to the definition of C (Section 3.3) we distinguish two cases:

1. Same-as: If this part is equal to $\top$, then there is nothing to show. Let $C' = v \downarrow w$ be a conjunct in C. Then, by definition of C there are two paths in G from the root (r_1, r_2) to a node (n_1, n_2) with attribute-label v and w, respectively ($v = w$ is allowed). By Definition of the product of graphs this implies that there are two paths from r_1 to n_1 in G_1 with attribute-label v and w, respectively. Thus, *subsumes*?(C', G_1).

2. Nodes and a-edges: Let $n = (n_1, n_2)$ be a node in G. Then, since G is connected, it follows from the definition of the product of description graphs that there is a path from r_1 to n_1 with attribute-label $al(n)$ in G_1. Now, let $((n1, n2), (m_1, m_2), A, F) \in E$. Then, by definition of G we know that $(n1, m_1, A, F_1) \in E_1$ where $F \subseteq F_1$. Thus, for $f \in F$ it follows *subsumes*?$(A : f, G_{1|n_1})$. We now show that *subsumes*?$(C_n, G_{1|n_1})$. Then, by the definition of *subsumes*? we can infer

$$subsumes?(\forall al(n). \left[C_n \sqcap \bigsqcap_{(n,m,A,F) \in E} \left(\bigsqcap_{f \in F} (A : f) \right) \right], G_1)$$

which then completes the proof that C subsumes C_1. In order to show *subsumes*?$(C_n, G_{1|n_1})$ we show that *subsumes*?$(D', G_{1|n_1})$ for every conjunct D' in C_n. If $l(n) = \bot$, then this implies $l_1(n_1) = \bot$ by definition of G. Thus, *subsumes*?$(C_n, G_{1|n_1})$. If $l_2(n_2) = \bot$, then by definition $l(n) = l_{N_1}(n_1)$ which also implies *subsumes*?$(C_n, G_{1|n_1})$. Otherwise, according to the definition of ν we distinguish three cases. Let $l_1(n_1) = (S_1, D_1, H_1)$, $l_2(n_2) = (S_2, D_2, H_2)$, and let $l(n) = (S, D, H)$.

   (a) If $S = \emptyset$, then $D' = \top$ which implies *subsumes*?$(D', G_{1|n_1})$. By definition of l we know $S \subseteq S_1$. Thus, for $S \neq \emptyset$ we also have *subsumes*?$(\bigsqcap_{E \in S} E, G_{1|n_1})$.

   (b) If $D = \top$, then it follows *subsumes*?$(D, G_{1|n_1})$ immediately. Otherwise, by definition of $l(n)$ we know $D \supseteq D_1$, which again shows *subsumes*?$(D, G_{1|n_1})$.

   (c) Let $(R, min, max, F, G') \in H$. According to the definition of l we distinguish three cases:
   i) There are r-edges $(R, k_1, s_1, F_1, G'_1) \in H_1$, $(R, k_2, s_2, F_2, G'_2) \in H_2$ such that $min \leq k_1$ and $max \geq s_1$. Furthermore, $F \subseteq F_1$. This implies *subsumes*?$((\geq min\ R) \sqcap (\leq max\ R) \sqcap \bigsqcap_{f \in F} (R : f), G_{1|n_1})$. Finally, it is $G' = G'_1 \times G'_2$. By the induction hypothesis this means *subsumes*?$(C_{G'}, G'_1)$. Thus, by definition of *subsumes*? we know *subsumes*?$(\forall R.C_{G'}, G_{1|n_1})$.

ii) There is an r-edge $(A, k_1, s_1, F_1, G_1') \in H_1$ and an a-edge $(n_2, n_2', A, F_2) \in E_2$ such that $min = 0$ and $max = s_1$. Furthermore, $F \subseteq F_1$. This implies $subsumes?(\bigsqcap_{f \in F}(A : f), G_{1|n_1})$. Finally, it is $G' = G_1' \times G_{2|n_2}$. By the induction hypothesis this means $subsumes?(C_{G'}, G_1')$. Thus, by definition of $subsumes?$ we know $subsumes?(\forall A.C_{G'}, G_{1|n_1})$.

iii) There is an a-edge $(n_1, n_1', A, F_1) \in E_1$ as well as an r-edge $(A, k_2, s_2, F_2, G_2') \in H_2$ such that $min = 0$ and $max = s_2$. Furthermore, $F \subseteq F_1$. This implies $subsumes?(\bigsqcap_{f \in F}(A : f), G_{1|n_1})$. Finally, it is $G' = G_{1|n_1'} \times G_2'$. By the induction hypothesis this means $subsumes?(C_{G'}, G_{1|n_1'})$. Thus, by definition of $subsumes?$ we know $subsumes?(\forall R.C_{G'}, G_{1|n_1})$.

We now show by induction over the size of D, C_1, and C_2 that if D subsumes C_1 and C_2 then D subsumes C: We distinguish different cases according to the definition of "*subsumes?*". Let $G_1 = (N_1, E_1, r_1, l_1)$ denote the canonical description graph of C_1, $G_2 = (N_2, E_2, r_2, l_2)$ the canonical description graph of C_2, and $G = (N, E, r, l) = G_1 \times G_2$. In the following we assume that $C_1 \sqsubseteq D$ and $C_2 \sqsubseteq D$, thus, $subsumes?(D, G_1)$ and $subsumes?(D, G_2)$. We show that $subsumes?(D, G)$:

1. If G is incoherent or $D = \top$, then there is nothing to show.

2. If D is a concept name, $\top_C$, $\top_H$, fills, one-of, or a number-restriction then by definition of the label of r and its a-edges and r-edges it is easy to see that $subsumes?(D, G)$.

3. If D is $v \downarrow w$, then there exist nodes n_1 in G_1 and n_2 in G_2 such that there are two paths from r_1 to n_1 with label v and w, respectively, as well as two paths from r_2 to n_2 with label v and w. Then, by the definition of G it is easy to see that there are two paths from $r = (r_1, r_2)$ to (n_1, n_2) with label v and w, respectively. This shows $subsumes?(D, G)$.

4. If D is $\forall R.C$, R role or attribute, then i) r_1 and r_2 have r-edges with role or attribute R and restriction graphs G_1' and G_2', respectively, such that $subsumes?(C, G_1')$ and $subsumes?(C, G_2')$, or (w.o.l.g.)

   ii) r_1 has an a-edge pointing to n_1 with attribute R $(G_1' := G_{1|n_1})$ and r_2 has an r-edge with restriction graph G_2' such that $subsumes?(C, G_1')$ and $subsumes?(C, G_2')$. In both of these cases, we know by induction *subsumes*?$(C, G'_1 \times G'_2)$. Furthermore, by definiton of G there is an r-edge with role R and restriction graph $G'_1 \times G'_2$ for r. This implies *subsumes*?(D, G).

Now, iii) assume that r_1 and r_2 have a-edges with attribute R leading to nodes n_1 and n_2, respectively, and *subsumes*?$(C, G_{1|n_1})$ and *subsumes*?$(C, G_{2|n_2})$. By induction we know *subsumes*?$(C, G_{1|n_1} \times G_{2|n_2})$. It is easy to see that $G_{|(n_1,n_2)} = G_{1|n_1} \times G_{2|n_2}$. Furthermore, by definition there is an a-edge with attribute R from (r_1, r_2) to (n_1, n_2) in G. This shows *subsumes*?(D, G).

iv) Finally, we have to consider the case that (w.o.l.g.) r_1 has no r-edge and no a-edge with role or attribute R. Then, this implies *subsumes*?$(C, G(\top))$ and r_1 has $\top_C$ in its atoms. Furthermore, since *subsumes*?(D, G_2) we know that r_2 has $\top_C$ in its atoms: by definition of description graphs if r_2 has an r-edge, then $\top_C$ is an atom of r_2; by definition of *subsumes*? if r_2 has no r-edge and no a-edge with role or attribute R, then $\top_C$ belongs to the atoms of r_2 as well. This means that G has $\top_C$ in its root (r_1, r_2) and thus, *subsumes*?(D, G).

