Automatic Strengthening of Graph-Structured Knowledge Bases

Or: How to Identify Inherited Content in Concept Graphs

Vinay Chaudhri, Nikhil Dinesh, Stijn Heymans, Michael Wessel[†]

SRI International
Artificial Intelligence Center
333 Ravenswood Avenue, Menlo Park, CA 94025-3493, USA
firstname.lastname@sri.com
([†] = corresponding author)

Abstract. We consider the problem of identifying inherited content in knowledge representation structures called *concept graphs (CGraphs)*. A CGraph is a visual representation of a concept; in the following, CGraphs and concepts are used synonymously. A CGraph is a node- and edge-labeled directed graph. Labeled (binary) edges represent relations between nodes, which are considered instances of the concepts in their node labels. CGraphs are arranged in a taxonomy (is-a hierarchy). The taxonomy is a directed acyclic graph, as multiple inheritance is allowed. A taxonomy and set of CGraphs is called a graph-structured knowledge base (GSKB).

A CGraph can inherit content from other CGraphs – intuitively, if C and D are CGraphs, then C may contain content inherited from D, i.e. labeled nodes and edges "from D" can appear in C, if D is a direct or indirect superconcept of C, or if C contains a node being labeled with either D or some subclass of D. In both cases, C is said to refer to D.

This paper contains three contributions. First, we describe and formalize the problem from a logical point of view and give a first-order semantics for CGraphs. We show that the identification of inherited content in CGraphs depends on some form of hypothetical reasoning and is thus not a purely deductive inference task, as it requires unsound reasoning. Hence, this inference is different from the standard subsumption checking problem, as known from description logics (DLs) [1]. We show that the *provenance problem* (from where does a logical atom in a CGraph get inherited?) strongly depends on the solution to the *co-reference problem* (which existentials in the first-order axiomatization of concepts as formulas denote identical domain individuals?) We demonstrate that the desired inferences can be obtained from a so-called *strengthened GSKB*, which is an augmented variant of the input GSKB. We present an algorithm which augments and strengthens an input GSKB, using model-theoretic notions. Secondly, we are addressing the problem from a graph-theoretic point of view, as this perspective is closer to the actual implementation. We show that we can identify inherited content by computing so-called *concept coverings*, which induce inherited content from superconcepts by means of *graph morphisms*. We argue that the algorithm solves a challenging (NP-hard) problem. Thirdly, we apply the algorithm to the large-scale biological knowledge base from the AURA project [2], and present a preliminary evaluation of its performance.

1 Introduction

Graph-structured knowledge bases (GSKBs) occur naturally in many application domains, for example, if biological knowledge is modeled graphically by means of *concept graphs (CGraphs)* as in the AURA project [2], see Fig. 1:

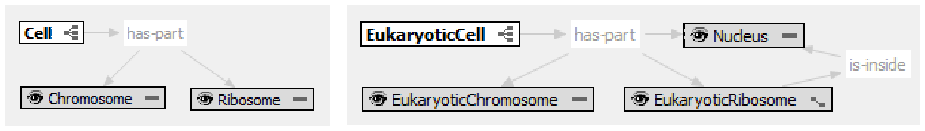

Fig. 1. (Simplified) Concept Graphs for $Cell$ and $EukaryoticCell$ in AURA.

In the AURA project, such CGraphs were modeled by subject-matter experts (SMEs) from the pages of a biology college textbook [3], following a detailed "text to CGraph" encoding process [4]:

S1 Every $Cell$ has part a $Ribosome$ and a $Chromosome$.
S2 Every $EukaryoticCell$ is a $Cell$.
S3 Every $EukaryoticCell$ has part a $EukaryoticRibosome$, a $EukaryoticChromosome$, a $Nucleus$, such that the $EukaryoticChromosome$ is inside the $Nucleus$.
S4 Every $EukaryoticRibosome$ is a $Ribosome$.
S5 Every $EukaryoticChromosome$ is a $Chromosome$.

The CGraphs in Fig. 1 naturally represent $S1$ and $S3$. However, the taxonomic information expressed by $S2, S4$ and $S5$, is visualized in a separate taxonomy view (hence, the superconcepts are not explicit in the CGraphs in Fig. 1).

Intuitively, a CGraph C *can inherit content* from a CGraph D if D is either a direct or indirect superconcept of C, or if C contains a node with either D or some subconcept of D in its label.

In this paper, we are addressing the following question: **Given a GSKB with a fixed taxonomy, which content in the CGraphs is inherited, and from where?** For example, if we assume that $EukaryoticCell$ has $Cell$ as a superconcept, as expressed by $S2$, then it also seems plausible to assume that its $EukaryoticChromosome$ part was inherited from $Cell$ as a $Chromosome$ which then got *specialized*. Under this assumption it seems reasonable to assume that the individual represented by the $EukaroticChromosome$ node in $EukaryoticCell$ is hence identical to the individual represented by the $Chromosome$ node in $Cell$ – we are saying that those nodes should be *co-referential*. However, this co-reference is not explicit in the CGraphs. We might consider the graphs as *underspecified* as it is neither possible to prove nor to disprove equality of those nodes resp. individuals represented by them in the logical models.

Similar questions of co-referentiality related to inherited content do arise if we use natural language as our primary means of knowledge representation, e.g. consider the sentences $S1$ and $S3$. Is the $EukaryoticChromosome$ that $S3$ is talking about actually the *same* $Chromosome$ that is mentioned in $S1$? These kinds of question are studied to some extent in the field of computational linguistics under the term *anaphora resolution* [5], [6]. We will use the term *co-reference resolution* in the following. Obviously, natural language is also underspecified in that sense.

From a *logical point of view*, the sentences $S1$ to $S5$ (and their corresponding CGraphs) naturally correspond to FOPL formulas of the following kind (in the following, we will be using the comma in consequents to denote conjunctions); the *graph-structured knowledge base (GSKB)* of $S1$ to $S5$ (and the corresponding CGraphs) then looks as follows:

S1 $\forall x : \mathit{Cell}(x) \Rightarrow \exists x_1, x_2 :$
$\quad \mathit{hasPart}(x, x_1), \mathit{Ribosome}(x_1),$
$\quad \mathit{hasPart}(x, x_2), \mathit{Chromosome}(x_2)$
S2 $\forall x : \mathit{EukaryoticCell}(x) \Rightarrow \mathit{Cell}(x)$
S3 $\forall x : \mathit{EukaryoticCell}(x) \Rightarrow \exists x_3, x_4, x_5 :$
$\quad \mathit{hasPart}(x, x_3), \mathit{Euk.Ribosome}(x_3),$
$\quad \mathit{hasPart}(x, x_4), \mathit{Euk.Chromosome}(x_4),$
$\quad \mathit{hasPart}(x, x_5), \mathit{Nucleus}(x_5), \mathit{inside}(x_4, x_5)$
S4 $\forall x : \mathit{Euk.Ribosome}(x) \Rightarrow \mathit{Ribosome}(x)$
S5 $\forall x : \mathit{Euk.Chromosome}(x) \Rightarrow \mathit{Chromosome}(x)$

Unary predicates represent *concepts*, and binary predicates predicates *relations*. The concept D is called a *superconcept* of C, with $C \neq D$, if $\forall x : C(x) \Rightarrow D(x), \ldots$ holds. Vice versa, C is called a *subconcept* of D then.

Following our example, we would like to prove that a $\mathit{EukaryoticCell}$ has a $\mathit{Chromosome}$ part which gets inherited from Cell, and indeed, the following entailment holds:

$$\{S1, S2, S4, S5\} \models$$
$$\forall x : \mathit{EukaryoticCell}(x) \Rightarrow \exists y : \mathit{hasPart}(x, y), \mathit{Chromosome}(y),$$

with $S3$ being removed here, as the entailment would hold true trivially otherwise, of course. This testifies that "having a Chromosome part" is inherited from Cell to $\mathit{EukaryoticCell}$.

However, a closer look reveals that our question was a bit too general, and that we really ought to be able to prove that *the Chromosome part inside the Nucleus of a EukaryoticCell* should be *the* Chromosome inherited from Cell; intuitively, the atoms $hasPart(x, x_4), Chromosome(x_4)$ in $S3$ should be "inherited" from $S1$ (their *provenance* should be Cell), and hence, be logically redundant in a sense. In order to check if those atoms from $S3$ are indeed inherited from some other concept in the GSKB to $\mathit{EukaryoticCell}$ and hence redundant, we can tentatively remove the atoms under question from $S3$, yielding $S3^-$, and then check if the original $S3$ axiom is entailed. Following this procedure, $S3^-$ reads

S3$^-$ $\forall x : \mathit{EukaryoticCell}(x) \Rightarrow \exists x_3, x_4, x_5 :$
$\quad \mathit{hasPart}(x, x_3), \mathit{Euk.Ribosome}(x_3),$
$\quad\quad \mathit{Euk.Chromosome}(x_4),$
$\quad \mathit{hasPart}(x, x_5), \mathit{Nucleus}(x_5), \mathit{inside}(x_4, x_5)$

Note that of the two atoms to be removed from $S3$, namely $hasPart(x, x_4)$, $Chromosome(x_4)$, only the former is actually present in $S3$ – obviously, we do not want to remove $Euk.Chromosome(x_4)$ from $S3$ for this test. Using $S3^-$ in the GSKB instead of $S3$ *unfortunately* yields:

$$\{S1, S2, S3^-, S4, S5\} \not\models S3$$

since already

$$\begin{array}{l}\{S1, S2, S3^-, S4, S5\} \not\models \\ \quad \forall x : \mathit{EukaryoticCell}(x) \Rightarrow \exists y_1, y_2 : \\ \qquad\qquad \mathit{hasPart}(x, y_1), \mathit{Chromosome}(y_1), \mathit{inside}(y_1, y_2).\end{array}$$

The reason that the desired consequence *does not* hold is obviously that there is *no way to prove the co-referentiality / equality* between the $Chromosome$ inherited from $Cell$, denoted by x_2 in $S1$, and the $EukaryoticChromosome$ denoted by x_4 in $S3^-$ *inside the Nucleus* x_5, and hence, the corresponding model individuals cannot be used for satisfaction / interpretation of y_1, y_2 in the above entailment query. If we could prove that x_2 from $S1$ must be equal to x_4 in $S3^-$ in the context of $EukaryoticCell$, then the entailment would hold. But all we can say is that there is some (potentially different) $Chromosome$ part inherited from $Cell$ to $EukaryoticCell$, which is not necessarily inside its $Nucleus$. So, we do not get the desired inference *unless* we *strengthen* the axiomatic content of our GSKB somehow such that those equalities hold.

One way of *strengthening* this GSKB in order to *get the desired inferences* is to *skolemize* the existentials, and establish co-references by virtue of equality atoms between ("local and inherited") Skolem function terms, as shown below. We hence call the following GSKB a *strengthened version* of the original GSKB:

S1' $\forall x : \mathit{Cell}(x) \Rightarrow$
$$\begin{array}{l}\mathit{hasPart}(x, f_1(x)), \mathit{Ribosome}(f_1(x)), \\ \mathit{hasPart}(x, f_2(x)), \mathit{Chromosome}(f_2(x))\end{array}$$

S3' $\forall x : \mathit{EukaryoticCell}(x) \Rightarrow$
$$\begin{array}{l}\mathit{hasPart}(x, f_3(x)), \mathit{Euk.Ribosome}(f_3(x)), \\ \mathit{hasPart}(x, f_4(x)), \mathit{Euk.Chromosome}(f_4(x)), \\ \mathit{hasPart}(x, f_5(x)), \mathit{Nucleus}(f_5(x)), \\ \mathit{inside}(f_4(x), f_5(x))), \\ f_3(x) = f_1(x), f_4(x) = f_2(x)\end{array}$$

Note that we are now getting the desired inference as follows. First we again remove the atoms under investigation from $S3'$ to get $S3'^-$:

S3'$^-$ $\forall x : \mathit{EukaryoticCell}(x) \Rightarrow$
$$\begin{array}{l}\mathit{hasPart}(x, f_3(x)), \mathit{Euk.Ribosome}(f_3(x)), \\ \qquad \mathit{Euk.Chromosome}(f_4(x)), \\ \mathit{hasPart}(x, f_5(x)), \mathit{Nucleus}(f_5(x)), \\ \mathit{inside}(f_4(x), f_5(x))), \\ f_3(x) = f_1(x), f_4(x) = f_2(x)\end{array}$$

and then we observe that the following entailment holds:

$$\{S1', S2, S3'^{-}, S4, S5\} \models S3'$$

because for an x satisfying $EukaryoticCell$, we can inherit $hasPart(x, f_2(x))$, $Chromosome(f_2(x))$ from $Cell$, and due to the equality atom $f_4(x) = f_2(x)$, we can establish co-referentiality / equality of the inherited $Chromosome$ with the $Euk.Chromosome$ in $EukaryoticCell$ and hence, the restrictions modeled in $S3'^{-}$ for $f_4(x)$ apply to the inherited $f_2(x)$, hence satisfying all of the necessary conditions in $S3'$. This shows that the $hasPart(x, f_4(x))$, $Chromosome(f_4(x))$ is redundant in $S3'$, or *inherited from somewhere*. More specifically, we can also verify *from where* these atoms are inherited, by removing axioms tentatively from the GSKB one by one, and re-checking the entailment after each removal – for example, if the entailment stops to hold as soon as we remove $S1'$ from the KB, then we know that $S1$ plays a crucial role in entailing these atoms, and that those atoms are, in that sense, "inherited" from $Cell$. Thus, $Cell$ should be their provenance then. This gives us a procedure for computing the provenance of atoms; see below for more details.

We have just demonstrated that inherited content can frequently only be recognized correctly in CGraphs if the GSKB was strengthened, i.e., equality between existentials / Skolem function terms can be proven. Shouldn't it then be *obligatory* that the equalities required for establishing the desired inferences are *explicitly specified in the first place?* We believe that the answer can be *no.* Certainly, the required equalities could be added by hand as in the examples above, but this is often tedious, even if there is some tool support (e.g. a graphical CGraph editor). However, as demonstrated, the input may also be considered *naturally underspecified* in the sense that *co-references are not explicit.* We therefore propose an *automatic co-reference resolution algorithm* which hypothesizes and adds these equality atoms automatically. This algorithm necessarily has to rely on *some sort of guessing,* and is hence in the realm of *hypothetical / logically unsound inference procedures*. The presented algorithm produces a strengthened GSKB in which these co-references are explicit.

