Reasoning about Sets via Atomic Decomposition

Hans Jürgen Ohlbach
Imperial College
Department of Computing
London SW7 2BZ
England
h.ohlbach@doc.ic.ac.uk

Jana Koehler
Institut fuer Informatik
Albert-Ludwigs-Universitaet
Am Flughafen 17
79110 Freiburg, Germany
koehler@informatik.uni-freiburg.de

Abstract

We introduce a new technique that translates cardinality information about *finite* sets into simple arithmetic terms thereby enabling a system to reason about such set cardinalities by solving arithmetic equation problems.

The *atomic decomposition technique* separates a collection of sets into smallest mutually disjoint components ("atoms") such that the cardinality of the sets are just the sum of the cardinalities of their atoms.

With this idea, it is possible to have languages combining arithmetic formulas with set terms and to translate the formulas of this combined logic into pure arithmetical formulas.

As a particular application, we show how this technique yields new inference procedures for concept languages with so-called number restriction operators.

Contents

1 Introduction **1**

2 Atomic Decomposition of Sets **3**
2.1 Set Hierarchies . . . 6

3 Arithmetic Reasoning with Decomposed Set-Terms **14**

4 Concept Languages **17**
4.1 The PCM-Example . . . 18
4.2 The Language $\mathcal{TF}^{++}$ . . . 19
4.3 Optimized Decomposition of Role Hierarchies . . . 22
4.4 Reasoning with Concept Terms . . . 25
4.4.1 A Normal-Form and the Consistency Check . . . 26
4.4.2 The Subsumption Test . . . 30

5 Summary and Outlook **32**

6 Appendix: The PCM Example **34**

1 Introduction

If Henry has two sons and three daughters, everybody knows immediately that he has five children. In any of the known general symbolic knowledge representation languages, predicate logic, concept logics etc., it is sometimes difficult to represent this information, let alone to make this simple derivation. In predicate logic it is not even a valid inference without further information. One must tell the system that sons and daughters are in fact children, that the sets are disjoint, i.e. there are no hermaphrodites, and that besides sons and daughters there are no other types of children. In most of the concept languages this information cannot even be expressed. In fact, simple arithmetical reasoning about cardinality of sets still seems to be a big problem in logical contexts.

In this paper, we present a technique for turning cardinality information into arithmetical terms which can be handled by arithmetical equation solvers. The basic idea is very simple and can be explained with the example above. In this example we want to show

$$|sons| = 2 \wedge |daughters| = 3 \Rightarrow |children| = 5 \tag{1}$$

where $|\ldots|$ denotes the set cardinality function. Without further information, sons, daughters and children can be arbitrary sets having arbitrary overlaps with each other. Therefore Implication (1) is not valid in general.

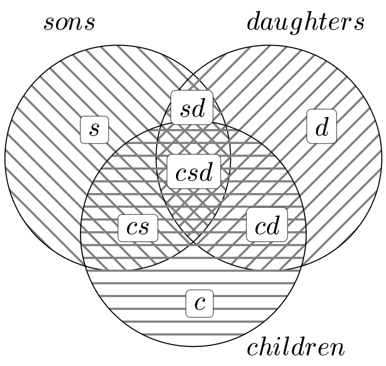


Figure 1: A general set structure

Figure 1 shows the most general way, three different sets can overlap with each other. As one can see, the three sets can be built up from seven mutually disjoint and unseparated areas. We gave these areas names with the following meaning:

c = children, not sons, not daughters.
s = sons, not children, not daughters.
d = daughters, not children, not sons.
cs = children, which are sons, not daughters.
cd = children, which are daughters, not sons.
sd = sons, which are daughters, not children.
csd = children, which are both sons and daughters.

The original sets can now be obtained from their "atomic" components:

$$\begin{aligned} children &= c \cup cs \cup cd \cup csd \\ sons &= s \cup cs \cup sd \cup csd \\ daughters &= d \cup cd \cup sd \cup csd \end{aligned}$$

Moreover, since this decomposition is mutually disjoint and exhaustive, the cardinalities of the sets just add up:

$$\begin{aligned}
|children| &= |c| + |cs| + |cd| + |csd| \\
|sons| &= |s| + |cs| + |sd| + |csd| \\
|daughters| &= |d| + |cd| + |sd| + |csd|
\end{aligned}$$