5. If D is $C \sqcap E$, then by definition of the subsumption algorithm we know *subsumes*?(C, G_1) and *subsumes*?(C, G_2). By induction we have *subsumes*?(C, G), and analogously, *subsumes*?(E, G). Thus, *subsumes*?(D, G). ■

As stated in Section 3.5 the canonical description graph for a CLASSIC concept description can be computed in time polynomial in the size of the concept description. Furthermore, it is not hard to verify that the product of two description graphs can be computed in time polynomial in the size of the graphs. In addition, the concept description corresponding to a description graph can be computed in time polynomial in the size of the graph. Thus, as a consequence of Theorem 4 we obtain

Corollary 2 *The lcs of two* CLASSIC *concept descriptions always exists and can be computed in time polynomial in the size of the concept descriptions.*

We now show that this is not true if one is interested in the lcs of a sequence of concept descriptions since the lcs of such a sequence might be exponential in the size of the sequence. As a result, an algorithm computing the lcs of a sequence of concept descriptions is at least exponential. Such an exponential time algorithm can easily be specified since

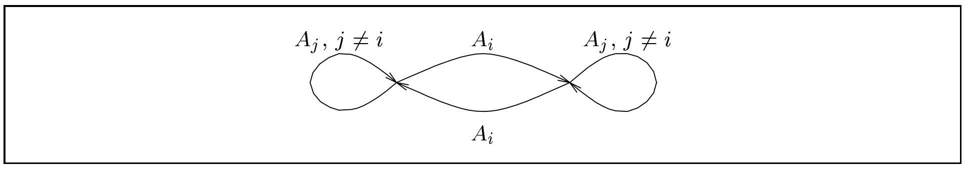

Figure 3: The canonical description graph for D_i without atoms, *dom*, r-edges, and fillers.

$lcs(D_1, \ldots, D_n) = lcs(D_n, lcs(D_{n-1}, lcs(\cdots lcs(D_2, D_1) \cdots)$; one only needs to iterate the algorithm for computing the lcs of two concept descriptions.

We will prove that the size of the lcs of a sequence might be exponential in the size of the descriptions in the sequence. To do so, we take an example proposed in [20], although in that case total attributes are assumed.

Proposition 1 *For all integers $n \geq 2$ there exists a sequence $D_1, \ldots, D_n$ of* Classic *concept descriptions such that the size of every* Classic *concept description equivalent to $lcs(D_1, \ldots, D_n)$ is at least exponential in n where the size of the $D_i{}'s$ is linear in n.*

Proof: For a given n the concept descriptions D_i are defined as follows:

$$D_i \ := \ \bigsqcap_{j \neq i} (\varepsilon \downarrow A_j) \sqcap \bigsqcap_{j \neq i} (A_i \downarrow A_i A_j) \sqcap (\varepsilon \downarrow A_i A_i)$$

where $A_1, \ldots, A_n$ denote attributes. The canonical description graph for D_i is depicted in Figure 3.

Using Algorithm 1 it is easy to see that $D_i \sqsubseteq v \downarrow w$ iff the number of $A_i{}'s$ in v and the number of $A_i{}'s$ in w are equal modulo 2 where v, w are words over $\{A_1, \ldots, A_n\}$. This implies that

$$D_1, \ldots, D_n \sqsubseteq v \downarrow w \quad \text{iff} \quad \begin{array}{l}\text{for all } 1 \leq i \leq n \text{ the number of } A_i{}'s \\ \text{in } v \text{ and the number of } A_i{}'s \text{ in } w \text{ are} \\ \text{equal modulo 2.}\end{array} \qquad (1)$$

Let $s \subseteq \{1, \ldots, n\}$ be a non-empty set. We define $v_s := A_{i_1} \cdots A_{i_k}$ where $i_1 < \cdots < i_k$ are the elements of s and $w_s := A^3_{i_1} A^3_{i_2} \cdots A^3_{i_k}$ with $A_j{}^3 := A_j A_j A_j$. Now let E be the lcs of $D_1, \ldots, D_n$ and G_E the corresponding canonical description graph with root r. From (1) we know that for every $s \subseteq \{1, \ldots, n\}$ it is $E \sqsubseteq v_s \downarrow w_s$. Algorithm 1 implies that the paths from r in G_E labelled v_s and w_s exist and that they lead to the same node q_s.

Assume, there are non-empty subsets s, t of $\{1, \ldots, n\}$, $s \neq t$, such that $q_s = q_t$. This would imply $E \sqsubseteq v_s \downarrow v_t$ in contradiction to (1). Thus, $s \neq t$ implies $q_s \neq q_t$. Since there are $2^n - 1$ non-empty subsets of $\{1, \ldots, n\}$ this shows that G_E contains at least $2^n - 1$ nodes. Finally, as G_E is linear in the size of E this completes the proof of the proposition. ■

5 The Lcs for Same-as and Total Attributes

In the previous sections, attributes were interpreted as partial functions. In this section, we will present the significant changes occurring when considering total functions instead of partial functions. More precisely, we will look at a sublanguage $\mathcal{S}$ of CLASSIC that only allows for concept conjunction and same-as equalities, but where we have the general assumption:

Attributes are interpreted as total functions.

We restrict our attention to the language $\mathcal{S}$ in order to concentrate on the changes caused by going from partial to total functions. However, we strongly conjecture that the results represented here can easily be transfered to CLASSIC by extending the description graphs for $\mathcal{S}$ as in Section 4.

First, we shall show that in $\mathcal{S}$ the lcs_t of two concept descriptions does not always exist. Then, we will present a polynomial decision algorithm for the existence of an lcs_t of two concept descriptions. Finally, it will be shown that if the lcs_t of two concept descriptions exists, then it might be exponential in the size of the given concept descriptions and it can be computed in exponential time. Note that the later result considerably generalizes the original result in [20], which only provides an exponential size for the lcs_t of a sequence of n concept descriptions.

In the sequel, we will simply refer to the lcs_t by lcs. Since throughout the section attributes are always assumed to be total, this does not lead to any confusion.

A useful observation for understanding the proofs in this section is that for total attributes we have $(u \downarrow v) \sqsubseteq_t (uw \downarrow vw)$ for any $u, w, v \in \mathcal{A}^*$, where $\mathcal{A}^*$ is the set of finite words over $\mathcal{A}$, the finite set of attribute names. Throughout this section, attributes are always denoted by small letters a, b, c, d.

5.1 The Existence of the Lcs

In this section, we shall prove that the lcs of two concept descriptions in $\mathcal{S}$ does not always exist. However, there is always an infinite representation of the lcs, which will be used in the next section to characterize the existence of the lcs.

To accomplish the above, we return to the graph-based characterization of subsumption proposed in [10] — the one modified for partial attributes in Section 3. For a concept description C, let G_C denote the corresponding canonical description graph, as defined in Section 3.4. Its semantics is specified as in Section 3.1, although now the set of interpretations is restricted to allow attributes to be interpreted as total functions only.

Since $\mathcal{S}$ contains no atomic concepts and does not allow for value-restrictions, the nodes in G_C do not contain atomic concepts and the set of r-edges is empty. Therefore, G_C can be defined by the triple (N, E, n_0) where N is a finite set of nodes, E is a finite set over $N \times \mathcal{A} \times N$, and n_0 is the root of the graph.