Another benefit of a GSKB is that it often *reduces modeling effort.* For example, suppose we updated $Cell$ by saying that its $Ribosome$ is *inside* $Cytosol$:

S1b $\forall x : Cell(x) \Rightarrow \exists x_1, x_2, x_6 :$
$hasPart(x, x_1), Ribosome(x_1),$
$hasPart(x, x_2), Chromosome(x_2),$
$inside(x_1, x_6), Cytosol(x_6)$

We would like to derive that this also holds for the $Euk.Ribosome(x_3)$ in $EukaryoticCell$ in $S3$ – analog to the case of the $Chromosome$ and $EukraryoticChromosome$ it is reasonable to assume that $Ribosome$ and $EukaryoticRibosome$ are co-referential as well, and indeed, we are getting this inference automatically with

S1b' $\forall x : Cell(x) \Rightarrow$
$hasPart(x, f_1(x)), Ribosome(f_1(x)),$
$hasPart(x, f_2(x)), Chromosome(f_2(x)),$
$inside(f_1(x), f_0(x)), Cytosol(f_0(x))$

as follows:

$$\begin{array}{l}\{S1b', S2, S3', S4, S5\} \models \\ \quad \forall x : \mathit{EukaryoticCell}(x) \Rightarrow \\ \qquad \exists y_1, y_2 : \mathit{hasPart}(x, y_1), \mathit{Euk.Ribosome}(y_1), \mathit{inside}(y_1, y_2), \mathit{Cytosol}(y_2)\end{array}$$

Note again that this entailment does *not* hold for $\{S1b, S2, S3, S4, S4\}$. the utility of the strengthened GSKB $\{S1b', S2, S3', S4, S4\}$ is hence that we do not need to remodel the fact that the $\mathit{EukaryoticRibosome}$ of a $\mathit{EukaryoticCell}$ is inside $\mathit{Cytosol}$ – we can simply inherit it from Cell, and it would be *logically redundant* to add these atoms to $S3'$.

In this paper, we first present the *logical perspective* on this problem. We present a so-called GSKB strengthening algorithm, which, given an input GSKB such as $\{S1b, S2, S3, S4, S5\}$, produces a strengthened GSKB similar to $\{S1b', S2, S3', S4, S5\}$, by using Skolemization and equality atoms between Skolem terms. From the strengthened GSKB we can compute the *provenance of atoms* and *co-references*, hence answering the question *from where did content get inherited?* We will use a model-theoretic notion of *preferred models* in order to characterize the desired inferences that we wish to obtain from the underspecified GSKB, as illustrated by the examples above. The information in the preferred model(s) is then used to produce the strengthened GSKB. Obviously, deciding entailment of atoms, and hence the provenance problem, are in general undecidable in FOPL(=), but decidable in the considered fragment of FOPL(=).

A further contribution of this paper is a description of the *actual implementation, which is best understood and described from a graph-theoretic perspective.* It is obvious that we can define CGraphs as node- and edge labeled graphs. We will argue that the problem of identifying inherited content (the provenance problem) and the co-reference problem can also be understood as solving a variant of the well-known *maximum common subgraph isomorphism (MCS) problem*. This problem, which is NP-hard, can be stated as follows:

Input: Two graphs G1 and G2.
Output: The largest subgraph of G1 isomorphic to a subgraph of G2.

We are considering morphisms instead of isomorphisms, as the node mappings do not have to be injective in our case.[1] Moreover, we also need to be more flexible regarding the node- and edge-label matching conditions, as inherited labels can be specialized in subconcepts. The resulting morphism-based graph algorithm is called the *GSKB covering algorithm* in the following.

We have implemented this novel GSKB covering in a scalable way, and have successfully applied it to the AURA GSKB [2], [4]. The AURA GSKB is the basis of the intelligent question answering textbook Inquire, see [7][2], and currently contains around

[1] There is a possibility that two inherited nodes get equated / merged into one node, which means that those two (or more) nodes will be mapped to one node by the morphism – the mapping is obviously no longer injective then.
[2] This video won the AAAI 2012 video award.

6430 concept graphs, with a total of 93,254 edges and 53,322 nodes, hence on average 14.5 edges and 8.2 nodes per concept. The biggest graph has 461 nodes and 1,308 edges. A third contribution of this paper is hence a preliminary evaluation and critical review of the performance of this algorithm on the AURA GSKB.

The paper is structured as follows: We first address the problem from a logical point of view. We formally define CGraphs and the GSKB framework, as well as the required notions of strengthened GSKBs, and the semantic notion of preferred models. We then present the GSKB-strengthening algorithm and show that a strengthened GSKB has models which are preferred models of the original GSKB, hence giving us the desired additional conclusions required to solve the co-reference and provenance problems. We then introduce the graph-based framework underlying the actual implementation. We formally define CGraph morphisms, as well as so-called CGraph patchworks and GSKB coverings, which describe how content is inherited from other CGraphs via morphisms throughout the whole GSKB. We present an algorithm for computing such a GSKB covering, and illustrate that the computed CGraph morphisms can be used to decide provenance and co-referentiality. Next we apply the GSKB covering algorithm to the AURA GSKB and evaluate its performance. We then discuss related work, and conclude with a summary and an outline for future work.

2 Graph Structured Knowledge Bases – The Logical Perspective

As outlined in the Introduction, in this Section we formalize the notion of GSKBs from a first-order logic perspective, and show how provenance of atoms and co-referentiality of variables can be formalized and computed from a strengthened GSKB. We present the so-called GSKB strengthening algorithm.

2.1 Definitions

In the following, we denote an atom or a conjunction of atoms with free variables $\{x, x_1, \ldots, x_n\}$ as $\varphi(x, \boldsymbol{x})$, with $\boldsymbol{x} = (x_1, \ldots, x_n)$. *Graph-structured knowledge bases* (GSKBs) are formulated in first order-logic with equality, FOPL(=). We assume that there is a function $terms$ which returns the terms in a formula, e.g. $terms(R(t_1, t_2)) \triangleq \{t_1, t_2\}$:

Definition 1. *Basic Definitions. Let $\mathcal{C}$ ($\mathcal{R}$) be a countably infinite set of unary (binary) predicate names, and $\mathcal{F} = \{f_1, f_2, \ldots\}$ be a countably infinite set of unary function names – hence, $(\mathcal{C} \cup \mathcal{R}, \mathcal{F})$ is the signature. Elements in $\mathcal{C}$ ($\mathcal{R}$) are called concepts (relations). Moreover, let $\mathcal{X} = \{x, x_1, x_2, \ldots\}$ be a countably infinite set of variables. A* GSKB term *is a term t such that $t \in \mathcal{X}$, or $t = f_i(x)$, or $t = f_i(f_j(x))$, with $\{f_i, f_j\} \subseteq \mathcal{F}$. Let t, t_1, t_2 be GSKB terms:*

GSKB atoms: *Let $\{C, D\} \subseteq \mathcal{C}$, $R \in \mathcal{R}$, $\{v, w\} \subseteq \mathcal{X}$. Then, $C(v)$ and $C(f_i(x))$ are concept atoms, and $R(v, w)$, $R(x, f_i(x))$ are relation atoms. Moreover, there are equality and inequality atoms of the following form: $f_i(x) = f_j(x)$, $f_i(x) = f_j(f_k(x))$, $f_j(f_k(x)) = f_i(x)$, and $f_i(x) \neq f_j(x)$, with i, j, k pairwise unequal.*

GSKB rule: *For a concept C, a formula $\rho_C \triangleq \forall x : C(x) \Rightarrow \exists! \boldsymbol{x} : \varphi(x, \boldsymbol{x})$ is called a GSKB rule, where $\varphi(x, \boldsymbol{x}) = \bigwedge_{i \in 1 \ldots m} \alpha_i$ is finite conjunction of GSKB atoms. This is shorthand for $\forall x : C(x) \Rightarrow \exists \boldsymbol{x} : \mathit{pairwise\_unequal}(x, \boldsymbol{x}) \wedge \varphi(x, \boldsymbol{x})$, $\boldsymbol{x} = (x_1, \ldots, x_n)$, with the macro $\mathit{pairwise\_unequal}(x, \boldsymbol{x}) \triangleq \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \wedge \bigwedge_{1 \leq i \leq n} x_i \neq x$.*
For a concept C with $\rho_C = \forall x : C(x) \Rightarrow \exists! \boldsymbol{x} : \varphi(x, \boldsymbol{x})$, denote $\varphi(x, \boldsymbol{x}) = \bigwedge_{i \in 1 \ldots m} \alpha_i$ as a set *by $\tau_C \triangleq \{\alpha_1, \ldots, \alpha_m\}$, and $\mathit{terms}(C) \triangleq \bigcup_{\alpha \in \tau_C} \mathit{terms}(\alpha)$.*
We require that the terms in $\mathit{terms}(\rho_C)$ are connected *to x: for all $t \in \mathit{terms}(C)$, $\mathit{connected}(x, t)$ holds, where connected is defined as follows: $\mathit{connected}(t_1, t_2)$ holds if $\{R(t_1, t_2), R(t_2, t_1)\} \cap \tau_C \neq \emptyset$, or there is some t s.t. $\mathit{connected}(t_1, t)$ and $\mathit{connected}(t, t_2)$ holds.*

GSKB: *A finite set of GSKB rules Σ in which there is at most one rule per concept.*

Input GSKB: *A GSKB which is function-free and without equality atoms.*

Auxiliary notions: *Given a GSKB Σ, we refer to the set of concepts used in Σ as $\mathit{concepts}(\Sigma)$, and $\tau_{C,\Sigma}$ to refer to the consequent of $\rho_C \in \Sigma$. We extend the other definitions to accept a Σ argument as well, e.g., $\mathit{terms}(C, \Sigma)$, etc.*

For example, $\{S1b, S2, S3, S4, S5\}$ is an (underspecified) input GSKB, and $\{S1b', S2, S3', S4, S5\}$ is a *strengthened* (output) GSKB; however, we need to replace the $\exists$ quantifier with $\exists!$. The formal definition of *strengthened* GSKB is given below. Note that sometimes the strengthening algorithm will not add anything, and hence output will equal input, e.g. for $\{S4, S5\}$.

We require that an input GSKB must be *coherent*:

Definition 2. *Coherent GSKB and coherent model. A GSKB Σ is coherent if there is standard first-order model $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\mathcal{I} \models \Sigma$, in which every concept C mentioned in Σ is interpreted in a non-empty way: $C^{\mathcal{I}} \neq \emptyset$. Such a model is called a* coherent model.

Moreover, we define standard notions such as $\mathit{superconcepts}$ as follows:

Definition 3. *Auxiliary Definitions. Let C be a concept, Σ be a GSKB. We then define the following functions and predicates w.r.t. Σ:*

- $\mathit{asserted\_types}(C, \Sigma) \triangleq \{D \mid D(t) \in \tau_{C,\Sigma}\}$
- $\mathit{has\_asserted\_type}_{\Sigma}(C, D)$
  iff $D \in \mathit{asserted\_types}(C, \Sigma)$
- $\mathit{asserted\_superconcepts}(C, \Sigma) \triangleq \{D \mid D(x) \in \tau_{C,\Sigma}\}$
- $\mathit{has\_asserted\_superconcept}_{\Sigma}(C, D)$
  iff $D \in \mathit{asserted\_superconcepts}(C, \Sigma)$
- $\mathit{superconcepts}(D, \Sigma)$
  $\triangleq \{E \mid \mathit{has\_asserted\_superconcept}^{+}_{\Sigma}(D, E)\ \}$
- $\mathit{has\_superconcept}_{\Sigma}(C, D)$
  iff $D \in \mathit{superconcepts}(C, \Sigma)$
- $\mathit{all\_types}_{\Sigma}(C)$
  $\triangleq \{E \mid D \in \mathit{asserted\_types}(C, \Sigma),$
  $E \in \mathit{superconcepts}(D, \Sigma)\}$
  $\cup\ \mathit{superconcepts}(C, \Sigma)$

– *$has\_type_{\Sigma}(C, D)$ iff $D \in all\_types(C, \Sigma)$*

where R^+ denotes the transitive closure of relation R.

We require that the relations $has\_superconcept_{\Sigma}$ and has_type_{Σ} are irreflexive and define:

Definition 4. *Admissible GSKB. An input GSKB Σ is called admissible if Σ is coherent, $has\_superconcept_{\Sigma}$ and has_type_{Σ} are irreflexive, and if there are no implied concept atoms in the rules: for all $C \in concepts(\Sigma)$, if $D(t) \in \tau_{C,\Sigma}$, then for all $E \in superconcepts(D, \Sigma)$: $E(t) \notin \tau_{C,\Sigma}$.*

The following is straightforward:

Proposition 1. *Every admissible GSKB Σ has a coherent, finite model.*

Proof. Given that we do not support negation of concepts or relations, and given that inequality atoms are only introduced by the $\exists!$ quantor, inconsistencies such as $x \neq x$ cannot occur. Moreover, since GSKB $has\_superconcept_{\Sigma}$ and has_type_{Σ} are irreflexive, the GSKB is acyclic, and the consequent of every rule can be "unfolded", analog to the unfolding of an acyclic TBox in description logics [1]. This produces a finite consequent for every rule. Next, for every $\rho_C \in \Sigma$, C can be instantiated s.t. $i_C \in C^{\mathcal{I}}$ holds, and we can easily satisfy the existentials in the consequent by creating one instance per variable. The process terminates and produces a model of Σ which is coherent and finite.