Formula (1) can now be rewritten into

$$\begin{aligned}
&|s| + |cs| + |sd| + |csd| = 2 \wedge |d| + |cd| + |sd| + |csd| = 3 \\
&\qquad \Rightarrow\ |c| + |cs| + |cd| + |csd| = 5
\end{aligned} \tag{2}$$

or by dropping the cardinality function $|...|$

$$s + cs + sd + csd = 2 \ \wedge\ d + cd + sd + csd = 3 \quad \Rightarrow \quad c + cs + cd + csd = 5 \tag{3}$$

In (3) we interpret the symbols $c, s, \ldots$ directly as the numbers denoting the cardinality of the corresponding sets. This makes sense because the sets are finite and mutually disjoint. This way, Problem (1) has been transformed into a pure non-negative linear Diophantine equation problem.
Diophantine equations are equations with integer-valued variables. They are called "linear" if no products of different variables occur and "non-negative" if variables are constrained to non-negative integers. Formula (3), of course, is still not valid. Further information is necessary.

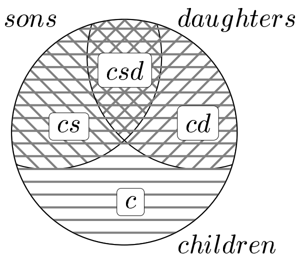


Figure 2: The correct subset relationships.

First of all we add the subset information. Both sons and daughters are children. This means the first picture was a bit too general. We are in a more specific situation which is depicted in Figure 2. Here all sets not containing c are empty reflecting the subset information. That means $s = 0$, $d = 0$, $sd = 0$ holds and we can simplify Problem (3) to

$$cs + csd = 2 \ \wedge\ cd + csd = 3 \quad \Rightarrow \quad c + cs + cd + csd = 5 \tag{4}$$

which is still not valid. The next piece of information, we need, is that there are no hermaphrodites. This means the intersection of sons and daughters is empty: $csd = 0$. We obtain

$$cs = 2 \ \wedge\ cd = 3 \quad \Rightarrow \quad c + cs + cd = 5$$

Finally, we exploit the fact that there are no other types of children besides sons and daughters, i.e. $c = 0$ and we end up with

$$cs = 2 \ \wedge\ cd = 3 \quad \Rightarrow \quad cs + cd = 5 \tag{5}$$

and this is in fact valid.

Let us recall the steps we performed. We started with the most general decomposition of the three sets into their atomic components. This allowed us to turn cardinality terms like $|sons|$ into arithmetic terms and we got a pure linear Diophantine equation problem. Then we exploited the *subset*, *disjointness*, and *exhaustiveness* relations between the different sets. Each such relation made some of the atoms empty and simplified the problem. This phenomena is in sharp contrast to most other encodings in logical systems where additional information complicates matters rather than simplifies it. Finally we ended up with the simple Formula (5) we could submit to an arithmetic equation solver.

The paper is organized as follows: In Section 2, we formalize the atomic decomposition idea and present optimized methods for reducing the exponential number of atoms. Section 3 embeds the transformation into a general arithmetic language with a suitable calculus for reasoning with arithmetic constraints. In Section 4, we apply the mechanism to the concept language $\mathcal{TF}^{++}$ and show how the consistency and subsumption problems over concept terms can be reduced to solvability tests for systems of equations. The language $\mathcal{TF}^{++}$ is an extension of the language $\mathcal{TF}$ [Neb90] which adds more complex role hierarchies and a more expressive arithmetic constraint part to the original core language. We conclude in Section 5 with some final remarks on equational problem solvers and an outlook on current work.

2 Atomic Decomposition of Sets

The introductory example in Figure 1 illustrated the semantic side of the decomposition idea. A set of sets is decomposed into basic non-overlapping components. In this section, we develop a syntactic representation of the decompositions and introduce inference systems working on the syntactic representation.

The basis for the syntactic representation is a *finite* set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of terms in some term language of some application. Each s_i is supposed to denote a subset of some global set $\mathcal{D}$. The internal structure of the terms s_i is irrelevant for the purpose of the decomposition, therefore we can treat them as constant symbols. In the introductory 'children' example, we had $\mathcal{S} = \{sons, daughters, children\}$. In addition to the symbols in $\mathcal{S}$ we have the fixed symbols $\top$ denoting the whole domain $\mathcal{D}$ and $\perp$ denoting the empty set.