According to [10], subsumption $C \sqsubseteq_t D$ of concept descriptions C and D in $\mathcal{S}$ can be decided with the following algorithm $\mathit{subsumes}_t?(D, G_C)$, which also provides us with a characterization of t-subsumption.[6]

Algorithm 2 *Let C, D be concept descriptions in $\mathcal{S}$, and $G_C = (N, E, n_0)$ be the canonical description graph of C. Then, $\mathit{subsumes}_t?(D, G_C)$ is defined to be true if and only if one of the following conditions hold:*

1. *D is $v \downarrow w$ and there are words v', w', and u over $\mathcal{A}$ such that $v = v'u$ and $w = w'u$, and there are rooted paths in G_C labeled v', resp. w', ending at the same node.*

2. *D is $D_1 \sqcap D_2$ and both $\mathit{subsumes}_t?(D_1, G_C)$ and $\mathit{subsumes}_t?(D_2, G_C)$ are true.*

The main Theorem of this section is

Theorem 5 *There are concept descriptions in $\mathcal{S}$ such that the lcs of these concept descriptions does not exist in $\mathcal{S}$.*

This result corrects the statement in [20] that the lcs always exists, a statement that inadvertently assumed that attributes were partial, not total.

To prove the theorem above, consider the two concept descriptions

$$\begin{aligned} C_0 &:= a \downarrow b \\ D_0 &:= a \downarrow ac \sqcap b \downarrow bc \sqcap ad \downarrow bd \end{aligned}$$

It will be shown that they do not have an lcs in $\mathcal{S}$. The graphs for these concepts are depicted in Figure 4. Intuitively, note that for an lcs E of C_0

[6]Note that $\mathit{subsumes}_t?$ as introduced here is different from the algorithm presented in Section 3.5 since we restricted the set of possible interpretations.

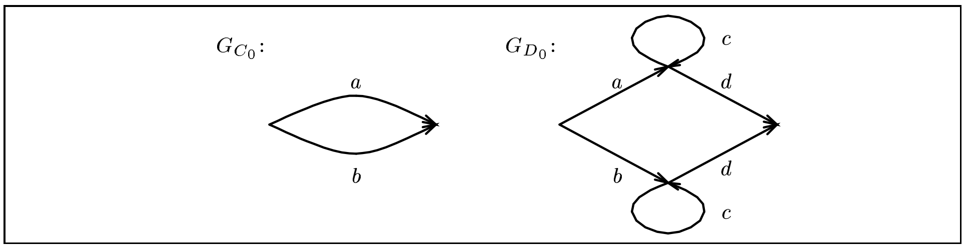

Figure 4: The canonical graphs for C_0 and D_0

and D_0 we have

$$E \sqsubseteq_t v \downarrow w \quad \text{iff} \quad v = w \text{ or there exists a nonnegative integer } n \text{ and } u \in \mathcal{A}^* \text{ such that } v = ac^n du \text{ and } w = bc^n du \text{ or vice versa,} \qquad (2)$$

which is no longer a regular structure. In fact, using Algorithm 2, one can show that no finite description graph can be equivalent to E:

Let G_E be the canonical graph for E with root n_0 and let $v_n := ac^n d$, $w_n := bc^n d$ for all nonnegative integers n. By (2) it follows $subsumes_t?(v_n \downarrow w_n, G_E)$. Thus, for all n there exists a node q_n in G_E as well as v_n', w_n', and u such that $v_n = v_n' u$, $w_n = w_n' u$, and there are two rooted paths to q_n with label v_n' and w_n', respectively. If $u \neq \varepsilon$ then $v_n' = ac^m$ and $w_n' = bc^m$ for some nonnegative integer m. But then we know $subsumes_t?(ac^m \downarrow bc^m, G_E)$, and thus, $E \sqsubseteq ac^m \downarrow bc^m$, which is a contradiction to (2). This implies, $u = \varepsilon$. Since G_E is a finite graph, there exist $n < m$ such that $q_n = q_m$. Thus, by definition of q_n and q_m, there exists a path from n_0 to q_n with label v_n and a path from n_0 to q_n with label w_m. This shows, $subsumes?(v_n \downarrow w_m, G_E)$, and hence, $E \sqsubseteq v_n \downarrow w_m$, which is again a contradiction to (2). Consequently, there is no lcs for the concept descriptions C_0 and D_0.

However one can prove that there always exists an infinite description graph representing the lcs. Such an infinite graph will then be used in Theorem 6 to characterize the existence of an lcs. In particular, from the infinite description graph representing the lcs of C_0 and D_0 in the example, one can deduce that there exists no lcs for these two concept descriptions. So, one could dispense with the proof just presented. Nevertheless, it gives a first idea of how the following more general statements can be shown.

An *infinite description graph* G is defined by a triple (N, E, n_0), as in the case of ordinary, finite description graphs, except that the set of nodes N and the set of edges E might be infinite. The semantics of these graphs is defined as for finite graphs in Section 3.1. Furthermore, infinite graphs are translated into concept descriptions as follows: take an (infinite) spanning tree T of G, and, as in the finite case, for every edge of G not contained in it, add to C_G a same-as equality. (The translation is simpler than the one in Section 3.3 because we are dealing with t-attributes.) Note, however, that such translations might yield concept descriptions with an infinite number of conjunctions (thus, *infinite concept descriptions*). The semantics of such concept descriptions is defined in the obvious way. Analogously to Lemma 2, one can show that an (infinite) graph G and its corresponding concept description C_G are equivalent, i.e., $C_G \equiv_t G$.

We call an (infinite) description graph G *deterministic* if and only if for every node n in G and every attribute $a \in \mathcal{A}$ there is at most one a-successor for n in G. The graph G is called *complete* if for every node n in G and every attribute $a \in \mathcal{A}$ there is (at least) one a-successor for n in G. For a deterministic and complete (infinite) description graph and a word $v \in \mathcal{A}^*$ there is exactly one path labeled v in G from the root n_0 to one node n. Furthermore, if there is a path between two nodes n, n', then there is exactly one path. As before, we write $nvn' \in G$ to say that there is a path in G from n to n' labeled v.

Obviously, Algorithm 2 can be generalized to infinite description graphs G_C. Thus, we can conclude:

Corollary 3 *Let $G = (N, E, n_0)$ be a deterministic and complete (infinite) description graph and $v, w \in \mathcal{A}^*$. Then,*

$$G \sqsubseteq_t v \downarrow w \quad \textit{iff} \quad n_0vn \in G \textit{ and } n_0wn \in G \textit{ for some node } n.$$

We shall construct an (infinite) graph representing the lcs of two concept descriptions in $\mathcal{S}$ as the product of the so-called completed canonical graphs. This infinite representation of the lcs will be used later on to characterize the existence of an lcs in $\mathcal{S}$, i.e., the existence of a finite representation of the lcs.

We now define the completion of a graph. Let G be an (infinite) description graph. The graph G' is an *extension* of G if for every node n in G and for every attribute $a \in \mathcal{A}$ such that n has no outgoing edges labeled a, a new node $m_{n,a}$ is added, as well as an edge $(n, a, m_{n,a})$. Now, let $G^0, G^1, G^2, \ldots$ be a sequence of graphs such that $G^0 = G$ and G^{i+1} is an extension of G^i, for $i \geq 0$. If $G^i = (N_i, E_i, n_0)$, then $G^\infty := (\bigcup_{i\geq 0} N_i, \bigcup_{i \geq 0} E_i, n_0)$ is called the *completion* of G. By construction, G^∞ is a complete graph. Furthermore, if G is deterministic, then G^∞ is deterministic as well. Finally, it is easy to see that a graph and its extension are equivalent. Thus, by induction, $G^\infty \equiv_t G$.