We need a notion of connectedness on models:

Definition 5. *Predicate $connected$ on models. Let $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of Σ. For $i, j \in \Delta_{\mathcal{I}}$, we define $connected_{\mathcal{I}}(i, j)$ if, for some $R \in \mathcal{R}$, $\{(i, j), (j, i)\} \cap R^{\mathcal{I}} \neq \emptyset$, or there is some $k \in \Delta_{\mathcal{I}}$ s.t. $connected_{\mathcal{I}}(i, k)$ and $connected_{\mathcal{I}}(k, j)$.*

In the following we are considering admissible GSKBs only, and we are interested in their *preferred models*. The intuition behind the notion of a preferred model is the following: for every concept C, there should be a *prototypical model* of C which is *not* also a model of some non-superconcept of C, in the form of a connected graph that "mirrors" the atoms in $\tau_{C,\Sigma}$ – due to the $pairwise\_unequal$ macro there will be at least one individual per variable in ρ_C in this model. Moreover, the prototypical model for C also contains inherited "graphs" from concepts in $all\_types_{\Sigma}(C)$. Hence, the graph satisfying the atoms $\tau_{C,\Sigma}$ is only a subgraph of the full model for C. Most importantly, the notion of a preferred model captures the intuition that inherited content can be specialized, and hence should give rise to co-references: in the prototypical model for $EukaryoticCell$, the $Chromosome$ inherited from its superclass $Cell$ will be represented by the same individual as its own local $Euk.Chromosome$. Note that this minimizes the extension of $Chromosome$. The same argument applies to arbitrary conjunctions: we will *not* identify the inherited $Chromosome$ with the local $Euk.Ribosome$, as this would result in a model in which $Chromosome \wedge Euk.Ribosome$ is interpreted non-empty, and there are models in which this conjunction is interpreted by the empty set. These intuitions are formalized as follows:

Definition 6. *Preferred model of admissible GSKB Σ. Let Σ be an admissible GSKB, and $\mathcal{I} \models \Sigma$ be a coherent finite model. Then, $\mathcal{I}$ is called* preferred *if the following holds:*

1. *for every concept $C \in concepts(\Sigma)$, there is (at least) one $i \in C^{\mathcal{I}}$ s.t. for all D, if $has\_superconcept(D, C)$, then $i \notin D^{\mathcal{I}}$ – hence, there is at least one element which is "unique" to C, and denoted by i_C.*
2. *for $C \in concepts(\Sigma)$, define $participants_{\mathcal{I}}(C) \triangleq \{j \mid connected_{\mathcal{I}}(i_C, j)\}$. Then, for all $C, D \in concepts(\Sigma)$, with $C \neq D$, the following holds: $participants_{\mathcal{I}}(C) \cap participants_{\mathcal{I}}(D) = \emptyset$.*
3. *for every non-empty subset $\mathcal{CS} \subseteq concepts(\Sigma)$, there is no preferred model $\mathcal{I} \neq \mathcal{I}'$, with $\Delta_{\mathcal{I}'} \subseteq \Delta_{\mathcal{I}}$ s.t. $\bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'} \subset \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}}$.*

Consider the preferred models of $\{S1, S2, S3, S4, S5\}$. We are forced to have at least one "unique" $Cell$ which is not a $EukaryoticCell$, due to 1. Otherwise, every $Cell$ would acquire the properties of $EukaryoticCell$s, which is not desirable. Moreover, none of the individuals connected to that unique $Cell$ are shared by another concept, due to 2. Hence, the concept models have the forms of "non-overlapping graphs", and inherited content is "mapped in". We are forced to minimize the extension of every concept, as well as of every conjunction of concepts. This prevents models in which, for example, $Ribosome^{\mathcal{I}} \cap Euk.Chromosome^{\mathcal{I}} \neq \emptyset$ holds, as there are smaller models in which they are interpreted disjointly: $Ribosome^{\mathcal{I}} \cap Euk.Chromosome^{\mathcal{I}} = \emptyset$. Note that the inequality atoms in Σ only prevent "merging" of variables within the same formula, but the individual for $Chromosome(x_2)$ inherited from $Cell$ could in principle be made co-referential with the local $Euk.Ribosome(x_3)$ in $EukaryoticCell$. *This is prevented in a preferred model.* Also, looking at the model of $EukaryoticCell$, the co-reference between the from $Cell$ inherited $Chromosome(x_2)$ and its own local $Euk.Chromosome(x_4)$ is made explicit, since this will result in the smallest (extension of) $Chromosome^{\mathcal{I}}$. A model in which a $EukaryoticCell$ would have two different $Chromosome$s would be larger and in violation to 3. So, we only make those conjunction true in a preferred model that we *have to make true* - for example, $Cell^{\mathcal{I}} \cap EukaryoticCell^{\mathcal{I}} \neq \emptyset$, due to $S2$, and there is no model in which this conjunction is interpreted by a smaller set.

Note that a preferred model is not a "minimal" model in the classical sense. Consider $\forall x : C(x) \Rightarrow \exists! x1 : R(x, x1), D(x1)$, $\forall x : SubC(x) \Rightarrow \exists! x2 : C(x), R(x, x2), E(x2)$. In the classical minimal model $\mathcal{I}$, we would have $\#\Delta_{\mathcal{I}} = 2$, and it would satisfy $D \wedge E$. Also, $C^{\mathcal{I}} = SubC^{\mathcal{I}}$. But this is not what we want. It violates 1, 2, as well as 3. The preferred model will need at least 5 nodes.

In principle, there can be more than one preferred model and hence, more than one strengthened version of the GSKB. For example, consider the GSKB

$$\begin{aligned} &C(x) \Rightarrow \exists! x_1 : R(x, x_1), E(x_1) \\ &SubC(x) \Rightarrow \exists! x_2, x_3 : C(x), \\ &\qquad R(x, x_2), E(x_2), F(x_2), \\ &\qquad R(x, x_3), E(x_3), G(x_3). \end{aligned}$$

Here, x_1 in C can be co-referential with either x_2 in $SubC$, or with x_3.

In the next section, we will show the following constructively, by specifying an algorithm which constructs a preferred model for a given admissible GSKB Σ:

Proposition 2. *Every admissible GSKB has a preferred model.*

We can now state the purpose of the GSKB strengthening algorithm more clearly. Given an admissible GSKB Σ (note that this is an input GSKB), we are interested in finding a strengthened version of Σ:

Definition 7. *Strengthened version of Σ. Given an admissible (input) GSKB Σ, we are calling Σ' a* strengthened version *of Σ if the following holds:*

1. *for every rule $\rho_C \in \Sigma$, there is a rule $\rho'_C \in \Sigma'$ that uses only the variable x: $terms(\rho'_C) \cap \mathcal{X} = \{x\}$.*
2. *if $\mathcal{I}' \models \Sigma'$ is a standard first-order model of Σ' which is coherent, then $\mathcal{I}' \models \Sigma$, and $\mathcal{I}'$ is a preferred model for Σ. Hence, $\Sigma' \models \Sigma$.*

From a strengthened GSKB, we can decide provenance and co-reference as follows:

Definition 8. *Provenance and co-reference determination. Let C be a concept, Σ' be a strengthened GSKB, and $\mathcal{P} \subseteq \tau_{C,\Sigma'}$. With $\beta = \bigwedge_{\alpha \in \mathcal{P}} \alpha$, we then say that β (and hence all the atoms in $\mathcal{P}$) are*

– *local (or asserted) in C if*
  $\Sigma' \setminus \{\rho_C\} \cup \{\forall x : C(x) \Rightarrow \bigwedge_{\alpha \in \tau_{C,\Sigma'} \setminus \mathcal{P}} \alpha\}$
  $\not\models \forall x : C(x) \Rightarrow \beta$,
– *and inherited otherwise. More specifically, β (and $\mathcal{P}$) is inherited from D, iff $D(t) \in \tau_{C,\Sigma'}$, and $\beta' = \bigwedge_{\alpha \in \mathcal{P}'} \alpha$ with $\mathcal{P}' = \{\alpha_{[f_i(t) \Rightarrow f_i(x)]} \mid \alpha \in \mathcal{P}\}$ is local in D, and there is no more general $SupD$ with $has\_superconcept_{\Sigma'}(D, SupD)$ such that β (and $\mathcal{P}$) is inherited from $SupD$.*

Moreover, given concepts C, D, two GSKB terms $t_1 \in terms(C)$, $t_2 \in terms(D)$ are said to be co-referential in Σ' iff either $t_1 = t_2 = x$, $t_1 = f_i(x)$, $t_2 = f_j(f_k(x))$, or $t_2 = f_i(x)$, $t_1 = f_j(f_k(x))$, and $\Sigma' \models (\forall x : C(x) \Rightarrow f_i(x) = f_j(x) \vee (\forall x : D(x) \Rightarrow f_i(x) = f_j(x))$.

Note that a conjunction β is local as soon as *some* atom is already local. Hence, if a complex conjunction β (resp. $\mathcal{P}$) is local, this does *not* mean that *all* its atoms have to be local – some atoms may be inherited.

Proposition 3. *Provenance and co-reference are decidable in a strengthened GSKB Σ'.*

The proof is given in the next Section.

2.2 Constructing a Strengthened GSKB

The algorithm works by performing the following steps:

1. Produce the skolemized version of Σ, Σ_S, by bringing every rule in Σ into Skolem normal form. The skolemized axioms contain no nested function terms, only terms of the form $f_i(x)$ and x. Let $\mathcal{O} \triangleq \{o_C \mid C \in concepts(\Sigma)\}$ be a set of constants, and also add $\{C(o_C) \mid C \in concepts(\Sigma)\}$ to Σ_S.

2. Construct the *minimal Herbrand model* $\mathcal{I}_{\mathcal{H}} = (\Delta_{\mathcal{H}}, \cdot^{\mathcal{I}_{\mathcal{H}}})$ of Σ_S. The minimal Herbrand model is unique and finite, given that Σ is admissible (and does not contain disjunctions in the consequents). Note that the minimal Herbrand model will automatically satisfy the inequality atoms, and it will also satisfy points 1 and 2 from Definition 6, due to the set of constants $\mathcal{O}$ which are instantiated as $\{C(o_C) \mid C \in \mathit{concepts}(\Sigma)\} \subseteq \Sigma_S$, and with the exception of x, there are no shared terms in the rules of Σ_S, as Skolemization creates fresh function symbols for every variable. Thus, o_C represents the root individual of the unique model for concept C, with $o_C^{\mathcal{I}_{\mathcal{H}}} = i_C, i_C \in C^{\mathcal{I}_{\mathcal{H}}}$.
3. Transform $\mathcal{I}_{\mathcal{H}}$ into a preferred model of Σ, $\mathcal{I}_{\mathcal{A}} = (\Delta_{\mathcal{A}}, \cdot^{\mathcal{I}_{\mathcal{A}}})$. $\Delta_{\mathcal{A}}$ is the quotient set of $\Delta_{\mathcal{H}}$ under the $=$ equivalence relation, $\Delta_{\mathcal{A}} = \Delta_{\mathcal{H}} \backslash =$. Hence, the elements of $\Delta_{\mathcal{A}}$ represent the equivalence classes of equated Skolem ground terms from the Herbrand universe $\Delta_{\mathcal{H}}$. This step is non-deterministic, as there may be more than one preferred model for Σ.
4. Use $\mathcal{I}_{\mathcal{A}}$ to construct a strengthened GSKB Σ' from Σ_S which is satisfied by that model. Use the equivalent clusters in $\Delta_{\mathcal{A}}$ to generate equality atoms.
5. From Σ' it is possible to decide the provenance and the co-reference problem, on a syntactic basis.

Since steps 1 and 2 are standard and well-know [8], let us define the algorithm for step 3. We need two more utility notions before we can proceed:

Definition 9. *Relations $\mathcal{E}$ and $\mathcal{U}$, and equivalence classes. Let $\mathcal{I}_{\mathcal{H}} = (\Delta_{\mathcal{H}}, \cdot^{\mathcal{I}_{\mathcal{H}}})$ be the minimal unique Herbrand model after step 2 of Σ_S above. Let $\mathcal{E}$ be a binary relation over terms from the Herbrand universe $\Delta_{\mathcal{H}}$, and define*

$$\begin{aligned}\mathit{closure}(\mathcal{E}) \triangleq \textstyle\bigcup_{C \in \mathit{concepts}(\Sigma), k \in \Delta_{\mathcal{H}}} & \\ & \{(f_1(k), f_2(k)) \mid (f_1(o_C), f_2(o_C)) \in \mathcal{E}^{\circledast}\} \cup \\ & \{(f_1(f_2(k)), f_3(k)) \mid (f_1(f_2(o_C)), f_3(o_C)) \in \mathcal{E}^{\circledast}\} \cup \\ & \{(f_1(f_2(k)), f_3(f_4(k))) \mid (f_1(f_2(o_C)), f_3(f_4(o_C))) \in \mathcal{E}^{\circledast}\}\end{aligned}$$

where $\cdot^{\circledast}$ denotes the reflexive, symmetric, and transitive closure *of a relation. Let $[i]^{\mathcal{E}} \triangleq \{j \mid (i,j) \in \mathit{closure}(\mathcal{E})\}$. Moreover, let $\mathcal{U} \triangleq \{[i]^{\mathcal{E}} \neq [j]^{\mathcal{E}} \mid i_1 \in [i]^{\mathcal{E}}, j_1 \in [j]^{\mathcal{E}}, C \in \mathit{concepts}(\Sigma), (i_1 \neq j_1) \in \tau_{C,\Sigma_S}\}$ be the set of inequality atoms.*

Intuitively, $(i,j) \in \mathcal{E}$ represents $i = j$, and $[i]^{\mathcal{E}}$ represents the equivalence class of i. The relation $\mathcal{E}$ (and hence the equivalence classes) will grow as pairs of equated individuals / terms are added by the algorithm given below. Intuitively, the closure operator makes sure that whenever two terms starting from the same root node o_C are equated in the unique model of C, that then this equality will also hold for all its C instantiations in other parts of the model. Note that also $\mathcal{U}$ will grow, representing inferences such as $i \neq j, k \neq l, j = k \Rightarrow i \neq l$.

The algorithm can now be stated as follows:

Algorithm 1 *Construction of a preferred model for Σ. Let $\mathcal{I}_{\mathcal{H}} = (\Delta_{\mathcal{H}}, \cdot^{\mathcal{I}_{\mathcal{H}}})$ be the minimal unique Herbrand model of Σ_S after step 2 above.*

1. *define* $hasRoot(i) \triangleq o_C$ *iff* $connected_{\mathcal{I}_\mathcal{H}}(o_C, i)$ *holds, for every* $C \in concepts(\Sigma)$.
2. *then, non-deterministically apply the following* merging rule *on the model as long as it is applicable:*

   if *there are individuals* $i, j \in \Delta_\mathcal{H}$, $i \neq j$, *with* $hasRoot(i) = hasRoot(j) = o_C$ *and* $ind\_types(i) \subseteq ind\_types(j)$, $i \notin [j]^\mathcal{E}$, $[i]^\mathcal{E} \neq [j]^\mathcal{E} \notin \mathcal{U}$, ***then*** $\mathcal{E} \triangleq \mathcal{E} \cup \{(i, j)\}$.