At a first glance, the atoms in an atomic decomposition of some sets seem to be some kind of intersection. We could, for example, take subsets $\{s_i, \ldots, s_j\} \subseteq \mathcal{S}$ as a representation of an atom. The subsets containing two and more elements might be interpreted as intersections $s_i \cap \ldots \cap s_j$. But this is only half the story. The actual atom is only that part of the intersection which does not belong to any other intersection. For example, in Figure 1, the set denoted by "*sd*"[1] does not mean the intersection $sons \cap daughters$, but $(sons \cap daughters) \setminus (sons \cap daughters \cap children)$. This way we also get an interpretation for the one-element subsets of $\mathcal{S}$. The set $\{s_i\}$ denotes that part of (the interpretation of) s_i which does not belong to any other intersection. Even the empty subset $\emptyset$ has a meaning. It denotes the complement of the union of all elements of $\mathcal{S}$. In our example, $\emptyset$ denotes all the non-children in the given domain.

This way, we get 2^n atoms for $\mathcal{S}$, which can be represented by the subsets of $\mathcal{S}$. To distinguish the atoms from the elements in $\mathcal{S}$, and to simplify notation we usually write them as strings. For example, if $\mathcal{S} = \{p, c, m\}$ then the *decomposition* $\alpha_{\mathcal{S}}(\top)$ is $\{\emptyset, p, c, m, pc, pm, cm, pcm\}$. That means the whole domain $\mathcal{D}$ is separated into 2^3 subsets denoted by these 8 strings. The string pc stands for the set $\{p, c\}$. As a further notational convention, a string wp where w is an atom and p a single term stands for $w \cup \{p\}$.

The atomic decomposition $\alpha_{\mathcal{S}}(s)$ of a single element $s \in \mathcal{S}$ now consists of those atoms in $\alpha_{\mathcal{S}}(\top)$ which have a s-component. For the above example we get $\alpha_{\mathcal{S}}(p) = \{p, pc, pm, pcm\}$.

[1] Take "*sd*" as an abbreviation for $\{sons, daughters\}$.

Definition 2.1 (Atomic Decomposition) *Let* $\mathcal{S} \stackrel{\text{def}}{=} \{s_1, \ldots, s_n\}$ *be a* finite *set of terms of some term language which we also denote with* $\mathcal{S}$ *(for the purpose of this paper there is no difference).*

i) We define the atomic decomposition $\alpha_{\mathcal{S}}(\top)$ *as the set of subsets of* $\mathcal{S}$*. The elements of the decomposition are called the* atoms*.*

ii) For a given $s \in \mathcal{S}$ *let* $\alpha_{\mathcal{S}}(s) \stackrel{\text{def}}{=} \{w \in \alpha_{\mathcal{S}}(\top) \mid s \in w\}$ *be the* $\mathcal{S}$-decomposition *(or* atomic decomposition *or just simply* decomposition*) of* s*. Notice that this definition automatically guarantees* $\alpha_{\mathcal{S}}(\bot) = \emptyset$*.*

The elements of $\mathcal{S} \cup \{\top, \bot\}$ *will be called* basic set-terms*.* ■

For the purpose of atomic decompositions, the internal structure of the terms $s \in \mathcal{S}$ is irrelevant. We only need to know that they denote sets. In technical terms this means that we can take any interpretation $\mathcal{E}$ for $\mathcal{S}$, which may come from some application, and which maps the elements of $\mathcal{S}$ to some sets, and extend it to an interpretation $\mathcal{E}' = \alpha_{\mathcal{S}}(\mathcal{E})$ which gives the intended meaning to the atoms in $\alpha_{\mathcal{S}}(\top)$.[2]

The atoms $s_i \ldots s_j$ denote those parts of the intersection $\mathcal{E}(s_i) \cap \ldots \cap \mathcal{E}(s_j)$ which do not belong to any other intersection, or, which is the same, those parts of the intersection which are not part of some $s_k \notin \{s_i, \ldots, s_j\}$. This is made precise in the next definition.