The nodes in $\bigcup_{i\geq 1} N_i$, i.e., the nodes in G^∞ that are not in G, are called *tree nodes*; the nodes of G are called *non-tree nodes*. By construction, for every tree node t in G^∞ there is exactly one direct predecessor of t in G^∞, i.e., there is exactly one node n and one attribute a such that (n, a, t) is an edge in G^∞; n is called *a-predecessor* of t. Furthermore, there is exactly one youngest ancestor n in G of a tree node t in G^∞; n is the *youngest ancestor* of t if there is a path from n to t in G^∞ which does not contain non-tree nodes except for n. Note that there is only one path from n to t in G^∞. Finally, observe that non-tree nodes have only non-tree nodes as ancestors.

As an example, the completion of G_{C_0} depicted in Figure 4 is shown in Figure 5, when $\mathcal{A} = \{a, b, c, d\}$. Note that the completion of a canonical description graph is always complete and deterministic. In the sequel, let C, D

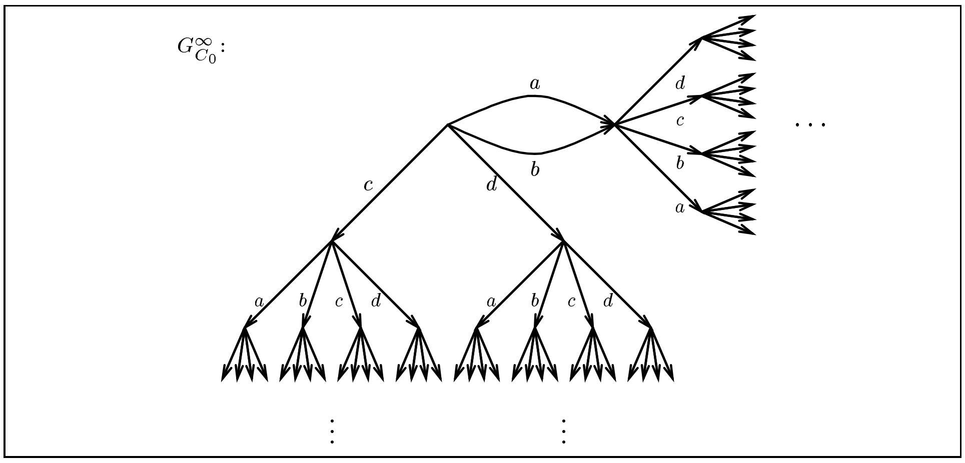

Figure 5: The complete graph for C_0

be two concept descriptions in $\mathcal{S}$, $G_C = (N_C, E_C, n_C)$, $G_D = (N_D, E_D, n_D)$ be their corresponding canonical graphs, and G^∞_C, G^∞_D be the completions of G_C, G_D. The products $G := G_C \times G_D$ and $G^\times_\infty := G^\infty_C \times G^\infty_D$ are defined analogously to Section 4.1, where again G and $G^\times_\infty$ only contain nodes that are reachable from the root (n_C, n_D), i.e., G and $G^\times_\infty$ are supposed to be connected.

A subgraph of $G^\times_\infty$ for the concept descriptions C_0 and D_0 (Figure 4) is depicted in Figure 6.

As an easy consequence of the fact $G_C \equiv_t G^\infty_C$ and Corollary 3 we get

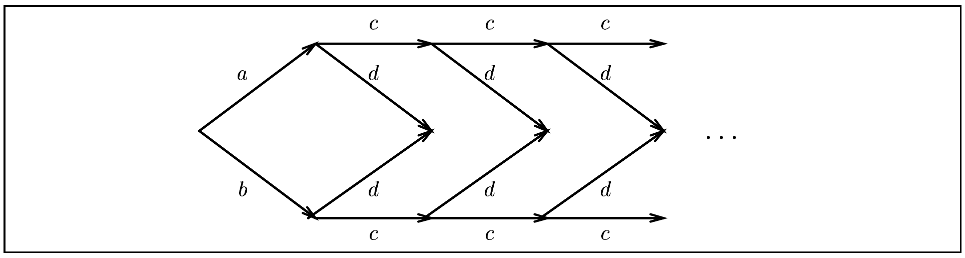

Figure 6: A subgraph of $G^{\infty}_{C_0} \times G^{\infty}_{D_0}$

Lemma 7 *$C \sqsubseteq_t v \downarrow w$ iff $n_C vn \in G^{\infty}_C$ and $n_C wn \in G^{\infty}_C$ for a node n in G^{∞}_C.*

But then, by the construction of $G^{\times}_{\infty}$ we know:

Proposition 2 *$C \sqsubseteq_t v \downarrow w$ and $D \sqsubseteq_t v \downarrow w$ iff $(n_C, n_D)vn \in G^{\times}_{\infty}$ and $(n_C, n_D)wn \in G^{\times}_{\infty}$ for a node n in $G^{\times}_{\infty}$.*

In particular, $G^{\times}_{\infty}$ represents the lcs of the concept descriptions C and D in $\mathcal{S}$. Formally, this means:

Corollary 4 *The (infinite) concept description $C_{G^{\times}_{\infty}}$ corresponding to $G^{\times}_{\infty}$ is the lcs of C and D, i.e., i) $C, D \sqsubseteq_t C_{G^{\times}_{\infty}}$ and ii) $C, D \sqsubseteq_t E'$ implies $C_{G^{\times}_{\infty}} \sqsubseteq_t E'$ for every concept description E' in $\mathcal{S}$.*

5.2 Characterizing the Existence of an Lcs

Let C, D be concept descriptions in $\mathcal{S}$ and let the graphs G_C, G_D, G, G^{∞}_C, G^{∞}_D, and $G^{\times}_{\infty}$ be defined as above.

It turns out that $G^{\times}_{\infty}$ can be used to characterize the existence of the lcs E of C and D. The existence depends on whether there is a finite or an infinite number of a certain kind of nodes which we will call same-as nodes.

Definition 10 *A node n of an (infinite) description graph H is called a* same-as node *if there exist two distinct nodes n_1, n_2 in H and an attribute $a \in \mathcal{A}$ such that (n_1, a, n) and (n_2, a, n) are edges in H.*

The graph depicted in Figure 6 contains an infinite number of same-as nodes. We will show that this is a sufficient and necessary condition for the lcs of C and D not to exist.

Before proving the main theorem of this section, we need to show a lemma that will be useful in the subsequent sections as well. We shall use the notation $n_0 u \cdot n_1 \cdot v n_2 \in H$ to describe a path in H labeled uv from node n_0 to n_2 that passes through node n_1 after u (i.e., $n_0 u n_1 \in H$ and $n_1 v n_2 \in H$); we generalize this the obvious way to interpret $n_0 u_1 \cdot n_1 \cdot u_2 \cdot n_2 \cdot u_3 n_3 \in H$.

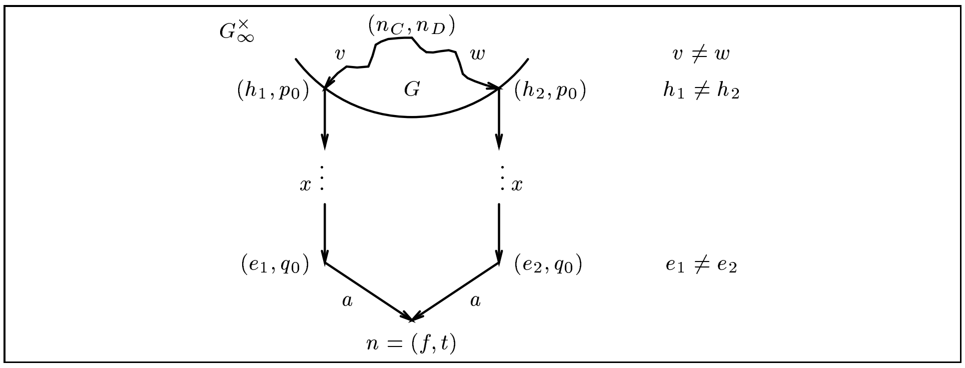

Figure 7: same-as nodes in $G^{\times}_{\infty}$

Lemma 8 *Given a node f in G_C and a tree-node t in G^{∞}_{D}, the node $n = (f, t)$ in $G^{\times}_{\infty}$ is a same-as node iff*

- *there exist nodes (h_1, p_0), (h_2, p_0) in G, $h_1 \neq h_2$;*
- *(e_1, q_0), (e_2, q_0) in $G^{\times}_{\infty}$, where e_1, e_2 are distinct nodes in G_C and q_0 is a node in G^{∞}_{D}; and*
- *$a \in \mathcal{A}$ and $v, w, x \in \mathcal{A}^*$, $v \neq w$, where $\mathcal{A}$ is the set of attributes in C,*

such that $(n_C, n_D)v \cdot (h_1, p_0) \cdot x \cdot (e_1, q_0) \cdot a(f, t)$ is a path in $G^{\times}_{\infty}$ as well as $(n_C, n_D)w \cdot (h_2, p_0) \cdot x \cdot (e_2, q_0) \cdot a(f, t)$ (see Figure 7) and the successor of p_0 in these paths is a tree node in G^{∞}_{D}.