   Assume the rule application stops with a global maximum of inequality atoms s.t. $\#\mathcal{U}$ *is maximized. Since this is a non-deterministic algorithm, such a run exists, and we can assume that the non-deterministic algorithm will produce it.*
3. *define* $\mathcal{I}_\mathcal{A} = (\Delta_\mathcal{A}, \cdot^{\mathcal{I}_\mathcal{A}})$ *as follows:*
   $\Delta_\mathcal{A} \triangleq \{[i]^\mathcal{E} \mid i \in \Delta_\mathcal{H}\}$, *and for all* $C \in concepts(\Sigma) : C^{\mathcal{I}_\mathcal{A}} \triangleq \{[i]^\mathcal{E} \mid i \in C^{\mathcal{I}_\mathcal{H}}\}$, *for all* $R \in \mathcal{R} : R^{\mathcal{I}_\mathcal{A}} \triangleq \{([i]^\mathcal{E}, [j]^\mathcal{E}) \mid (i, j) \in R^{\mathcal{I}_\mathcal{H}}\}$.

The algorithm terminates, since $\mathcal{I}_\mathcal{H}$ is finite, so there is a finite number of merging possibilities in the rule. The solution which maximizes $\#\mathcal{U}$ can obviously be found by search in a deterministic version.

Lemma 1. $\mathcal{I}_\mathcal{A} = (\Delta_\mathcal{A}, \cdot^{\mathcal{I}_\mathcal{A}})$ *is a preferred model for* Σ.

Proof. Obviously, $\mathcal{I}_\mathcal{A}$ is finite and coherent, as it was constructed by the algorithm based on the unique finite Herbrand model. Assume that $\mathcal{I}_\mathcal{A}$ is not a preferred model for Σ. By construction, $\mathcal{I}_\mathcal{A}$ is a model of Σ_S, as the merging rule preserves the model character of $\mathcal{I}_\mathcal{H}$. Since $\mathcal{I}_\mathcal{H}$ is a model of the skolemized version, it is also a model of Σ, since $\Sigma_S \models \Sigma$ for the skolemized GSKB [8]. Hence, $\mathcal{I}_\mathcal{A}$ is a model of Σ, also.

It remains to show that it is preferred. Assume that it is not. Since points 1 and 2 from Definition 6 are already satisfied by construction, only 3 can be violated. Then, there must be some other model $\mathcal{I}'$ and some $\mathcal{CS} \subseteq concepts(\Sigma)$ such that $\bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'} \subset \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}}$, witnessed by $[i]^\mathcal{E} \in \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}}$ with $[i]^\mathcal{E} \notin \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'}$.

1. If $\bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'} = \emptyset$, then this violates the assumption that $\Sigma_\mathcal{H}$ was a minimal Herbrand model (which does not make things true without need). Hence, $\bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}} = \emptyset$ as well, which contradicts $[i]^\mathcal{E} \in \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}}$.
2. Assume $\mathcal{CS} = \{D\}$ is a single concept name. As $\mathcal{I}_\mathcal{H}$ was a minimal model, the existence of i, with $i \in [i]^\mathcal{E}$, is somehow enforced by Σ_S, hence there is some term $t_i \in terms(C, \Sigma_S)$ with $D \in ind\_types(t_i)$. Moreover, for the same reason, $D^{\mathcal{I}'} \neq \emptyset$, as otherwise it wouldn't be a model, but $i \notin D^{\mathcal{I}'}$. Consequently, there is some $j \in D^{\mathcal{I}'}$ with $i \neq j$. Then, there must also be some $t_j \in terms(C, \Sigma_S)$ with $D \in ind\_types(t_j)$, with $t_i \neq t_j$.
   There are a couple of cases:
   (a) Assume $ind\_type(t_i) \subseteq ind\_types(t_j)$
      i. if $C' = C$ and hence $hasRoot(i) = hasRoot(j) = C$, then $(t_i \neq t_j) \notin \tau_{C,\Sigma_S}$ and $[i]^\mathcal{E} \neq [j]^\mathcal{E} \notin \mathcal{U}$, as otherwise $\mathcal{I}$ would not be a model. But then, the merging rule would have been applied and merged i and j, such that $[i]^\mathcal{E} = [j]^\mathcal{E} = \{i, j\}$. Rule application could not have been blocked by the precondition $[i]^\mathcal{E} \neq [j]^\mathcal{E} \notin \mathcal{U}$, because $\mathcal{I}_\mathcal{A}$ was produced by a run in which $\#\mathcal{U}$ was maximized. This means that the rule will be applicable and equate i and j, contradicting the assumption that the algorithm has terminated.

ii. otherwise, $C \neq C'$, then we don't have to worry: as stated in Definition 6, $\mathit{participants}_{\mathcal{I}_\mathcal{A}}(C) \cap \mathit{participants}_{\mathcal{I}_\mathcal{A}}(C') = \emptyset$.

(b) Assume $\mathit{ind\_type}(t_j) \subseteq \mathit{ind\_types}(t_i)$: analog to the previous case.

(c) Assume $\mathit{ind\_type}(t_i) \not\subseteq \mathit{ind\_types}(t_j)$. Then there is some $E \in \mathit{ind\_type}(t_i), E \notin \mathit{ind\_types}(t_j)$. As $\mathcal{I}_\mathcal{A}$ was a minimal Herbrand model, and there is no way for $[i]^\mathcal{E}$ to "vanish" from $E^{\mathcal{I}_\mathcal{A}}$, there must be $[i]^\mathcal{E} \in E^{\mathcal{I}_\mathcal{A}}$ and hence $[i]^\mathcal{E} \in \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'}$. Contradiction.

3. If $\mathcal{CS} = \{D_1, \ldots, D_n\}$, then there must already be some $\mathcal{CS}' = \{D_m, D_n\}, \mathcal{CS}' \subseteq \mathcal{CS}$ for which we have such an i. If $\mathit{has\_superconcept}(D_m, D_n)$ or vice versa, then there is already some $\mathcal{CS}' = \{D_m\}$, and this is handled by 2. Otherwise, D_m, D_n are not in a sub/superconcept relationship, and corresponding instances are not getting merged by the merging rule. But similar to 2c), this will lead us to conclude that $[i]^\mathcal{E} \in \bigcap_{C \in \mathcal{CS}} C^{\mathcal{I}'}$, contradicting the assumption.

Hence, $\mathcal{I}_\mathcal{A}$ is a preferred model. Note that this proves Proposition 2.

For what remains to be shown is how we can compute a strengthened GSKB from Σ_S and $\mathcal{I}_\mathcal{A}$.

Definition 10. *Construction of strengthened GSKB Σ'. Let Σ_S be the skolemized version of the admissible GSKB, and $\mathcal{I}_\mathcal{A}$ be a preferred model of Σ. We then rewrite the rules in Σ_S as follows; note that $\alpha_{[t_1 \Rightarrow t_2]}$ means "in α, substitute all occurrences of t_1 with t_2":*

$$\Sigma' \triangleq \{\mathit{rewrite}(\rho_C, \mathit{terms}(C, \Sigma_S)) \mid \rho_C \in \Sigma_S\}, \text{with}$$

$$\begin{aligned}
&\mathit{rewrite}(\rho_C, \mathit{terms}) \triangleq C(x) \Rightarrow \\
&\quad \textstyle\bigwedge_{\alpha \in \tau_{C,\Sigma_S}} \alpha \wedge \\
&\quad \textstyle\bigwedge_{t \in \mathit{terms}, t \neq o_C} \mathit{hasRoot}(t, x)_{[o_C \rightarrow x]} \wedge \\
&\quad \textstyle\bigwedge_{t_1, t_2 \in \mathit{terms}, t_1 \neq t_2} t_1 \neq t_{2[o_C \rightarrow x]} \wedge \\
&\quad \textstyle\bigwedge_{t_1 \in \mathit{terms}, t_2 \in [t_1]} t_1 = t_{2[o_C \rightarrow x]}
\end{aligned}$$

In addition, we need the following axioms:

1. $\Sigma' \triangleq \Sigma' \cup \{C(o_C) \mid C \in \mathit{concepts}(\Sigma)\}$
2. $\Sigma' \triangleq \Sigma' \cup \{o_C \neq o_D \mid C, D \in \mathit{concepts}(\Sigma), C \neq D\}$
3. $\Sigma' \triangleq \Sigma' \cup \{\forall x, y, z :$
   $\mathit{hasRoot}(x, y), \mathit{hasRoot}(y, z) \Rightarrow \mathit{hasRoot}(x, z)\ \}$
4. $\Sigma' \triangleq \Sigma' \cup \{\forall x, y :$
   $\mathit{hasRoot}(x, o_C), \mathit{hasRoot}(y, o_D) \Rightarrow x \neq y\ \}$,
   for all $C, D \in \mathit{concepts}(\Sigma), C \neq D$.

Lemma 2. *If $\mathcal{I} \models \Sigma'$, then $\mathcal{I}$ is a preferred model for Σ.*

Proof. As Σ' has been constructed from Σ_S by adding equality atoms to explicitly represent the co-references with inherited Skolem function successors, which have been identified by the merging rule from a preferred model of Σ, it is clear that any model of Σ' will force the same co-references, and hence, satisfy point 3 in Definition 6.

Moreover, point 1 in Definition 10 makes sure that we have non-empty concept models for every concept by requiring an instance, hence satisfying condition 1 in Definition 6. Point 2 in Definition 10 enforces distinctness between those constants, and point 3 declares $hasRoot$ as a transitively closed relation. In combination with the added $hasRoot$ atoms in Σ', and with the axioms in point 4 of Definition 10, this ensures that condition 2 in Definition 6 is satisfied, requiring that the unique concept models do not overlap (no sharing of participants).

Let us return to our example. For $\Sigma = \{S1b, S2, S3, S4, S5\}$ we will get Σ_S as follows:

$$
\begin{array}{l}
Cell(x) \Rightarrow \\
\quad hasPart(x, f_1(x)), Ribosome(f_1(x)), \\
\quad hasPart(x, f_2(x)), Chromosome(f_2(x)), \\
\quad inside(f_1(x), f_0(x)), Cytosol(f_0(x)), \\
\quad pairwise\_unequal(\{x, f_0(x), f_1(x), f_2(x)\}) \\
EukaryoticCell(x) \Rightarrow Cell(x) \\
\quad hasPart(x, f_3(x)), Euk.Ribosome(f_3(x)), \\
\quad hasPart(x, f_4(x)), Euk.Chromosome(f_4(x)), \\
\quad hasPart(x, f_5(x)), Nucleus(f_5(x)), \\
\quad inside(f_4(x), f_5(x)), \\
\quad pairwise\_unequal(\{x, f_3(x), f_4(x), f_5(x)\}) \\
Cell(o_{Cell}), Euk.Cell(o_{Euk.Cell}), Ribosome(o_{Ribosome}) \ldots
\end{array}
$$

If we look at the minimal Herbrand model of Σ_S, we find that the following atoms are satisfied for $o_{Euk.Cell}$:

$$
\begin{array}{l}
hasPart(o_{Euk.Cell}, f_1(o_{Euk.Cell})), \\
hasPart(o_{Euk.Cell}, f_2(o_{Euk.Cell})), \\
inside(f_1(o_{Euk.Cell}), f_0(o_{Euk.Cell})), \\
Ribosome(f_1(o_{Euk.Cell})), \\
Chromosome(f_2(o_{Euk.Cell})), \\
Cytosol(f_0(o_{Euk.Cell})), \\
hasPart(o_{Euk.Cell}, f_3(o_{Euk.Cell})), \\
hasPart(o_{Euk.Cell}, f_4(o_{Euk.Cell})), \\
hasPart(o_{Euk.Cell}, f_5(o_{Euk.Cell})), \\
inside(f_4(o_{Euk.Cell}), f_5(o_{Euk.Cell})), \\
Euk.Ribosome(f_3(o_{Euk.Cell})), \\
Euk.Chromosome(f_4(o_{Euk.Cell})), \\
Nucleus(f_5(o_{Euk.Cell})),
\end{array}
$$

Moreover, there are pairwise inequality atoms between $o_{Euk.Cell}, f_3(o_{Euk.Cell}), f_4(o_{Euk.Cell}), f_5(o_{Euk.Cell})$ and between $o_{Euk.Cell}, f_0(o_{Euk.Cell}), f_1(o_{Euk.Cell}), f_2(o_{Euk.Cell})$.

If we next look at $\mathcal{I}_{\mathcal{A}}$, we will find that $[f_3(o_{Euk.Cell}]) = [f_1(o_{Euk.Cell})] = \{f_3(o_{Euk.Cell}), f_1(o_{Euk.Cell})\}$ holds, and likewise $[f_4(o_{Euk.Cell}]) = [f_2(o_{Euk.Cell})] = \{f_2(o_{Euk.Cell}), f_4(o_{Euk.Cell})\}$. Hence, the desired co-references have been established, e.g., the from $Cell$ inherited $Ribosome(f_1(o_{Euk.Cell}))$ is identified as being co-referential with the "local" $Euk.Ribosome(f_3(o_{Euk.Cell}))$.