Definition 2.2 (Interpretation of Atomic Decompositions) *For a set* $\mathcal{S}$ *of terms and its decomposition* $\alpha_{\mathcal{S}}(\top)$ *we extend the definition of* $\alpha_{\mathcal{S}}$ *to interpretations:*

i) Let $\mathcal{E}$ *be an* interpretation *function which assigns some set* $\mathcal{E}(s) \subseteq \mathcal{D}$ *of objects in some domain* $\mathcal{D}$ *to each member of* $s \in \mathcal{S}$*.* $\mathcal{E}(\top) = \mathcal{D}$ *and* $\mathcal{E}(\bot) = \emptyset$ *are required. We define the* extended interpretation function $\alpha_{\mathcal{S}}(\mathcal{E})$ *to be a new function which is like* $\mathcal{E}$*, but which assigns sets to the atoms as well: For* $w \in \alpha_{\mathcal{S}}(\top)$
$$\alpha_{\mathcal{S}}(\mathcal{E})(w) \stackrel{\text{def}}{=} \bigcap_{r \in w} \mathcal{E}(r) \setminus \bigcup_{s \in \mathcal{S}, s \notin w} \mathcal{E}(s).$$
Notice that the intersection over an empty index set is the whole domain $\mathcal{D}$*. Therefore* $\alpha_{\mathcal{S}}(\mathcal{E})(\emptyset) = \mathcal{D} \setminus \bigcup_{s \in \mathcal{S}} \mathcal{E}(s)$ *is defined as well.*

ii) For a set $\{w_1, \ldots, w_k\}$ *of atoms we define*
$$\alpha_{\mathcal{S}}(\mathcal{E})(\{w_1, \ldots, w_k\}) \stackrel{\text{def}}{=} \alpha_{\mathcal{S}}(\mathcal{E})(w_1) \cup \ldots \cup \alpha_{\mathcal{S}}(\mathcal{E})(w_k).$$
■

As an example, consider the set $\mathcal{S} = \{r, s\}$ with $\alpha_{\mathcal{S}}(\top) = \{\emptyset, r, s, rs\}$.
Suppose $\mathcal{D} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $\mathcal{E}(r) = \{1, 2, 3, 4, 5\}$ and $\mathcal{E}(s) = \{4, 5, 6, 7\}$. Then we get

$$\begin{array}{lclcl}
\alpha_{\mathcal{S}}(\mathcal{E})(\emptyset) & = & \mathcal{D} \setminus (\mathcal{E}(r) \cup \mathcal{E}(s)) & = & \{8, 9, 10\} \\
\alpha_{\mathcal{S}}(\mathcal{E})(r) & = & \mathcal{E}(r) \setminus \mathcal{E}(s) & = & \{1, 2, 3\} \\
\alpha_{\mathcal{S}}(\mathcal{E})(s) & = & \mathcal{E}(s) \setminus \mathcal{E}(r) & = & \{6, 7\} \\
\alpha_{\mathcal{S}}(\mathcal{E})(rs) & = & \mathcal{E}(r) \cap \mathcal{E}(s) & = & \{4, 5\},
\end{array}$$

and this is a partitioning of $\mathcal{D}$. With some quite simple arguments we can confirm that the definition of $\alpha_{\mathcal{S}}(\mathcal{E})$ gives the intended meaning to the atoms. We have to check that each set is decomposed into an exhaustive and mutually disjoint set of atoms.

Lemma 2.3 (Adequacy of $\alpha_{\mathcal{S}}(\mathcal{E})$) *For a set* $\mathcal{S}$ *of terms and* extended *interpretation function* $\mathcal{E}' \stackrel{\text{def}}{=} \alpha_{\mathcal{S}}(\mathcal{E})$ *(Definition 2.2):*

i) $\mathcal{E}'(u) \cap \mathcal{E}'(v) = \emptyset$ *for each* $u \neq v \in \alpha_{\mathcal{S}}(\top)$*.* *(disjointness)*

ii) For each $r \in \mathcal{S}$*:* $\mathcal{E}(r) = \bigcup_{w \in \alpha_{\mathcal{S}}(r)} \mathcal{E}'(w)$*.* *(exhaustiveness)*

[2] In the concept language example in Section 4, the elements of $\mathcal{S}$ will be so-called role names. They are interpreted as binary relations.