Proof: The if direction is obvious. We proceed with the only-if direction and assume that n is a same-as node in $G^{\times}_{\infty}$. Let p_0 be the youngest ancestor of t in G^{∞}_{D}. In particular, p_0 is a node in G_D and there exists $p_0 x \cdot q_0 \cdot at$ in G^{∞}_{D} with $a \in \mathcal{A}$ and $x \in \mathcal{A}^*$ such that the successor of p_0 in this path is a tree node in G_D. Since $G^{\times}_{\infty}$ is connected, we can conclude that there exist nodes h_1, h_2, e_1, e_2 in G_C such that $(h_1, p_0)x \cdot (e_1, q_0) \cdot a(f, t)$ and $(h_2, p_0)x \cdot (e_2, q_0) \cdot a(f, t)$ are paths in $G^{\times}_{\infty}$. In particular, xa is a label of a path from h_1 to f in G_C. Consequently, the label xa only consists of attributes contained in C.

Furthermore, since t has only one predecessor in G_D^∞, namely, the a-predecessor q_0, we know that $e_1 \neq e_2$, and consequently, since G_C is deterministic, $h_1 \neq h_2$. Otherwise, n could not be a same-as node.

Finally, since $G_\infty^\times$ is connected there are paths (in G) from (n_C, n_D) to (h_1, p_0) and (h_2, p_0) labeled v, resp. w. As G is deterministic and $h_1 \neq h_2$, it follows $v \neq w$. ■

The main results of this section is stated in the next theorem. As a direct consequence of this theorem, we obtain that there exists no lcs in $\mathcal{S}$ for the concept descriptions C_0 and D_0 of our example.

Theorem 6 *The lcs of C and D exists iff the number of same-as nodes in $G_\infty^\times$ is finite.*

Proof. We start by proving the only-if direction. For this purpose, we assume that $G_\infty^\times$ contains an infinite number of same-as nodes and show that there is no lcs for C and D in $\mathcal{S}$.

Since $G_\infty^\times$ contains only a finite number of nodes of the form (f, g) where f and g are non-tree nodes, the remaining (infinitely many) same-as nodes are of the form (f, t) or (t, f). This is because a node of the form (t_1, t_2), where t_1, t_2 are tree nodes, cannot be a same-as node in $G_\infty^\times$ since it has only one direct predecessor. Thus, w.l.o.g., $G_\infty^\times$ has an infinite number of same-as nodes $n_i = (f_i, t_{1,i})$ such that f_i is a node in G_C and $t_{1,i}$ is a tree node in G_D^∞. According to Lemma 8, for every same-as node n_i there exist nodes $h_{1,i}, h_{2,i}, e_{1,i}, e_{2,i}$ in G_C, $p_{0,i}$ in G_D, and $q_{0,i}$ in G_D^∞ as well $a_i \in \mathcal{A}$ and $x_i \in \mathcal{A}^*$ with the properties stated in Lemma 8.

Since G_C and G_D are a finite objects, the number of tuples of the form $h_{1,i}, h_{2,i}, e_{1,i}, e_{2,i}, f_i, a_i$ is finite. Thus, there must be an infinite number of i's yielding the same tuple h_1, h_2, e_1, e_2, f, a. In particular, $h_1 \neq h_2$ and $e_1 \neq e_2$ are nodes in G_C and there is an infinite number of same-as nodes of the form $n_i = (f, t_{1,i})$.

Finally, as in the lemma, let v, w be the label of paths (in G) from (n_C, n_D) to (h_1, p_0) and (h_2, p_0).

Now, assume there is an lcs E of C and D in $\mathcal{S}$. According to Corollary 4, $E \equiv_t C_{G_\infty^\times}$. Let G_E be the finite canonical graph for E with root n'. By Proposition 2 and Lemma 8 we know $E \sqsubseteq_t vx_ia \downarrow wx_ia$. From Algorithm 2 it follows that there are words v', w', and u such that $vx_ia = v'u$ and $wx_ia = w'u$, where the paths in G_E starting from n' labeled v', w' lead to the same node in G_E.

If $u \neq \varepsilon$, then $u = u'a$ for some word u'. Then, Algorithm 2 ensures $E \sqsubseteq_t vx_i \downarrow wx_i$. However, by Lemma 8 we know that the words vx_i and wx_i lead to different nodes in $G^{\times}_{\infty}$, namely, $(e_1, q_{0,i})$ and $(e_2, q_{0,i})$, which, with Proposition 2, leads to the contradiction $E \not\sqsubseteq_t vx_i \downarrow wx_i$. Thus, $u = \varepsilon$.

As a result, for every $i \geq 1$ there exists a node q_i in G_E such that $n'vx_iaq_i$ and $n'wx_iaq_i$ are paths in G_E. Because G_E is finite there exist $i, j \geq 1$, $i \neq j$, with $q_i = q_j$. By Algorithm 2, this implies $E \sqsubseteq_t vx_ia \downarrow wx_ja$. On the other hand, the path in $G^{\times}_{\infty}$ starting from (n_C, n_D) with label vx_ia leads to the node n_i and the one for wx_ja leads to n_j. Since $n_i \neq n_j$ by definition, from Proposition 2 it follows $E \not\sqsubseteq_t vx_ia \downarrow wx_ja$, which is a contradiction.

This shows that there is no lcs of C, D in $\mathcal{S}$ which completes the proof of the only-if direction.

We now prove the if direction of Theorem 6. For this purpose, we assume that $G^{\times}_{\infty}$ has only a finite number of same-as nodes. Note that every same-as node in $G^{\times}_{\infty}$ has only a finite number of direct predecessors: i) a node of the form (g_1, g_2) in G has only predecessors in G; ii) if t is a tree node and g a non-tree node, then a predecessor of (g, t) in $G^{\times}_{\infty}$ is of the form (g', t') where t' is the unique predecessor (tree or non-tree node) of t and g' is a non-tree node. Since the number of nodes in G_C and G_D is finite, in both cases we only have a finite number of predecessors. But then, the spanning tree T of $G^{\times}_{\infty}$ coincides with $G^{\times}_{\infty}$ except for a finite number of edges because if T does not contain a certain edge, then this edge leads to a same-as node. As a result, $C_{G^{\times}_{\infty}}$ is a concept description in $\mathcal{S}$ because it is a finite conjunct of same-as equalities. By Corollary 4, $C_{G^{\times}_{\infty}}$ is the lcs of C and D. ■

If $v \downarrow w$ is a conjunct in $C_{G^{\times}_{\infty}}$, then v and w lead from the root of $G^{\times}_{\infty}$ to a same-as node. As shown in the proof of Lemma 8, same-as nodes are of the form $(f, g), (f, t)$, or (t, f), where t is a tree node and f, g are non-tree nodes. Consequently, v and w must be paths in G_C or G_D. Thus, they only contain attributes occurring in C or D. Therefore, we get

Corollary 5 *If the lcs of two concept description C and D in $\mathcal{S}$ exists, then there is a concept description in $\mathcal{S}$ that is equivalent to the lcs and that contains only attributes already occurring in C or D.*

Therefore, when asking for the existence of an lcs, we can w.o.l.g. assume that the set of attributes $\mathcal{A}$ is finite. This fact will be used in the following section.