The abridged strengthened GSKB Σ' then looks as follows:

$$
\begin{array}{l}
Cell(x) \Rightarrow \\
\quad hasPart(x, f_1(x)), Ribosome(f_1(x)), \\
\quad hasPart(x, f_2(x)), Chromosome(f_2(x)), \\
\quad hasRoot(f_1(x), x), hasRoot(f_2(x), x), \\
\quad pairwise\_unequal(\{x, f_1(x), f_2(x)\}) \\
EukaryoticCell(x) \Rightarrow Cell(x), \\
\quad hasPart(x, f_3(x)), Euk.Ribosome(f_3(x)), \\
\quad hasPart(x, f_4(x)), Euk.Chromosome(f_4(x)), \\
\quad hasPart(x, f_5(x)), Nucleus(f_5(x)), \\
\quad inside(f_4(x), f_5(x)), f_3(x) = f_1(x), f_4(x) = f_2(x), \\
\quad hasRoot(f_3(x), x), hasRoot(f_4(x), x), \\
\quad hasRoot(f_5(x), x), \\
\quad pairwise\_unequal(\{x, f_3(x), f_4(x), f_5(x)\}) \\
Ribosome(o_{Ribosome}), Chromosome(o_{Chromosome}) \\
\ldots o_{Cell} \neq o_{Euk.Cell} \ldots \text{(axiom sets 2–4 from Def. 10)}
\end{array}
$$

We claim that we can decide the provenance problem for the strengthened GSKB Σ' syntactically as follows; also recall that in an admissible KB, the consequents do not contain redundant concept atoms:

Definition 11. *Syntactic provenance of atoms in Σ'. In a strengthened GSKB Σ', for $C \in concepts(\Sigma)$, let $\alpha \in \tau_{C,\Sigma'}$ be an atom:*

– *$\alpha = C(f(x))$ is inherited from D if $D(f_s(x)) \in \tau_{C,\Sigma'}$ with $D \in \{C\} \cup all\_superclasses(C, \Sigma')$ and $f'(f_s(x)) = f(x) \in \tau_{C,\Sigma'}$ with $C(f'(x)) \in \tau_{D,\Sigma'}$, and there is no more general class $SupD$ with $has\_superconcept(D, SupD)$ for which this is also the case.*
– *$\alpha = R(f_1, f_2)$ is inherited from D if $D(f_s(x)) \in \tau_{C,\Sigma'}$ with $D \in \{C\} \cup all\_superclasses(C, \Sigma')$ and $\{f'_1(f_s(x)) = f_1(x), f'_2(f_s(x)) = f_2(x)\} \subseteq \tau_{C,\Sigma'}$ with $R(f'_1, f'_2) \in \tau_{D,\Sigma'}$, and there is no more general class $SupD$ with $has\_superconcept(D, SupD)$ for which this is also the case.*

If α is not inherited from some concept, it is called local *to C.*

Looking at the example GSKB Σ', we see that the atoms $hasPart(x, f_3(x))$ are inherited from $Cell$, due to $f_3(x) = f_1(x)$, and $hasPart(x, f_4(x))$, due to $f_4(x) = f_2(x)$. Consequently, $hasPart(x, f_5(x)), Nucleus(f_5(x)), inside(f_4(x), f_5(x))$ are local to $EukaryoticCell$. Hence, for the original GSKB Σ, $hasPart(x, x_3)$ and $hasPart(x, x_4)$ were inherited from $Cell$, and $hasPart(x, x_5), Nucleus(x_5), inside(x_4, x_5)$ are local to $EukaryoticCell$.

We claim that we can decide the co-reference problem for the strengthened GSKB Σ' syntactically as follows:

Definition 12. *Syntactic co-reference of terms in Σ'. Two terms with $t_1 \in terms(C, \Sigma')$, $t_2 \in terms(D, \Sigma')$ are co-referential, if $t_1 = t_2 = x$, or $t_1(x) = t_2(t) \in \tau_{C,\Sigma'}$, or $t_2(x) = t_1(t) \in \tau_{D,\Sigma'}$ (note that $t = x$, or $t = f_s(x)$).*

Looking at the example GSKB Σ', we see that $f_3(x) = f_1(x)$ are co-referential and hence the $Ribosome$ in $Cell$ is the same as the $Euk.Ribosome$ in $EukaryoticCell$, and likewise for the $Chromosome$ due to $f_4(x) = f_2(x)$.

Lemma 3. *Syntactic provenance according to Def. 11 is sound and complete for deciding semantic provenance according to Def. 8. Syntactic co-reference according to Def. 12 is sound and complete for deciding semantic co-reference according to Def. 8.*

Proof. Soundness is immediate. Completeness is a straightforward too, as Skolem functions are not shared by different consequents in Σ', and Σ was admissible. Moreover, for two different Skolem functions f_i, f_j, with $i \neq j$, $f_i(t) = f_j(t)$ will hold for a certain term t in *all* models of Σ' if and only if this was explicitly enforced by an equality atom. Note that this also proves Proposition 3.

We can generalize these results to the original GSKB Σ as follows. To check the provenance of $\tau_{C,\Sigma}$ we need to keep track during Skolemization which atom $\alpha' \in \tau_{C,\Sigma'}$ corresponds to α, and check the provenance of α' in Σ'. Likewise, to check to co-referentiality of two variables, let t_1 and t_2 be its corresponding (Skolem function) terms in the skolemized versions. Now, x_1 and x_2 are co-referential in Σ iff t_1 and t_2 are co-referential in Σ'. Looking at the example GSKB Σ, we see that x_1 from $S1$ is co-referential with x_3 in $S3$ since $f_3(x) = f_1(x)$ in Σ', and x_2 from $S1$ is co-referential with x_4 in $S3$ due to $f_4(x) = f_2(x)$ in Σ'.

However, given that a GSKB may have more than one strengthened version, "to decide" should be understood in a *credulous* way here. Only in case a provenance information or co-reference holds in *all* strengthened GSKBs, this would be a *skeptical* conclusion; it is clear that all strengthened GSKBs can in principle be constructed, due to finiteness of $\mathcal{I}_{\mathcal{H}}$. We can hence present the main result of this paper as follows:

Corollary 1. *Given a strengthened GSKB Σ', we can decide the provenance and co-reference problem on a syntactic basis. For an admissible (input) GSKB Σ, we can decide the* credulous provenance and credulous co-reference problem *by constructing* a *strengthened GSKB Σ' first, and check there. The* skeptical provenance and skeptical co-reference problem *can be decided by constructing* all *strengthened GSKBs, and checking if a positive answer holds in all of them.*

3 Graph Structured Knowledge Bases – The Graph-Based Perspective

As outlined in the Introduction, in this Section we address the provenance and co-reference problems from a graph-theoretic perspective. GSKB and CGraphs are defined as graph-based notions, and the so-called GSKB covering algorithm is presented as an algorithm which establishes graph morphisms between CGraphs. Those morphisms describe how content is inherited from other CGraphs, throughout the GSKB. The actual implementation of the algorithm is also discussed, together with implementation tricks which make it scalable.

A major drawback of the previously given logic-based framework was the disallowance of *cyclical concepts*, i.e. of concepts in which the transitive closure of the *refers to* (or *uses*) relation is not irreflexive. In the AURA GSKB many concepts are actually cyclical in that sense, for example, the concept $AnimalCell$ refers to $Animal$, and vice versa. We disallowed cyclical concepts in the logic-based formalization in order to ensure the existence of *finite* Herbrand models. In the following *graph-based formalization* we *allow* cyclical concepts and describe a strategy for dealing with them. In addition, we will also show how to handle a relation hierarchy (something which could have been done in the logical formalization, too, but was omitted for the sake of brevity – the same applies to disjointness axioms and other straight-forward representation means).

Intuitively, in case of a refers-to cycle, the algorithm will make a non-deterministic choice, as illustrated by the following example. Consider it is stated in $Animal$ that $Animal$s have $AnimalCell$s as parts, and conversely within $AnimalCell$, that $AnimalCell$s are $partOf$ $Animal$s. In this case the algorithm will tell us that the fact "$Animal$s have $AnimalCell$ parts" is either inherited from $Animal$ to $AnimalCell$, or vice versa, but not both. Hence, a fact (atom, triple) is always owned or local to one concept.

3.1 Definitions

Definition 13. *Graph-Structured Knowledge Base. A graph-structured knowledge base (GSKB) is a tuple $(\mathcal{C}, \mathcal{R}, \mathcal{O}_\mathcal{C}, \mathcal{O}_\mathcal{R}, \mathcal{I}_\mathcal{R}, \mathcal{G})$, where $\mathcal{C}$ is a set of concepts, $\mathcal{R}$ is a set of binary relations, $\mathcal{O}_\mathcal{C} \subseteq \mathcal{C} \times \mathcal{C}$ is the concept taxonomy, and $\mathcal{O}_\mathcal{R} \subseteq \mathcal{R} \times \mathcal{R}$ is the relation hierachy. Both are strict partial orders. For a relation R, $(R, S) \in \mathcal{I}_\mathcal{R}$ means that S is the inverse relation of R, and vice versa: $\mathcal{I}_\mathcal{R} = {\mathcal{I}_\mathcal{R}}^{-1}$, so $\mathcal{I}_\mathcal{R}$ is closed under the symmetric closure. We denote the inverse of R with R^{-1} – note that for every R, either $(R, S) \in \mathcal{I}_\mathcal{R}$, for some $R \neq S$, or $(R, R^{-1}) \in \mathcal{I}_\mathcal{R}$. Moreover, $\mathcal{G}$ is the set of* CGraphs *for $\mathcal{C}$, one per concept. With $\mathcal{G}_C$ we denote the CGraph of concept C.*

Intuitively, $(C, D) \in {\mathcal{O}_\mathcal{C}}^+$ means that D is a superconcept or a more general concept than C, and that it is possible that the CGraph of C contains inherited content from the CGraph of D. But note that C may also contain inherited content from non-superconcepts, which are instantiated in C. CGraphs are defined as follows:

Definition 14. *CGraph of a concept C. A CGraph of a concept C, $\mathcal{G}_C$, is a node- and edge-labeled graph (N_C, E_C, L^N_C, L^E_C), with nodes N_C and edges E_C. There is a special node $root_C \in N_C$, the* root node. *Two edges (n_1, n_2) and (n_3, n_4) are called* adjoined *if they share a node: $\{n_1, n_2\} \cap \{n_3, n_4\} \neq \emptyset$. We require that every node n is connected to the* root *node $root_C$: $(root_C, n) \in E^\otimes_C$, and $E^\otimes_C$ denotes the symmetric and transitive closure of E_C.*

The labeling functions $L^N_C : N_C \mapsto 2^\mathcal{C}$ and $L^E_C : E_C \mapsto 2^\mathcal{R}$ are total functions; labeling with $\emptyset$ is permitted.

The augmented CGraph *of $\mathcal{G}_C$ is $\mathcal{AG}_C$, which is a CGraph that satisfies the following:*

– *$E_C = E_C^{-1}$, so E_C is closed under symmetric closure.*
– *for all $(n_1, n_2) \in E_C$, if $R \in L_C^E(n_1, n_2)$, then $R^{-1} \in L_C^E(n_2, n_1)$.*
– *$L_C^N(n) = \mathcal{L}$, then $\mathcal{L}$ is closed under implied superconcepts: if $D \in \mathcal{L}$, and $(D, E) \in {\mathcal{O}_\mathcal{C}}^+$, then $E \in \mathcal{L}$.*
– *$L_C^E(n_1, n_2) = \{S \mid R \in L_C^E(n_1, n_2),$ and $S = R$ or $(R, S) \in {\mathcal{O}_\mathcal{R}}^+\}$.*

As a shorthand, we say that $F(n)$ is a concept atom *in C if $n \in N_C$ and $F \in L_C^N(n)$. Analogously, we are saying that $R(n_1, n_2)$ is a* relation atom *in C if $(n_1, n_2) \in E_C$ and $R \in L_C^E((n_1, n_2))$. Two relation atoms $R(n_1, n_2)$ and $S(n_3, n_4)$ are adjoined if $\{n_1, n_2\} \cap \{n_3, n_4\} \neq \emptyset$.*

We refer to the set of concept (relation) atoms in $\mathcal{AG}_C$ as CA_C (RA_C).

Note that a CGraph of C can be trivial, i.e. if C has no further structure, then $N_C = \{root_C\}, L_C^N = \{(root_C, \{C\})\}, E_C = \emptyset, L_C^E = \emptyset$. We also use the atom notation to denote graphs, e.g., $C = \{C(root_C)\}$.

3.2 Concept Graph Morphisms and Inherited Atoms

Definition 15. *CGraph Morphism from C into D. Let G_C, G_D be the* augmented *CGraphs for C, D, with $C \neq D$. A CGraph morphism (morphism for short) from C into D, or $\mathcal{G}_C$ into $\mathcal{G}_D$, is a partial function $\mu_{C \rightsquigarrow D} : N_C \rightarrow N_D$ (denoted as μ in case C, D are clear from the context or irrelevant) such that*

1. *$\mu(root_C) = n$ is defined - n will also be called a C* expansion start node.
2. *for all $t \in N_C$ s.t. $\mu(t)$ is defined: if $L_C^N(t) = \mathcal{L}$, then $L_D^N(\mu(t)) = \mathcal{L}'$ with $\mathcal{L} \subseteq \mathcal{L}'$.*
3. *if $\mu(m_k) = n$, $m_k \neq root_C$, $k \geq 1$, then there is a sequence of adjoining relation atoms $\langle R_1(m_0, m_1), \ldots, R_k(m_{k-1}, m_k)\rangle$ in C with $m_0 = root_C$ such that there is a corresponding sequence of adjoining relations atoms in D: $\langle R_1(\mu(m_0), \mu(m_1)), \ldots, R_k(\mu(m_k), n)\rangle$ in D.*

More specifically, the morphism with $\mu_{C \rightsquigarrow D}(root_C) = n$ is denoted as $\mu_{C \rightsquigarrow D|n}$. Note that n is a node in D for which $C \in L_D^N(n)$ holds, or equivalently, $C(n) \in CA_D$, and that $\mu_{C \rightsquigarrow D|n}(root_C) = n$. If n is irrelevant, we only write $\mu_{C \rightsquigarrow D}$.

C is called the domain (or source) concept, and D the range (or target) concept; we also use domain (or source) and range (or target) CGraph.

A morphism $\mu_{C \rightsquigarrow D}$ is more general *than $\mu_{SubC \rightsquigarrow D}$ iff $(SubC, C) \in {\mathcal{O}_\mathcal{C}}^+$.*

Hence, the labels of the nodes in the source CGraph have to be subsets of the labels of the corresponding nodes in the target CGraph which are, hence, more specific (or identical). Note that the analog does *not* hold for the edges. Rather, a morphism *induces* inherited edges in the target CGraph, to be defined below.