Proof:

i) Let $u = r_1 \dots r_k$ and $v = s_1 \dots s_l$. Since $u \neq v$ there is w.l.o.g. some $r_j \in u$ but $r_j \notin v$. If for some a we have $a \in \mathcal{E}'(u) = [\mathcal{E}(r_1) \cap \dots \cap \mathcal{E}(r_k)] \setminus \bigcup_{s \notin u} \mathcal{E}(s)$ then $a \in \mathcal{E}(r_j)$ and $a \notin \mathcal{E}'(v) = [\mathcal{E}(s_1) \cap \dots \cap \mathcal{E}(s_l)] \setminus [\mathcal{E}(r_j) \cup \dots]$. Thus, $\mathcal{E}'(u) \cap \mathcal{E}'(v) = \emptyset$.

ii) "$\subseteq$": Let $a \in \mathcal{E}(r)$. Let $w \stackrel{\text{def}}{=} \{s_1, \dots, s_l\}$ be such that $a \in \mathcal{E}(s_j)$, $1 \leq j \leq l$ and $a \notin \mathcal{E}(s)$ for $s \notin w$, $s \in \mathcal{S}$. In particular, $r \in w$. Now $a \in \mathcal{E}'(w)$ and therefore $a \in \bigcup_{w \in \alpha_{\mathcal{S}}(r)} \mathcal{E}'(w)$. Thus, $\mathcal{E}(r) \subseteq \bigcup_{w \in \alpha_{\mathcal{S}}(r)} \mathcal{E}'(w)$.

"$\supseteq$": Let $a \in \mathcal{E}(w)$, $w \in \alpha_{\mathcal{S}}(r)$, i.e. $w = rs_1 \dots s_k$. Then $a \in [\mathcal{E}(r) \cap \mathcal{E}(s_1) \cap \dots \cap \mathcal{E}(s_k)] \setminus \dots$, which means $a \in \mathcal{E}(r)$. Thus, $\mathcal{E}(r) \supseteq \bigcup_{w \in \alpha_{\mathcal{S}}(r)} \mathcal{E}(w)$. ■

Based on the symbols in $\mathcal{S}$, we can now define *set-terms* constructed with the usual Boolean set connectives $\cap$, $\cup$ and $\setminus$. It turns out that terms composed with set connectives can be decomposed by decomposing the arguments and then applying the set connective to the decomposed arguments.

To illustrate this, we again use the example from Section 1. In Figure 1 we can directly see that for example

$$\begin{aligned}
\alpha_{\mathcal{S}}(sons \cap daughters) &= \{sd, csd\} \\
&= \{s, cs, sd, csd\} \cap \{d, cd, sd, csd\} \\
&= \alpha_{\mathcal{S}}(sons) \cap \alpha_{\mathcal{S}}(daughters) \\
\alpha_{\mathcal{S}}(sons \cup daughters) &= \{s, cs, d, cd, sd, csd\} \\
&= \{s, cs, sd, csd\} \cup \{d, cd, sd, csd\} \\
&= \alpha_{\mathcal{S}}(sons) \cup \alpha_{\mathcal{S}}(daughters) \\
\alpha_{\mathcal{S}}(sons \setminus daughters) &= \{s, cs\} \\
&= \{s, cs, sd, csd\} \setminus \{d, cd, sd, csd\} \\
&= \alpha_{\mathcal{S}}(sons) \setminus \alpha_{\mathcal{S}}(daughters).
\end{aligned}$$

Definition 2.4 (Set-Terms and their Decomposition) *Again let $\mathcal{S}$ be a finite set of terms.*

i) We define the set $S_{\mathcal{S}}$ of (composed) set-terms *based on $\mathcal{S}$ as the least set of terms such that $\mathcal{S} \subseteq S_{\mathcal{S}}$ and if φ and ψ are set-terms in $S_{\mathcal{S}}$ then $\varphi \cup \psi$, $\varphi \cap \psi$ and $\varphi \setminus \psi$ are set-terms in $S_{\mathcal{S}}$.*

ii) The atomic decomposition $\alpha_{\mathcal{S}}(\varphi)$ of a set-term φ is defined in the obvious way:
If $\varphi \in \mathcal{S}$ then $\alpha_{\mathcal{S}}(\varphi)$ is as in Definition 2.1,ii.

$$\begin{aligned}
\alpha_{\mathcal{S}}(\varphi \cup \psi) &\stackrel{\text{def}}{=} \alpha_{\mathcal{S}}(\varphi) \cup \alpha_{\mathcal{S}}(\psi) \\
\alpha_{\mathcal{S}}(\varphi \cap \psi) &\stackrel{\text{def}}{=} \alpha_{\mathcal{S}}(\varphi) \cap \alpha_{\mathcal{S}}(\psi) \\
\alpha_{\mathcal{S}}(\varphi \setminus \psi) &\stackrel{\text{def}}{=} \alpha_{\mathcal{S}}(\varphi) \setminus \alpha_{\mathcal{S}}(\psi).
\end{aligned}$$

iii) The interpretation of set-terms is done by taking the standard definition of the set connectives ($\cap$ = intersection, $\cup$ = union, $\setminus$ = set difference). ■

Notice that we used the same symbols $\cap$, $\cup$ and $\setminus$ at both the language level and the meta level. This is justified because, as the next lemma confirms, they have exactly the same meaning.