5.3 Deciding the Existence of an Lcs

From Lemma 8 and Theorem 6 we can derive a corollary which we will use to specify a decision algorithm for the existence of an lcs of two concept descriptions in $\mathcal{S}$. To state the corollary we need to introduce the regular language $L_{G_C}(q_1, q_2) := \{w \in \mathcal{A}^* \mid$ there is a path from the node q_1 to q_2 in G_C labeled $w\}$. Moreover, let $a\mathcal{A}^*$ denote the set $\{aw \mid w \in \mathcal{A}^*\}$ for any attribute $a \in \mathcal{A}$, where $\mathcal{A}$ is a finite alphabet.

Corollary 6 *$G^{\times}_{\infty}$ contains an infinite number of same-as nodes iff either (i) there exist nodes (h_1, p_0), (h_2, p_0) in G as well as nodes f, e_1, e_2 in G_C, $a, b \in \mathcal{A}$ such that*

1. *$h_1 \neq h_2$, $e_1 \neq e_2$;*
2. *p_0 has no b-successor in G_D;*
3. *(e_1, a, f), (e_2, a, f) are edges in G_C; and*
4. *$L_{G_C}(h_1, e_1) \cap L_{G_C}(h_2, e_2) \cap b\mathcal{A}^*$ is an infinite set of words;*

or
(ii) the same statement as (i) but with switched rôles for C and D.

Proof. We first prove the only-if direction. Assume that $G^{\times}_{\infty}$ contains an infinite number of nodes. Then, w.l.o.g., we find the configuration in $G^{\times}_{\infty}$ described in the proof of Theorem 6. In particular, we have an infinite number of same-as nodes $n_i = (f, t_{1,i})$. Observe that there is only one path from p_0 to $q_{0,i}$ in G^{∞}_{D}. Consequently, the x_i's are pairwise distinct. Since $\mathcal{A}$ is finite, we can assume, w.l.o.g., that all of these paths have prefix $b \in \mathcal{A}$ for some fixed b. This proves condition 4 stated in the corollary.

We now prove the if direction of the corollary. For this purpose, let $bx \in L_{G_C}(h_1, e_1) \cap L_{G_C}(h_2, e_2) \cap b\mathcal{A}^*$. Since p_0 has no b-successor in G_D it follows that there are tree nodes t, t' in G^{∞}_{D} such that $p_0 bx{\cdot}t{\cdot}at' \in G^{\infty}_{D}$. Thus, we have $(h_1, p_0)bx{\cdot}(e_1, t){\cdot}a(f, t') \in G^{\times}_{\infty}$ and $(h_2, p_0)bx{\cdot}(e_2, t){\cdot}a(f, t') \in G^{\times}_{\infty}$. Since $e_1 \neq e_2$, we have $(e_1, t) \neq (e_2, t)$, which means that (f, t') is a same-as node. Analogously, for $by \in L_{G_C}(h_1, e_1) \cap L_{G_C}(h_2, e_2) \cap b\mathcal{A}^*$ there are tree nodes s, s' in G^{∞}_{D} such that $p_0 by{\cdot}s{\cdot}as' \in G^{\infty}_{D}$ and (f, s') is a same-as node in $G^{\times}_{\infty}$. Since bx and by both start with b, and the b-successor of p_0 in G^{∞}_{D} is a tree node, $x \neq y$ implies $s' \neq t'$, and thus, (f, t'), (f, s') are distinct same-as nodes.

This shows that if the set $L_{G_C}(h_1, e_1) \cap L_{G_C}(h_2, e_2) \cap b\mathcal{A}^*$ is infinite, $G^{\times}_{\infty}$ must have an infinite number of same-as nodes, which completes the proof of the corollary. ■

For given nodes (h_1,p_0), (h_2,p_0) in G, attributes $a,b \in \mathcal{A}$, and nodes $f,e_1,e_2 \in G_C$ the conditions 1. to 3. in Corollary 6 can obviously be checked in time polynomial in the size of the concept descriptions C and D. As for the last condition, note that an automaton accepting the language $L_{G_C}(h_1,e_1)\cap L_{G_C}(h_2,e_2)\cap b\mathcal{A}^*$ can be constructed in time polynomial in the size of C. Furthermore, for a given finite automaton it is decidable in time polynomial in the size of the automaton if it accepts an infinite language (see [26] for details). Thus, condition 4. can be tested in time polynomial in the size of C and D as well. Finally, since the size of G and G_C is polynomial in the size of C and D, only a polynomial number of configurations need to be tested. Hence, we have shown

Corollary 7 *For given concept descriptions C and D in $\mathcal{S}$ it is decidable in time polynomial in the size of C and D if there is an lcs for C and D in $\mathcal{S}$.*

5.4 Computing the Lcs

In this section we first look at the minimal size of an lcs, and then present an exponential time algorithm which computes the lcs of two concept descriptions in $\mathcal{S}$. As it turns out, the minimal size of an lcs can be exponential in the size of the two concept descriptions. This is a stronger result than that presented in [20], since in their work it has only been shown that the lcs of a sequence of concept descriptions in $\mathcal{S}$ can grow exponentially.

In order to show that the lcs might be of exponential size we consider the following example, where $\mathcal{A} := \{a,b,c,d\}$. For an attribute α, let α^k, $k \geq 0$, denote the word $\alpha\cdots\alpha$ of length k. We define

$$\begin{aligned} C' &:= a \downarrow b, \\ D_k &:= \bigsqcap_{i=1}^{k} ac^i \downarrow ad^i \sqcap \bigsqcap_{i=1}^{k} bc^i \downarrow bd^i \sqcap ac^k a \downarrow bc^k a. \end{aligned}$$

The corresponding canonical description graphs $G_{C'}$ and G_{D_k} are depicted in Figure 8.

A finite graph representing the lcs of C' and D_k is depicted in Figure 9 for $k=2$. This graph can easily be derived from $G^{\infty}_{C'} \times G^{\infty}_{D_k}$. The graph comprises two binary trees of height k, and thus, it contains at least 2^k nodes. In the following, we will show that there is no canonical description graph G_{E_k} (with root n_0) representing the lcs E_k of C' and D_k with less