We say that a morphism *induces atoms*:

Definition 16. *Induced / Inherited and Local Atoms. Let $\mu_{C \rightsquigarrow D|n}$ be a morphism. Then, the morphism induces the following atoms:*

– *$E(m) \in CA_D$ is induced if there is $\mu(m') = m$ with $E(m') \in CA_C$.*

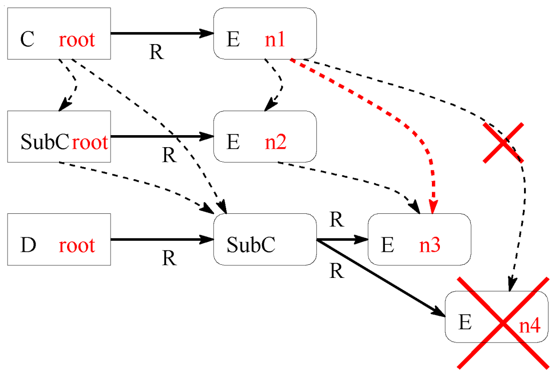

Fig. 2. Illustration of closure and inconsistency.

– *$R(m_1, m_2) \in RA_D$ is induced if there is $\mu(m'_1) = m_1, \mu(m'_2) = m_2$ with $R(m'_1, m'_2) \in RA_C$.*

Induced atoms are also called inherited atoms. *More specifically, we say that an atom is* induced by C into D, *or* inherited from C to D. *The set of induced atoms is denoted by $\psi_{C \rightsquigarrow D|n}$. An atom in D which is not induced is called a* local *atom, more specifically:* local to D.

3.3 Concept Patchworks and Coverings

Obviously, a complex CGraph may have more than one concept atom, and content may be inherited from any of those. Thus, *multiple morphisms* are required to describe all inherited content. For example, if D contains $C(n_1)$ and $C(n_2)$ with $n_1 \neq n_2$, we already require two morphisms: $\mu_{C \rightsquigarrow D|n_1}, \mu_{C \rightsquigarrow D|n_2}$ (μ is a function). A morphisms is hence already associated with the *expansion start node*, here n_1 and n_2. A *CGraph patchwork* describes how *different parts of the concept are "patched" together*:

Definition 17. *CGraph patchwork into D, induced atoms, unified nodes. A* CGraph patchwork (patchwork for short) *into D for a subset of its concept atoms $T \subseteq CA_D \setminus \{D(root_D)\}$ is a set of morphisms into D, $\mathcal{PW}_D \triangleq \{\mu_{E \rightsquigarrow D|n} \mid E(n) \in T\}$.*

The set of induced or inherited atoms *is $\psi(\mathcal{PW}_D) = \bigcup_{\mu \in \mathcal{PW}} \psi_\mu$.*

Let $\mathcal{I}_\alpha = \{\mu \mid \alpha \in \psi_\mu, \mu \in \mathcal{PW}_D\}$ be the set of morphism that induces the atom α. We say that α is inherited by *$\mu_{C \rightsquigarrow D} \in \mathcal{I}_\alpha$ if $\mu_{C \rightsquigarrow D}$ is a (or the)* most general morphism *in $\mathcal{I}_\alpha$. Given $\mathcal{PW}_D$, we hence say that α is* inherited from C.

If $\{\mu_{E_1 \rightsquigarrow D}, \mu_{E_2 \rightsquigarrow D}\} \subseteq \mathcal{PW}_D$ with $\mu_{E_1 \rightsquigarrow D}(n_1) = \mu_{E_2 \rightsquigarrow D}(n_2) = n$, then we say that n_1 and n_2 have been unified (merged, equated) into n, and n_1 and n_2 are said to be co-referential. *Note that possibly $E_1 = E_2$.*

The *closure of a set of morphisms* is defined as some kind of transitive closure over the different morphisms. Consider the concepts C, $SubC$, and D, with $(SubC, C) \in \mathcal{O_C}^+$, and the morphisms $\mu_{C \rightsquigarrow SubC}, \mu_{SubC \rightsquigarrow D}$ with $\mu_{C \rightsquigarrow SubC}(root_C) = root_{SubC}$, hence $\mu_{C \rightsquigarrow SubC|root_{SubC}}$ (note that $C(root_{SubC})$ is a concept atom in $SubC$, because of $(SubC, C) \in \mathcal{O_C}^{+}!$) , and $\mu_{SubC \rightsquigarrow D}(root_{SubC}) = n_{SubC}$, hence $\mu_{SubC \rightsquigarrow D|n_{SubC}}$. Moreover, assume that $\mu_{C \rightsquigarrow SubC|root_{SubC}}(n_1) = n_2$. Then, if also $\mu_{SubC \rightsquigarrow D|n_{SubC}}(n_2) = n_3$ holds, we have to ensure that $\mu_{C \rightsquigarrow D|n_{SubC}}(n_1) = n_3$ holds (note that $C(n_{SubC})$ is a concept atom in D). Suppose to the contrary that also $\mu_{C \rightsquigarrow D|n_{SubC}}(n_1) = n_4$, with $n_3 \neq n_4$. This describes a situation where n_1 was inherited directly and indirectly from C into D, and gets duplicated as n_3, n_4. The situation is depicted in Fig. 2. We hence require that the composition of the morphisms does not violate functionality of the morphisms.

Definition 18. *Closure of $\mathcal{PW}$. The* closure *of a set of morphisms $\mathcal{PW}$, denoted $\mathcal{PW}^{+}$, is defined as follows:*

– $con(\mathcal{PW}) \triangleq \{\mu_{C \rightsquigarrow D|n_{SubC}} \cup \{(n_1, n_3)\} \mid (C, SubC) \in \mathcal{O_C}^{+},$
$\{\mu_{C \rightsquigarrow SubC|root_{SubC}}, \mu_{SubC \rightsquigarrow D|n_{SubC}}\} \subseteq \mathcal{PW},$
$\mu_{C \rightsquigarrow SubC|root_{SubC}}(n_1) = n_2, \mu_{Sub_C \rightsquigarrow D|n_{SubC}}(n_2) = n_3\}$
– $\mathcal{PW}^{+} \triangleq \bigcup_{i \in 1 \ldots \infty} con^{i}(\mathcal{PW})$

(Note that $(D, SubC) \in \mathcal{O_C}^{+}$ is possible, also.) The relations in $\mathcal{PW}^{+}$ are now no longer necessarily functions, since we extended them by means of $\mu_{C \rightsquigarrow D|n_{SubC}} \cup \{(n_1, n_3)\}$, i.e., we added $\mu_{C \rightsquigarrow D|n_{SubC}}(n_1) = n_3$). However, the notion of *consistency of a set of morphisms* requires exactly this:

Definition 19. *Consistency of $\mathcal{PW}$. A set of morphisms $\mathcal{PW}$ is called* consistent *iff, for all C, D, n, every morphism in $\mathcal{PW}^{+}$ satisfies the following:*

1. *functionality: if $\mu_{C \rightsquigarrow D|n}(n_1) = n_2 \in \mathcal{PW}^{+}$ and $\mu_{C \rightsquigarrow D|n}(n_1) = n_3 \in \mathcal{PW}^{+}$, then $n_2 = n_3$.*
2. *no self mapping: for all $C, n, n_1 \neq n_2$, $\mu_{C \rightsquigarrow C|n}(n_1) = n_2 \notin \mathcal{PW}^{+}$.*
3. *no self inheritance: there is no sequence of morphisms $\mu_1, \ldots, \mu_m \in \mathcal{PW}^{+}, m > 1$ such that $(\mu_1 \circ \cdots \circ \mu_m)(n) = n$ (where $\circ$ denotes composition).*

The functionality criterion has already been explained. The second criterion prevents that a CGraph C inherits content to itself, which is a reasonable assumption.[3] The third criterion simply prevents that a node is inherited from itself, directly or indirectly.

Our goal is to identify the inherited content in a GSKB by means of patchworks, for all its concepts. Often, there is more than one possible patchwork for a given concept C, and different patchworks may result in different sets of inherited / induced atoms ψ. We are interested in those patchworks that *globally maximize (for the whole GSKB)* the set of inherited atoms ψ, and among those the *smallest* such patchworks, i.e., those that require as few morphisms as possible. In case an atom can be inherited via two different morphisms such that one is more general than the other, preference should be given to the more general morphism. This captures the intuition that an atom should be inherited from the most general concept that has it. These ideas are formalized in the notion of a *GSKB covering* as follows:

[3] Even if C was cyclical in the sense that $C(n)$ is a concept atom in C, for $n \neq root_C$.

Definition 20. *GSKB Covering. A* covering *of a GSKB K is a union of CGraph patchworks for a (not necessarily true) subset of its concepts $\mathcal{C}' \subseteq \mathcal{C}$: $\mathcal{PW}_K \triangleq \bigcup_{C \in \mathcal{C}'} \mathcal{PW}_C$. We extend the definitions of induced / implied atoms to whole knowledge bases as follows: $\psi(\mathcal{PW}_K) \triangleq \bigcup_{\mathcal{PW}_C \in \mathcal{PW}_K} \{(C, \alpha) \mid \alpha \in \psi_{\mathcal{C}}(\mathcal{PW}_C)\}$. We then require that $\mathcal{PW}_K$ a* minimal set (w.r.t. set inclusion) *that satisfies the following three* principles:

1. *consistency: $\mathcal{PW}_K$ is consistent.*
2. *ψ-maximality: there is no GSKB covering $\mathcal{PW}'_K$ of the same or smaller cardinality as $\mathcal{PW}_K$ with $\psi(\mathcal{PW}_K) \subset \psi(\mathcal{PW}'_K)$.*
3. *μ-generality: if $\alpha \in \psi_{C \rightsquigarrow D} \in \mathcal{PW}_K$ for some C, D, and there exists a more general morphisms $\psi_{C' \rightsquigarrow D}$ with $\alpha \in \psi_{C' \rightsquigarrow D}$ with $(C, C') \in \mathcal{O}_{\mathcal{C}}{}^+$, such that its addition to $\mathcal{PW}_K$ would retain consistency of $\mathcal{PW}_K$, then $\psi_{C' \rightsquigarrow D} \in \mathcal{PW}_K$.*

In general, there can be more than one covering for a given GSKB. An example is given by the cyclical GSKB with concepts C, D, such that $C = \{C(root_C), R(root_C, n_1), D(n_1)\}$, and $D = \{D(root_D), R^{-1}(root_D, n_2), C(n_2)\}$. Here, we can either have a morphisms $\mu^1_{C \rightsquigarrow D|n_2}(root_C) = n_2, \mu^1_{C \rightsquigarrow D|n_2}(n_1) = root_D$, or a morphisms $\mu^2_{D \rightsquigarrow C|n_1}(root_C) = n_2, \mu^2_{D \rightsquigarrow C|n_1}(n_2) = root_C$, but not both. Hence, a covering forces that one of the two relation atoms in C, D is local, and the other one is inherited. If both were inherited, then point point 3 in Def. 19 would be violated. If both were local, then the principle of ψ-maximality would be violated. It is left unspecified which of those two coverings is constructed by the algorithm given below. Suppose we chose $\mu^1_{C \rightsquigarrow D}$ for $\mathcal{PW}_K$. Note that we cannot even add $\mu^3_{D \rightsquigarrow C} = \{(root_D, n_1)$ to $\mathcal{PW}_K$, as this would result in $(\mu^1_{C \rightsquigarrow D} \circ \mu^3_{D \rightsquigarrow C})(n_1) = n_1$ and $(\mu^3_{D \rightsquigarrow C} \circ \mu^1_{C \rightsquigarrow D})(root_D) = root_D$, hence $D(root_D)$ in D and $D(n_1)$ in C would become self-inherited, hence violating consistency again.

The principle of μ-generality is explained as follows. Consider the CGraphs C, C', C'', such that $\{(C', C), (C'', C')\} \subseteq \mathcal{O}_{\mathcal{C}}$, with $C = \{C(root_C), R(root_C, n_1), D(n_1)\}$, $C' = \{C'(root_{C'}, R(root_{C'}, n_2), D(n_2), R(n_2, n_3), E(n_3)\}$ and $C'' = \{C''(root_{C''}, R(root_{C''}, n_4), D(n_4), R(n_4, n_5), E(n_5), R(n_5, n_6), F(n_6)\}$. Looking at C'', the atoms $R(root_{C''}, n_4), D(n_4), R(n_4, n_5), E(n_5)$ can all be induced by C' into C'' via the obvious morphism.[4] However, $R(root_{C''}, n_4), D(n_4)$ can already be induced by C into C', and this is the more general morphism.[5] We hence require that this morphism is also present: $\mu_{C \rightsquigarrow C''} \in \mathcal{PW}_K$, since those atoms are induced by both morphisms. Also, we need $\mu_{C' \rightsquigarrow C''} \in \mathcal{PW}_K$ as otherwise, we would not be able to inherit $R(n_4, n_5), E(n_5)$ from C' to C'', hence violating ψ-maximality. The desired covering will hence consider the following atoms as local: all atoms in C, the atoms $R(n_2, n_3), E(n_3)$ in C', and the atoms $R(n_5, n_6), F(n_6)$.

3.4 Computation of a Covering

Our goal is to compute a GSKB covering, and it is clear that this is a hard problem:

[4] Be means of $\mu_{C' \rightsquigarrow C''}(root_{C'}) = root_{C''}, \mu_{C' \rightsquigarrow C''}(n_2) = n_4, \mu_{C' \rightsquigarrow C''}(n_3) = n_5$.
[5] By means of $\mu_{C \rightsquigarrow C''}(root_C) = root_{C''}, \mu_{C \rightsquigarrow C''}(n_1) = n_4$.

Proposition 4. *Computing a Covering is NP-Hard and in PSPACE. It is clear that we can compute a covering of K in a simple non-deterministic "guess and validate" style – we guess the morphisms, and check whether they satisfy the conditions Def. 19 and Def. 20. However, storing the morphisms may require polynomial space. It is also clear that the problem is NP-hard, by reduction from the clique problem: let G be the input graph to the clique problem, and the CGraph of $SupG$ be a k-clique, and $(G, SupG) \in \mathcal{O_C}^+$. Assume $L_{SupG}(n) = \emptyset$ for all $n \in N_{SupG}$, $n \neq root_{SupG}$, and $L_{SupG}(n) = \emptyset$ for all $n \in N_G$ for $n \neq root_G$, and $L_G(root_G) = \{G, SupG\}, L_{SupG}(root_{SupG}) = \{SupG\}$, with $\mathcal{C} = \{SupG, G\}$. Then, G contains a clique of size k iff, for its covering $\mathcal{PW}_K$: $\psi_G(\mathcal{PW}_K) = CA_G \cup RA_G \setminus \{SupG(root_G)\}$ (i.e., all relation and concept atoms with the exception of the root concept atom are inherited), and all morphisms are injective.*

The deterministic version of the sketched nondeterministic algorithm requires search. The basic ideas for a deterministic version of the algorithm are outlined as follows.