Lemma 2.5 *For a finite set $\mathcal{S}$ of terms, a corresponding extended interpretation function $\mathcal{E}$ and two set-terms φ and ψ:*

i) $\mathcal{E}(\alpha_{\mathcal{S}}(\varphi \cup \psi)) = \mathcal{E}(\alpha_{\mathcal{S}}(\varphi)) \cup \mathcal{E}(\alpha_{\mathcal{S}}(\psi))$

ii) $\mathcal{E}(\alpha_{\mathcal{S}}(\varphi \cap \psi)) = \mathcal{E}(\alpha_{\mathcal{S}}(\varphi)) \cap \mathcal{E}(\alpha_{\mathcal{S}}(\psi))$

iii) $\mathcal{E}(\alpha_{\mathcal{S}}(\varphi \setminus \psi)) = \mathcal{E}(\alpha_{\mathcal{S}}(\varphi)) \setminus \mathcal{E}(\alpha_{\mathcal{S}}(\psi))$.

Proof: We verify only the second statement. The others are analogous.

$$\begin{aligned}
\mathcal{E}(\alpha_{\mathcal{S}}(\varphi \cap \psi)) &= \mathcal{E}(\alpha_{\mathcal{S}}(\varphi) \cap \alpha_{\mathcal{S}}(\psi)) && \text{(Definition 2.4,}\textit{ii}\text{)} \\
&= \textstyle\bigcup_{w \in \alpha_{\mathcal{S}}(\varphi) \cap \alpha_{\mathcal{S}}(\psi)} \mathcal{E}(w) && \text{(Definition 2.2,}\textit{ii}\text{)} \\
&= \textstyle\bigcup_{w \in \alpha_{\mathcal{S}}(\varphi)} \mathcal{E}(w) \cap \bigcup_{w \in \alpha_{\mathcal{S}}(\psi)} \mathcal{E}(w) \\
&= \mathcal{E}(\alpha_{\mathcal{S}}(\varphi)) \cap \mathcal{E}(\alpha_{\mathcal{S}}(\psi)) && \text{(Definition 2.2,}\textit{ii}\text{)}
\end{aligned}$$

In the second-but-last step we made use of the disjointness of the decomposition (cf. Lemma 2.3,*i*). The "$\subseteq$" inclusion is trivial. For the "$\supseteq$" direction, suppose a is a member of the lower set. There must be some $w \in \alpha_{\mathcal{S}}(\varphi) \cup \alpha_{\mathcal{S}}(\psi)$ with $a \in \mathcal{E}(w)$. If w is not in the intersection $\alpha_{\mathcal{S}}(\varphi) \cap \alpha_{\mathcal{S}}(\psi)$, it is either in $\alpha_{\mathcal{S}}(\varphi)$ or in $\alpha_{\mathcal{S}}(\psi)$, but not in both. However, since the atoms are interpreted as mutually disjoint sets, a cannot be in $\mathcal{E}(w)$ for this w. Thus, w must be in the intersection, and therefore a is a member of the upper set as well. ∎

As a consequence of this Lemma, it is now easy to verify that not only for the terms in $\mathcal{S}$, but for all set-terms built on $\mathcal{S}$, one can compute the cardinality of the set associated to the set-term by adding up the cardinalities of the atoms in its decomposition.