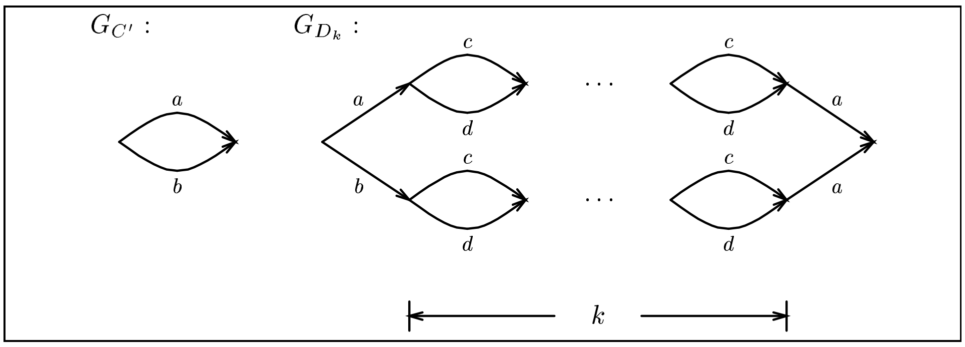

Figure 8: The canonical description graphs for C' and D_k

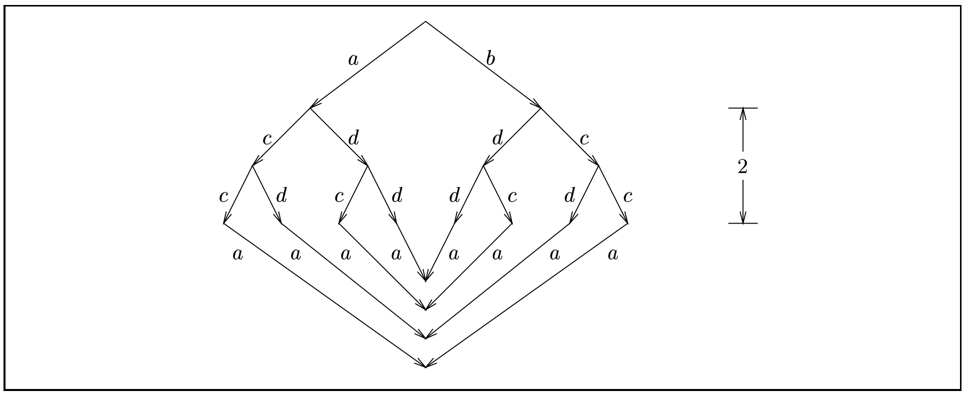

Figure 9: A finite graph representing the lcs of C' and D_2

than 2^k nodes. Let $x \in \{c, d\}^k$, i.e., x is a word of length k over $\{c, d\}$, and let $v := axa$, $w := bxa$. Using the canonical description graphs $G_{C'}$ and G_{D_k} it is easy to see that $C' \sqsubseteq_t v \downarrow w$ and $D_k \sqsubseteq_t v \downarrow w$. Thus, $E_k \sqsubseteq_t v \downarrow w$. By Algorithm 2, this means that there are words v', w', u such that $v = v'u$, $w = w'u$, and there are paths from n_0 labeled v' and w' in G_{E_k} leading to the same node in G_{E_k}. Suppose $u \neq \varepsilon$. By Algorithm 2 this implies $E_k \sqsubseteq_t ax \downarrow bx$. But according to G_D, $D \not\sqsubseteq_t ax \downarrow bx$. Therefore u must be ε. This, proves that in G_{E_k} there is a path from n_0 labeled axa for every $x \in \{c, d\}^k$. Hence, there is a path for every ax. Now, let $y \in \{c, d\}^k$, $x \neq y$. If the paths for ax and ay from n_0 in G_{E_k} lead to the same node, then this implies $E_k \sqsubseteq_t ax \downarrow ay$ which is a contradiction to $C' \not\sqsubseteq_t ax \downarrow ay$. As a result, ax and ay lead to different nodes in G_{E_k}. Since $\{c, d\}^k$ contains 2^k words, this shows that G_{E_k} has at least 2^k nodes.

Now, observe that the canonical graph of a concept description in $\mathcal{S}$ is linear in the size of the concept. Thus, we have proved

Theorem 7 *If the lcs of two concept descriptions in $\mathcal{S}$ exists, then it can be exponential in the size of the concept descriptions.*

As a result, there can be no algorithm for computing the lcs of concept descriptions in $\mathcal{S}$ only using polynomial space. However, one can specify an exponential time algorithm.

Algorithm 3

Input: *concept descriptions C, D in $\mathcal{S}$, for which the lcs exists in $\mathcal{S}$;*

Output: *lcs of C and D in $\mathcal{S}$;*

1. *Compute $G' := G_C \times G_D$;*

2. *For every combination*

   - *of nodes (h_1, p_0), (h_2, p_0) in $G = G_C \times G_D$, $h_1 \neq h_2$;*
   - *$a \in \mathcal{A}$, e_1, e_2, f in G_C, $e_1 \neq e_2$, where (e_1, a, f) and (e_2, a, f) are edges in G_C*

   extend G' as follows: Let $G_{h_1,t}$, $G_{h_2,t}$ be two trees representing the (finite) set of words in

   $$L := \left(L_{G_C}(h_1, e_1) \cap L_{G_C}(h_2, e_2) \cap \bigcup_{b \notin succ(p_0)} b\mathcal{A}^* \right) \cup \{\varepsilon \mid a \notin succ(p_0)\}$$

   where $succ(p_0) := \{b \mid p_0$ has a b-successor$\}$. Furthermore, the set of nodes of $G_{h_1,t}$, $G_{h_2,t}$, and G' are assumed to be disjoint. Now, replace the root of $G_{h_1,t}$ by (h_1, p_0), the root of $G_{h_2,t}$ by (h_2, p_0), and extend G' by the nodes and edges of these two trees. Finally, add a new node n_v for every word v in L, and for each node of the trees $G_{h_1,t}$ and $G_{h_2,t}$ corresponding to v, add an edge with label a from it to n_v. The extension is illustrated in Figure 10.

3. *The same as in step 2, with switched rôles for C and D.*

4. *Compute the canonical graph of G', which we will call G' again. Then, output the concept description $C_{G'}$ of G'.*

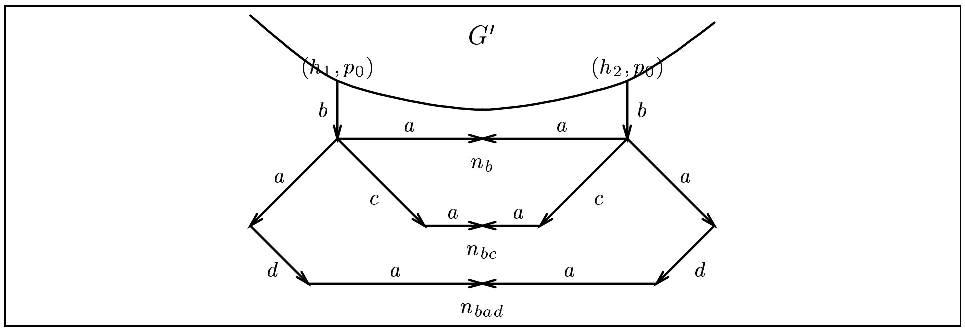

Figure 10: The extension at the nodes (h_1,p_0), (h_2,p_0) in G' where $L = \{b, bc, bad\}$

Proposition 3 *The translation $C_{G'}$ of the graph G' computed by Algorithm 3 is the lcs E of C and D.*

Proof. It is easy to see that if there are two path in G' labeled y_1 and y_2 leading from the root (n_C, n_D) to the same node, then $G^{\times}_{\infty}$ contains such paths as well. Consequently, $(E \equiv_t) G^{\times}_{\infty} \sqsubseteq_t G'$.

Now, assume $E \sqsubseteq_t y_1 \downarrow y_2$, $y_1 \neq y_2$. By Proposition 2 we know that there are paths in $G^{\times}_{\infty}$ labeled y_1 and y_2 leading to the same node n. W.l.o.g, we may assume that n is a same-as node in $G^{\times}_{\infty}$. Otherwise, there exist words y_1', y_2', u with $y_1 = y_1'u$, $y_2 = y_2'u$ such that y_1' and y_2' lead to a same-as node. If we can show that G' contains paths labeled y_1' and y_2' leading to the same node, then by Algorithm 2 this is sufficient for $G' \sqsubseteq_t y_1 \downarrow y_2$. So let n be a same-as node. We distinguish two cases:

1. If n is a node in $G = G_C \times G_D$, then the paths for y_1 and y_2 are paths in G. Since G is a subgraph of G' this shows that y_1 and y_2 are paths in G', which by Algorithm 2 implies $C_{G'} \sqsubseteq_t y_1 \downarrow y_2$.