Since the GSKB can contain cyclical references, we must prevent the construction of morphisms which induce cycles in order to satisfy the requirements of Def. 19, namely points 2 and 3. In principle, we could compute the patchworks for the concepts in any order of sequence. When the patchwork for concept D is constructed, we are establishing maximal morphisms to all concepts mentioned in concept atoms in D, and we make sure that we do not violate consistency, ensuring ψ-maximality. Moreover, for every concept atom $C(n)$ in D, if $(C, SupC) \in \mathcal{O_C}^+$, then also $SupC(n)$ in D, and hence, we will not only establish $\mu_{C \rightsquigarrow D}$, but also $\mu_{SupC \rightsquigarrow D}$, hence satisfying μ-generality.

Unfortunately, checking the conditions in Def. 19 to ensure consistency in Def. 20 can be costly, and we hence propose a slightly optimized version which works in *two phases* by exploiting ideas akin to topological sorting w.r.t. a $refers\_to^+$ relation:

Definition 21. *Cycles, Refers-To Relation, Refers-To Violation. A concept C* refers to *concept D, iff $D(n) \in CA_C$: $refers\_to(C, D)$. A concept C is cyclical iff $refers\_to^+(C, C)$. A GSKB K is cyclical if it contains a cyclical concept. Given two concepts C_i, C_j with $1 \leq i < j \leq m$ in a sequence of concepts $\langle C_1, \ldots, C_m \rangle$, we speak of a refers-to violation iff $refers\_to(C_i, C_j)$ holds. The number of refers-to violations for the sequence is $ref\_vio(\langle C_1, \ldots, C_m \rangle) \triangleq \Sigma_{1 \leq i \leq m} |dom(refers\_to^+(C_i)) \cap \{C_{i+1}, \ldots, C_m\}|$. An* optimal concept processing sequence *is a sequence $seq = \langle C_1, \ldots, C_m \rangle$, $\mathcal{C} = \{C_1, \ldots, C_m\}$, which minimizes the number of $refers\_to$ violations: $\xi = argmin(ref\_vio(seq), seq \in \pi(\mathcal{C}))$; π denotes all possible sequences / permutations of $\mathcal{C}$.*

In case $\xi = 0$, we can simply topologically sort the GSKB w.r.t. the $refers\_to^+$ relation and process in that order of sequence. For such GSKB we only need to check point 1 in Def. 19. And, even with $\xi > 0$, we can skip the checks for point 2 and 3 in Def. 19 if we organize the computation of the covering in *two phases:* In *phase one*, after having computed an optimal sequence, to construct a patchwork for concept C_k we consider only morphisms from concepts C_i into C_k with $i < k$ w.r.t. that ordering, hence, C_i was processed before. We make sure that patchworks do not violate point 1 in Def. 19. The result of phase 1 is a covering in case $\xi = 0$. In case $\xi > 0$, ψ-maximality

```
**Input** : A GSKB $K = (\mathcal{C}, \mathcal{R}, \mathcal{O}_\mathcal{C}, \mathcal{O}_\mathcal{R}, \mathcal{I}_\mathcal{R}, \mathcal{G})$
**Output**: $\mathcal{PW}_K$: a covering of $K$
$\mathcal{PW}_K \leftarrow \emptyset$ ;
$\langle C_1, \ldots, C_n \rangle \leftarrow argmin(ref\_vio(seq), seq \in \pi(\mathcal{C}))$ ;
$\mathcal{S} \leftarrow \emptyset$   % concepts to reconsider in Phase 2 ;
$\mathcal{P} \leftarrow \emptyset$   % already processed concepts ;
% Phase 1 ;
for $i \leftarrow 1$ to $n$ do
    if $dom(refers\_to^+(C_i)) \cap \{C_{i+1}, \ldots, C_n\} \neq \emptyset$ then
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{C_i\}$ ;
    end
    $\mathcal{PW}_K \leftarrow \mathcal{PW}_K \cup compute\_patchwork(C_i, \mathcal{PW}_K, \mathcal{P}, 1)$ ;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{C_i\}$ ;
end
% Phase 2 ;
for $C \in \mathcal{S}$ do
    $\mathcal{PW}_K \leftarrow \mathcal{PW}_K \setminus \{\mu_{D \rightsquigarrow C} \mid \mu_{D \rightsquigarrow C} \in \mathcal{PW}_K\}$ ;
    $\mathcal{PW}_K \leftarrow \mathcal{PW}_K \cup compute\_patchwork(C, \mathcal{PW}_K, \mathcal{P}, 2)$ ;
end
return $\mathcal{PW}_K$
```

Algorithm 1: $cover(k)$ – The GSKB Covering Algorithm

is not yet satisfied. There is (at least) a concept C_k with $refers\_to^+(C_k, C_l)$ and $l > k$. Hence, C_k may contain inherited content from C_l, but $\mu_{C_l \rightsquigarrow C_k}$ was not established. Those concepts C_k are identified and collected in phase 1, and simply re-processed again in *phase 2*, by re-computing their patchworks, leading to $\mu_{C_l \rightsquigarrow C_k}$. During morphisms construction, we now have to checks all points in Def. 19, which is more costly. However, the number of concepts to re-consider in phase 2 is usually much smaller than $|\mathcal{C}|$. This is Algorithm 1, which uses Algorithm 2 for patchwork computation.

The function $most\_specific\_atom(CA)$ is non-deterministic and chooses a concept atom $D(n) \in CA$ for which there is no concept atom $SubD(n) \in CA$ with $(SubD, D) \in \mathcal{O}_\mathcal{C}{}^+$. Within $compute\_patchwork(C, \mathcal{PW}, \mathcal{P}_K, phase)$, we make sure to process the concept atoms in CA_C in *most specific first* order. This has the advantage that the closures can be more easily computed: at the time when $\mu_{SupC \rightsquigarrow SubC}$ is computed, the morphisms $\mu_{SupC \rightsquigarrow C}$ and $\mu_{C \rightsquigarrow SubC}$ are already available.

The function $max\_consistent\_morphism(C, D(n), \mathcal{PW}_K \cup \mathcal{PW}_C, phase)$ finds the maximal consistent morphism from D into C. In case $phase = 1$, it only checks point 1 in Def. 19 in the consistency check, otherwise all points in Def. 19. It tries to compute a maximal consistent morphism which induces as many atoms as possible into C. In case no consistent mapping can be found, the empty set is returned. In order to check consistency, $max\_consistent\_morphism$ has to compute the closure of the given morphism $\mathcal{PW}_K \cup \mathcal{PW}_C$. If a maximally consistent morphism $\mu_{D \rightsquigarrow C|n}$ can be found, it is returned together with the additional morphisms resulting from the closure computation: $closure(\mathcal{PW}_K \cup \mathcal{PW}_C \cup \{\mu_{D \rightsquigarrow C|n}\}) \setminus \mathcal{PW}_K$. This function is not further detailed here, but its implementation described below.

```
Input  : A concept, a set of morphisms, a set of processed concepts, a $phase \in \{1, 2\}$:
         $(C, \mathcal{PW}, \mathcal{P}_K$,phase)
Output: A $C$-patchwork $\mathcal{PW}_C$
$\mathcal{S} \leftarrow \{D(n) \mid D \in \mathcal{P}, D(n) \in CA_C\}$ ;
$\mathcal{PW}_C \leftarrow \emptyset$ ;
while $\mathcal{S}$ do
    $D(n) \leftarrow most\_specific\_atom(\mathcal{S})$ ;
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{D(n)\}$ ;
    $(\mu_{D \rightsquigarrow C|n}, \mathcal{P}^+) \leftarrow max\_consist.\_morphism(C, D(n), \mathcal{PW}_K \cup \mathcal{PW}_C, phase)$ ;
    $\mathcal{PW}_C \leftarrow \mathcal{PW}_C \cup \{\mu_{D \rightsquigarrow C|n}\} \cup \mathcal{P}^+$ ;
end
return $\mathcal{PW}_C$
```

Algorithm 2: $compute\_patchwork(C, \mathcal{PW}, \mathcal{P}_K, phase)$ – Helper Function

3.5 The Implementation

The so-far presented algorithm is an idealized version of the actual implemented algorithm. During our implementation efforts, we learned that the idealized algorithm is not able to compute a covering for the AURA KB in a reasonable amount of time. Scalability is of concern here. In the AURA KB, we have 695 concepts with more than 20 nodes, on average 50.1 nodes and 104.5 edges. The average number of concept atoms in those concepts is 104.7. That means we have to consider potentially 104.7×695 graph morphisms. If runtime is exponential in the number of nodes, this results in 8.78×10^{19} node mappings to consider if a naive approach is followed, all of which have to be checked for consistency, etc. A lot of effort has been put into implementing a more scalable version of $compute\_patchwork$.

In the following we describe the main *implementation techniques that enabled scalability*; we think that it might be insightful to document and preserve those techniques for researchers working on similar problems.

First, there is no computation of an optimal sequence – rather, the optimal sequence is constructed iteratively / incrementally during phase 1.

More importantly, instead of finding a maximal morphism for each $D(n) \in CA_C$ at a time, the algorithm constructs and patches morphisms together in a piecewise fashion. It is agenda-driven – the agenda consists of the currently non-induced edges of C. Such an edge is called *open*. In a loop, an open edge $R(n_1, n_2)$ is selected from the agenda, and it is then searched for a morphism which induces it. Whenever a node mapping is established, we are checking the consistency conditions as required, and prevent cycles, etc. When the agenda is empty, i.e., there are either no more open edges or no more additional morphisms can be found, then a solution was found. The quality of the computed solution / patchwork is determined by the number of its induced atoms – *the score of the solution.*

The implemented algorithm produces perfect coverings if given indefinite time. Otherwise, the quality of the approximation increase the more time it is given (there is a timeout switch, see below). After a solution has been found, the algorithm can continue the search for additional solutions with higher scores. Only the best solutions are kept. During this search for better solutions, the score of the best so-far found solution is used to reduce the search space – sometimes work on a partial solution can be stopped if it can be determined that the full solution based on it cannot outperform the best so-far found solutions.

We can parameterize the algorithm such that the search in $compute\_patchwork$ terminates as soon as a total number of m_1 solutions has been found, or if we were able to improve the best-so-far solution m_2 times, or if a timeout after t seconds has been reached. In case the timeout occurred before a (suboptimal) patchwork was found, we can either continue the search until at least one patchwork is available, or accept the incomplete partial patchwork (hence, all remaining non-induced atoms as local).

We have three **strategies a), b), and c)** for finding a morphism for $R(n_1, n_2)$:

Strategy a) We can often find a morphism starting from $n = n_1$ or $n = n_2$, for some concept atom $D(n)$. In case there is more than one such $D(n)$, we are trying the *most specific concept atoms first*. We inspect the CGraph of D and try to inherit a path of edges, starting from the $root_D$, hence producing a morphism $\mu_{D \rightsquigarrow C|n}$. Each inherited path and hence morphism μ is associated with a *score* which is the product $|dom(\mu)| \times (penalty + 1)$. The $penalty$ is the number of node and edge label specializations required over D. Ideally, we are looking for a perfect match with penalty 0 and maximal size. However, since this “path harvesting” already requires graph search in D in order to find the inherited path, the max. path length is constrained to a small number, say 3. The inherited edges are now no longer open.

Note that an edge can be induced by more than one morphism, and every $D(n)$ will eventually be considered if it has to. E.g., consider $C = \{C(root_C), R(root_C, n_1), S(n_1, n_2)\}$, $C' = \{C'(root_{C'}), R(root_{C'}, n_3), T(n_3, n_4)\}$, and $C'' = \{C''(root_{C''}), R(root_{C''}, n_5), S(n_5, n_6), T(n_5, n_7)\}$, with $\{(C', C), (C'', C')\} \subseteq \mathcal{O}_{\mathcal{C}}$. Here, $\{C''(root_{C''}), C'(root_{C''}), C(root_{C''})\} \subseteq CA_{C''}$. To induce $S(n_5, n_6)$ we require $\mu_{C \rightsquigarrow C''|root_{C''}}$, and to induce $T(n_5, n_7)$ we require $\mu_{C \rightsquigarrow C'|root_{C''}}$. Note that $R(root_{C''}, n_5)$ is induced by both morphisms.

Strategy b) Here it is checked whether $R(n_1, n_2)$ can be induced by *extending* an existing morphism; this is useful because Strategy a) has a length cut-off, as described. If we find that for $R(n_1, n_2)$ there is an edge $S(n_1, n_3)$ which is already induced by a morphism $\mu_{D \rightsquigarrow C|n}$, we then restart the search for n in C by looking at D again. In case there is more than one such morphism, we select the morphism with the highest score. We skip forward to the node n'_1 with $\mu_{D \rightsquigarrow C|n}(n'_1) = n_1$ and try to extend the morphism. This way, we can inherit arbitrary graph structures from D, or simply extend the path to become longer, by patching together and extending partial morphisms for *one inherited path at a time.* Restarting has the advantage that Strategy a) does not invest a lot of time constructing longer and longer paths in a depth-first fashion with increasing penalties, hence the max. length is constrained to a small value. By restarting with the partial morphism which have the highest score, we achieve a kind of best-first search effect (similar to beam search), because the control is given back quickly to the agenda, so search can re-focus quickly on the most promising partial morphisms to extend.

For example, consider $C = \{C(root_C), R(root_C, n_1), S(root_C, n_2), T(n_1, n_2)\}$ and isomorphic $C' = \{C'(root_{C'}), R(root_{C'}, n_3), S(root_{C'}, n_4), T(n_3, n_4)\}$ with

$(C', C) \in \mathcal{O}_{\mathcal{C}}$. We can first find $\mu_{C \rightsquigarrow C'}$ which induces $\{C'(root_{C'}), R(root_{C'}, n_3), S(root_{C'}, n_4)\}$, and then restart and extend such that $\{T(n_1, n_2)\}$ is induced by $\mu_{C \rightsquigarrow C'}$.

Strategy c) If Strategies a) and b) fail for an edge edge $R(n_1, n_2)$, then, for it to be induced, it must be induced by a morphism which already maps to some node n with $E(n) \in CA_C$ such that n is connected to n_1, n_2, but n has not been considered yet as an expansion start node. To find such a morphism, we consider all morphisms that induce $E(n)$ such that n is connected to n_1, n_2 , and start the search in E at $root_E$ such that $\mu_{E \rightsquigarrow C|n}(root_E) = n$ holds.

For example, consider $C = \{C(root_C), R(root_C, n_1), S(root_C, n_2), T(n_1, n_2)\}$, $C' = \{C'(root_{C'}), R(root_{C'}, n_3), S(root_{C'}, n_4), T(n_3, n_4), U(n_4, n_5), D(n_3)\}$, and $D = \{D(root_D), T(root_D, n_6), U(n_6, n_7)\}$ with $(C', C) \in \mathcal{O}_{\mathcal{C}}$. The edge $U(n_4, n_5)$ in C' can obviously not be induced by the morphism $\mu_{C \rightsquigarrow C'|root_{C'}}$. Rather, we need to establish $\mu_{D \rightsquigarrow C'|n_3} = \{(root_D, n_3), (n_6, n_4), (n_7, n_5)\}$. Note that the edge $T(n_3, n_4)$ is induced by $\mu_{C \rightsquigarrow C'|root_{C'}}$ as well as by $\mu_{D \rightsquigarrow C'|n_3}$. It is interesting to note that this edge is hence *inherited from both C and D,* as both are incomparable w.r.t. ${\mathcal{O}_{\mathcal{C}}}^{+}$.

4 Evaluation

We have applied the GSKB covering algorithm to the AURA GSKB, and an approximate covering with sufficient quality was computed in 18 hours. We used a timeout of 3 minutes, and after the timeout the algorithm was allowed to produce at least one solution (hence $t = 300$, and $m_1 = 50$, and $m_2 = 3$). Only 1431 concepts (22 % of the concepts) had to be re-processed in phase 2, and 20 concepts timed out – three concept required 34, 19, and 15 minutes, resp. It should be noted that the algorithm can compute approximate coverings of lower quality in much less than 18 hours, e.g. in about 2 to 3 hours.

The following table shows the stats of the covering produced by the 18 hours run, for concepts which have at least $\{0, 20, 50, 100, 200\}$ nodes ($|N_C| \geq$), and the number and size of their morphisms, as well as the percentages of inherited concept and relation atoms. It can be seen that the algorithm has performed a non-trivial task. For $|N_C| \geq 50$, 5690 morphism have been computed, with an average size of $|dom(\mu)| = 7.36$ nodes. If we look at the biggest concept with 461 nodes, we find that the algorithm has created a patchwork with 126 morphisms, with an avg. size of 9.49 nodes. Altogether, the algorithm has established $14,146$ morphisms of average size 5.47, and identified 57 % of the concept atoms and 69 % of the relation atoms as inherited. We are also able to tell from *where* an atom is inherited from, see Def. 17. This information can be called the *provenance* of atoms and is of great importance to the modelers of the KB [4].

$\|N_C\|$	≥ 0	≥ 20	≥ 50	≥ 100	≥ 200
concepts (#)	6430	695	224	57	3
avg. $\|N_C\|$ (# nodes)	8.2	50.1	88.46	139.52	342
avg. $\|E_C\|$ (# edges)	14.5	104.5	198.22	328.29	866.33
avg. $\|CA_C\|$ (# atoms)	26.9	104.68	158.125	238.23	546
% inherited $\|RA_C\|$	69	75.6	77.4	77	71.6
% inherited $\|CA_C\|$	57	71.1	74	74.6	74.9
$\|\mathcal{PW}_K\|$ (# morphisms)	14146	9900	5690	2264	287
avg. $\|dom(\mu)\|$ (# nodes)	5.47	6.71	7.36	7.73	8.25
avg. $\|\mathcal{PW}_C\|$ (# morphisms)	2.2	14.24	25.4	39.71	95.67

5 Related Work

As stated in the introduction, the co-reference problem has been studied to some extent in the natural language processing literature under the the term anaphora resolution; for example, [5, 6] use default rules to hypothesize equality assertions between referents in order to guess and establish co-references.

The reasoning system *Knowledge Machine (KM)*, [9], uses a so-called unification mechanism, Umap, to strengthen the GSKB by establishing co-references. Unfortunately, the lack of a formal semantics makes it very difficult to understand. A major problem in KM is that unifications are not reversible, since they are not represented explicitly as (equality or Umap) atoms in the KM GSKB. Instead, unifications are performed by destructive substitutions of constants in the GSKB. Retraction and comprehension of Umap unifications can be very difficult and time consuming, as unification is heuristic in nature and frequently goes wrong.

KM provides a function `get-supports` for computing provenance of every triple in a GSKB. The function returns the concept from where the triple gets inherited. This function relies on KM's so-called *explanation database*, which became corrupted during the AURA project, due to software errors and a changing semantics, hence forcing us to recompute the provenance information. This was the main motivation for the work described in this paper. It turned out that the recomputed provenance information was quite accurate, as confirmed by the experts in our knowledge factory [4].

The work of [10] uses answer set programming (ASP) to formalize KM's Umap operator. The GSKB is specified as an ASP program, together with an axiomatic system of ASP inference rules. These rules capture the semantics of object-oriented knowledge bases, including inheritance, and formalize the UMap operator. The semantics is given by the semantics of the ASP rules, whereas our approach starts with a notion of desirable models and is hence more model-theoretic in nature. Moreover, constants are distinct by definition in ASP programs, so equality (UMap) needs to be modeled on the meta-level. Moreover, the approach has not yet been applied successfully to the full-scale AURA GSKB, so scalability of the approach is still open.

Considering our graph-based approach, we note that graph morphism-based approaches were employed since the early days of KL-ONE [11], to decide subsumption between concepts in description logics. Those approaches are called *structural subsumption algorithms* nowadays [1]. Note that we do not decide subsumption between concepts, as the taxonomy is considered given and fixed here. Instead, we determine for a (possibly singleton) set of atoms from a concept from where they got inherited, or if they are local to that concept, and hence non-redundant. Determining from where an atom is inherited was called provenance computation. Nevertheless, we suspect that our algorithm could be turned into a structural subsumption checker for certain description logics, similar as in [12] for $\mathcal{EL}$. So-called *simulations* are employed in [13] to decide concept subsumption in $\mathcal{EL}^{++}$, which are similar to morphisms. But note that, unlike $\mathcal{EL}$, we are supporting graph-based descriptions, and this is a key requirement in AURA.

Morphisms are also considered in [14] for simple conceptual graphs for deciding "projection" (a form of subsumption). However, no implementation is described there, and we are using different morphisms, as we do not require that *all* relation atoms in C have to be projected into D by a morphism $\mu_{C \rightsquigarrow D}$.

The graph-based algorithm was very successful in the AURA project. We were not able to achieve similar results with any other form of formal reasoning, i.e., description logic reasoners. As argued, in order to compute provenance of atoms / triples, one needs a form of hypothetical, unsound reasoning which requires guessing of co-references / equalities. Considering the size of the AURA GSKB and the complexity of the problem, we consider our algorithm a success story.

It is well-known that the modeling of graph structures is challenging in description logic (DL), as derivations from the tree-model property usually result in decidability problems [15] which can often only be regained by imposing severe artificial modeling restrictions. Although some progress has been made on modeling graph structures with DLs [16], those extensions are still too restricted to be useful here. Our experience is that graph structures are central to biology, and approximating them by trees results in coarse models. Our framework allows us to express the graph structures truthfully, but comes with other restrictions, too. To the best of our knowledge, there is no body of work in the DL community that provides answers to the problems addressed in this paper, and we are not aware of any abduction or hypothesization algorithm which has ever successfully been applied to a GSKB of this size.

We also claim that the algorithm and its implementation techniques can be applied in related areas, for example, for ontology alignment tasks in graph-structured lightweight ontologies, for applications in computational biochemistry (identification of chemical substructures), in text understanding, and in the semantic web (e.g., identification of substructures in RDFS triple stores). This is not surprising, since the MSC problem is a very general one, which shows up in many disguises in many application contexts.

6 Conclusions and Outlook

We showed how to identify inherited content in graph-structured knowledge bases, and did this from two perspectives. From a logical perspective, we argued that accurate provenance of atoms requires identification of / the proper solution of the co-reference problem. We demonstrated that inheritance structures can be captured by means of Skolem function and equality atoms. We described a so-called KB-strengthening algorithm which guesses / hypothesizes co-references between Skolem function terms in order to maximize the inherited content in concept graphs.

For the actual implementation, we employed graph-based notions and demonstrated how inherited content can be described by virtue of graph morphisms. The implemented algorithm was successfully applied to the large-scale AURA KB.

The AURA system and the actual implementation of the algorithm covers additional expressive means that we have not formally reconstructed yet, i.e., transitive and functional relations, number restrictions, and disjointness axioms. The logical formalization of these expressive means is future work, but we are optimistic that it can be done. Especially in the case of cyclical GSKBs, it is not clear yet how to apply a similar line of argumentation, as the Herbrand models are no longer necessarily finite. However, it is in principle clear how to handle disjointness, functionality etc. from a graph-based point of view.

From the morphisms computed by the graph-based algorithm we can compute a strengthened GSKB. The established equality axioms between the Skolem function terms describe the correct inheritance structures, and the quality of the mappings got testified by the subject matter experts in our knowledge factory.

The strengthened GSKB in first-order logic is also the basis for a couple of AURA knowledge base exports in SILK [17], answer-set programming syntax [18], and TPTP FOF syntax [19]. We also have a set of OWL2 [20, 21] exports [22], but the OWL2 KBs are underspecified in the sense that we cannot use Skolem functions here and hence, no equality atoms can be used to establish co-references, so those KBs are approximations. This suite of exported KBs is called the Bio_KB_101 suite, and is made available to the public under a *Creative Commons License*, and can be found here for download [23], together with a detailed description of the various variants of the exports.

In future work, we need to establish a closer correspondence between the logical and the graph-based formalizations. It should be relatively straight forward to show that the graph-based algorithm is sound and complete w.r.t. the given logical semantics.

Acknowledgments: This work was funded by Vulcan Inc. under Project Halo

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press (2003)
2. Gunning, D. and Chaudhri, V. K. et al.: Project Halo Update - Progress Toward Digital Aristotle. AI Magazine (2010)
3. Reece, J.B., Urry, L.A., Cain, M.L., Wasserman, S.A., Minorsky, P.V., Jackson, R.B.: Campbell Biology, 9th ed. Harlow: Pearson Education (2011)
4. Chaudhri, V. and Dinesh, N., et al: Preliminary Steps Towards a Knowledge Factory Process. In: The Sixth International Conference on Knowledge Capture. (2011)
5. Carpenter, B.: Skeptical and Credulous Default Unification with Applications to Templates and Inheritance. In: Inheritance, Defaults and the Lexicon, Cambridge University Press (1994) 13–37
6. Cohen, A.: Anaphora Resolution as Equality by Default. In Branco, A.H., ed.: Anaphora: Analysis, Algorithms and Applications, 6th Discourse Anaphora and Anaphor Resolution Colloquium, DAARC 2007, Lagos, Portugal, March 29-30, 2007. Selected Papers. Volume 4410 of Lecture Notes in Computer Science., Springer (2007) 44–58
7. Overholtzer, A., Spaulding, A., Chaudhri, V.K., Gunning, D.: Inquire: An Intelligent Textbook. In: Proceedings of the Video Track of AAAI Conference on Artificial Intelligence, AAAI (2012) See `http://www.aaaivideos.org/2012/inquire_intelligent_textbook/`.
8. Hedman, S.: A First Course in Logic: An Introduction to Model Theory, Proof Theory, Computability, and Complexity. Oxford Texts in Logic (2004)
9. Clark, P., Porter, B.: Building Concept Representations from Reusable Components. In: Proceedings of AAAI, AAAI Press (1997)
10. Chaudhri, V.K., Tran, S.C.: Specifying and Reasoning with Under-Specified Knowledge Base. In: International Conference on Knowledge Representation and Reasoning. (2012)

11. Brachman, R.J., Levesque, H.J.: The Tractability of Subsumption in Frame-Based Description Languages. In: AAAI, AAAI Press (1984)
12. Haarslev, V., Möller, R.: The Revival of Structural Subsumption in Tableau-Based Description Logic Reasoners. In: International Workshop on Description Logics (DL'08). (2008)
13. Baader, F.: Restricted Role-value-maps in a Description Logic with Existential Restrictions and Terminological Cycles. In: International Workshop on Description Logics (DL'03). (2003)
14. Chein, M., Mugnier, M., Simonet, G.: Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In: International Conference on Knowledge Representation (KR'98, Morgan Kaufmann (1998)
15. Vardi, M.Y.: Why is Modal Logic so Robustly Decidable? Descriptive Complexity and Finite Models, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **31** (1996) 149–184
16. Motik, B., Cuenca Grau, B., Horrocks, I., Sattler, U.: Representing Ontologies Using Description Logics, Description Graphs, and Rules. Artif. Intell. **173** (2009) 1275–1309
17. Grosof, B.: The SILK Project: Semantic Inferencing on Large Knowledge (2012) See `http://silk.semwebcentral.org/`.
18. Chaudhri, V.K., Heymans, S., Wessel, M., Son, T.C.: Query Answering in Object Oriented Knowledge Bases in Logic Programming: Description and Challenge for ASP. In: Proc. of 6th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP), Istanbul, Turkey (2013)
19. Chaudhri, V.K., Wessel, M.A., Heymans, S.: KB_Bio_101 : A Challenge for TPTP First-Order Reasoners. In: In KInAR - Knowledge Intensive Automated Reasoning Workshop at CADE-24, the 24th International Conference on Automated Deduction, Lake Placid, New York, USA (2013)
20. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-overview/`.
21. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: International Conference on Knowledge Representation and Reasoning. (2006)
22. Chaudhri, V.K., Wessel, M.A., Heymans, S.: KB_Bio_101: A Challenge for OWL Reasoners. In: Proc. of 2nd OWL Reasoner Evaluation Workshop (ORE 2013), Ulm, Germany (2013)
23. Chaudhri, V.K., Heymans, S., Wessel, M.A.: The Bio_KB_101 Download Page (2012) See `http://www.ai.sri.com/halo/halobook2010/exported-kb/biokb.html`.