2. Assume n is not a node in G. Then, since n is a same-as node, we know that n is of the form (f, t) or (t, f) where f is a non-tree node and t is a tree node. By symmetry, we may assume that $n = (f, t)$. Now it is easy to see that there exist nodes h_1, h_2, e_1, e_2 in G_C, p_0 in G_D, and a tree node q_0 in G^{∞}_D as well as $a \in \mathcal{A}$ and $x, v, w \in \mathcal{A}^*$ as specified in Lemma 8 such that $y_1 = vxa$ and $y_2 = wxa$. But then, with $h_1, h_2, e_1, e_2, p_0, f$ and a the preconditions of Algorithm 3 are satisfied and $x \in L$. Therefore, by construction of G' there are paths labeled y_1, resp. y_2, from the root to the same node. ∎

We note that the product G of G_C and G_D can be computed in time polynomial in the size of C and D. Furthermore, there is only a polynomial number of combinations of nodes (h_1, p_0), (h_2, p_0) in G, e_1, e_2, f in G_C, $a \in \mathcal{A}$. Finally, the finite automaton for L can be computed in time polynomial in the size of C and D. This automaton is acyclic, which implies that the set of words accepted by this automaton is at most exponential in the size of the automaton. Consequently, trees representing the words in L can be computed in time exponential in the size of C and D.

Corollary 8 *If the lcs of two concept descriptions in $\mathcal{S}$ exists, then it can be computed in time exponential in the size of the concept descriptions.*

6 Conclusion

Attributes — binary relations that can have at most one value – have been distinguished in many knowledge representation schemes and other object-centered modeling languages. This had been done to facilitate modeling and, in description logics, to help identify tractable sets of concept constructors (e.g., restricting same-as to attributes). In fact, same-as restrictions are quite important from a practical point of view, because they support the modeling of actions and their components (e.g., [7]).

A second distinction, between attributes as total versus partial functions had not been considered so essential until now. This paper has shown that this distinction can sometime have significant effects.

In particular, we have first shown that the approach for computing subsumption of Classic concepts with total attributes, presented in [10], can be modified to accommodate partial attributes, by treating partial attributes as roles until they are guaranteed to have at least one filler, in which case they are "converted" to total attributes. As a result, we obtain polynomial-time algorithms for subsumption and consistency checking in this case also.

In the case of computing least common subsumers, which was introduced as a technique for learning non-propositional descriptions of concepts, we first noted that several of the papers in the literature [18, 24] (implicitly) used partial attributes, when considering Classic. Furthermore, these papers used a weaker version of the "concept graphs" employed in [10], which make the results only hold for the case of same-as restrictions that do not generate "cycles". Furthermore, the algorithm in [24] does not handle inconsistent concepts, which can easily arise in Classic concepts as a result of conflicts between lower and upper bounds of roles.

Therefore, we have provided an lcs algorithm together with a formal proof of correctness for full CLASSIC with partial attributes. In this case, the lcs always exists, and it can be computed in time polynomial in the size of the two initial concept descriptions. As suggested in [20], there are sequences of concept descriptions for which the lcs might grow exponentially in the size of the sequence.

To complete the picture, and as the main part of the paper, we then examined the question of computing lcs in the case of total attributes. Surprisingly, the situation here is very different from the partial attribute case (unlike with subsumption). First, for the language $\mathcal{S}$ the lcs may not even exist. (The existence of the lcs mentioned in [20] is due to an inadvertent switch to partial semantics for attributes.) Nevertheless, the existence of the lcs of two concept descriptions can be decided in polynomial time. But if the lcs exists, it might grow exponentially in the size of the concept descriptions, and hence the computation of the lcs may take time exponential in the size of the two given concept descriptions.

As an aside, we note that it has been pointed out in [20] that concept descriptions in $\mathcal{S}$ correspond to a finitely generated right congruence. Furthermore, in this context the lcs of two concept descriptions is the intersection of right congruences. Thus, the results presented in this paper also show that the intersection of finitely generated right congruences is not always a finitely generated right congruence, and that there is a polynomial algorithm for deciding this question. Furthermore, if the intersection can be finitely generated, then the generating system might be exponential and can be computed with an exponential time algorithm in the size of the generating systems of the given right congruences.

The results in this paper therefore lay out the scope of the effect of making attributes be total or partial functions in a description logic that supports the same-as constructor. Moreover, we correct some problems and extend results in the previous literature.

We believe that the disparity between the results in the two cases should serve as a warning to other researchers in knowledge representation and reasoning, concerning the importance of explicitly considering the difference between total and partial attributes.

References

[1] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineer-*

ing, 20(3):347–383, 1996.

[2] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.

[3] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*, pages 452–457, Sydney (Australia), 1991. Morgan Kaufmann.

[4] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In O. Herzog and A. Günter, editors, *Proceedings of the 22nd Annual German Conference on Artificial Intelligence, KI-98*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany, 1998. Springer–Verlag.

[5] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.

[6] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[7] A. Borgida and P. Devanbu. Adding more "DL" to IDL: towards more knowledgeable component inter-operability. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 378–387, Los Angeles, CA USA, 1999. ACM.

[8] A. Borgida and D.W. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In R.J. Brachman and R. Reiter H.J. Levesque, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 33–43, Toronto, Canada, 1989. Morgan Kaufmann.

[9] A. Borgida and R. Küsters. What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Technical Report DCS-TR-391, Rutgers University, USA, 1999. Available via ftp://ftp.cs.rutgers.edu/pub/technical-reports/.

[10] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[11] M. Buchheit, F.M. Donini, W. Nutt, and A. Schaerf. A refined architecture for terminological systems: Terminology = Schema + View. *Artificial Intelligence*, 99(2):209–260, 1999.

[12] D. Calvanese, G. De Giacomo, and M. Lenzerini. What can knowledge representation do for semi-structured data? In *Proceedings of the 16th National Conference of the American Association for Artificial Intelligence, AAAI-98*, pages 205–210. AAAI Press/The MIT Press, 1998.

[13] D. Calvanese, G. De Giacomo, and M. Lenzerini. Modeling and querying semi-structured data. *Network and Information Systems*, 2(2):253–273, 1999.

[14] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 84–89. Morgan Kaufmann, 1999.

[15] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In A.G. Cohn, L.K. Schubert, and S.C. Shapiro, editors, *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13. Morgan Kaufmann, 1998.

[16] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120. Morgan Kaufmann, Los Altos, 1994.

[17] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 1999. To appear.

[18] W. W. Cohen and H. Hirsh. Learnability of description logics with equality constraints. *Machine Learning*, 17(2/3):169–199, 1994.

[19] W. W. Cohen and H. Hirsh. Learning the classic description logic: Theoretical and experimental results. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, San Francisco, Calif., 1994. Morgan Kaufmann.

[20] W.W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 754–760, San Jose, CA, July 1992. MIT Press.

[21] P.T. Devanbu and M.A. Jones. The use of description logics in KBSE systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(2):141–172, 1997.

[22] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[23] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[24] M. Frazier and L. Pitt. Classic learning. *Machine Learning Journal*, 25:151–193, 1996.

[25] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In C. MacNish, D. Pearce, and Luís Moniz Pereira, editors, *Proceedings of the Fourth European Workshop on Logics in Artificial Intelligence (JELIA'94)*, volume 838 of *Lecture Notes in Artificial Intelligence*, pages 332–346. Springer-Verlag, 1994.

[26] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory.* Addison-Wesley Publ. Co., 1979.

[27] Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI-91)*, pages 471–476, 1991.

[28] D.L. McGuinness and J.R. Wright. An industrial strength description logic-based configurator platform. *IEEE Intelligent Systems*, 13(4):66–77, 1998.

[29] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[30] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, editor, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431, Toronto, Ont., 1989.