Corollary 2.6 *For a finite set $\mathcal{S}$ of terms, an interpretation function $\mathcal{E}$ and a set-term φ with decomposition $\alpha_{\mathcal{S}}(\varphi) = \{w_1, \ldots, w_k\}$:*

i) $\mathcal{E}(\varphi) = \alpha_{\mathcal{S}}(\mathcal{E})(\alpha_{\mathcal{S}}(\varphi)) = \alpha_{\mathcal{S}}(\mathcal{E}(w_1)) \cup \ldots \cup \alpha_{\mathcal{S}}(\mathcal{E}(w_k))$.

ii) $|\mathcal{E}(\varphi)| = |\alpha_{\mathcal{S}}(\mathcal{E})(w_1)| + \ldots + |\alpha_{\mathcal{S}}(\mathcal{E})(w_k)|$ *where* $|\ldots|$ *denotes the cardinality of a set and* $+$ *is the addition of cardinal numbers.* ∎

The proof of (*i*) is by induction on the structure of φ using the same arguments as in Lemma 2.5. Statement (*ii*) is a consequence of (*i*) and the disjointness of the decomposition (cf. Lemma 2.3,*i*).

Recall the different notions of interpretations we have introduced. We started with an interpretation function $\mathcal{E}$ for the terms $\mathcal{S}$, and we assumed, this was given to us by somebody. $\mathcal{E}$ was then extended to $\alpha_{\mathcal{S}}(\mathcal{E})$ by adding suitable interpretations for the atoms, and finally we integrated the standard set operations $\cap$, $\cup$ and $\setminus$. *From now on, an interpretation for the set-terms always means an extended interpretation of this kind.*

2.1 Set Hierarchies

The decomposition of a set $\mathcal{S}$ with n elements yields 2^n, i.e. *exponentially many* atoms. Even for small numbers n, this can become unmanageably large. Therefore, every possible way to reduce the number of atoms needs to be exploited. The 2^n atoms describe the most general way n sets can be related to each other. Special relationships between the n sets, in particular subset and disjointness, however, puts us into a more special situation where some of the atoms denote empty sets. The best way to exploit this information is by eliminating these empty atoms from the decomposition itself. That means, if for example $\alpha_{\mathcal{S}}(r) = \{w_1, \ldots, w_n\}$ and, by some extra condition we know that say, w_1 denotes an empty set, we do not store $w_1 = \emptyset$, but we change the definition of $\alpha_{\mathcal{S}}$ such that $\alpha'_{\mathcal{S}}(r) = \{w_2, \ldots, w_n\}$.

In this way α does not only depend on $\mathcal{S}$, but also on extra information about the relationships between sets. In order to formalize this, we introduce *hierarchy specifications*, and we present two different ways for generating an optimal decomposition function from a hierarchy specification.[3]

Definition 2.7 (Hierarchy Specification) *A* hierarchy specification $\mathcal{H}$ *over a set $\mathcal{S}$ is a list of* hierarchy declarations *of the following kind:*

i) a basic set-term $r \in \mathcal{S}$ is a hierarchy declaration (non-emptyness).

ii) If φ, ψ are set-terms in $S_{\mathcal{S}}$ then $\varphi \subseteq \psi$ is a hierarchy declaration (subset).

iii) If φ and ψ are set-terms in $S_{\mathcal{S}}$ then $\varphi \dagger \psi$ is a hierarchy declaration (disjointness).

iv) If $\varphi_1, \ldots, \varphi_n, \varphi$ are set-terms in $S_{\mathcal{S}}$ then
$\varphi_1, \ldots, \varphi_n \lhd \varphi$ is a hierarchy declaration (partitioning).

∎

[3]Note that the decomposition function is not meant to be an optimal function in the sense of memory or runtime requirements, but in the sense that it computes an *optimal decomposition* containing only the non-empty atoms.

An example for a hierarchy specification is

$$\begin{aligned}\mathcal{H} \quad = \quad & \{children,\\ & daughters, sons \lhd children,\\ & favourite\text{-}children \subseteq children\}\end{aligned} \tag{6}$$

The first declaration, *children*, just introduces the constant *children*. In principle this would not be necessary, but we shall use the non-emptyness declaration for guiding the optimal computation of a decomposition function. The second declaration $daughters, sons \lhd children$ declares that the set *children* is partitioned into *daughters* and *sons*. Finally, the third declaration $favourite\text{-}children \subseteq children$ introduces *favourite-children* as a subset of children. Notice that all the terms in hierarchy declarations may be arbitrary set terms. For example

$$favourite\text{-}children \subseteq children \cap blondes$$

which specifies *favourite-children* as a subset of the intersection of *children* and *blondes* is a valid declaration. Another example is

$$sons \cap blondes, daughters \cap blondes \lhd blonde\text{-}children$$

which declares that the blonde children consist of blonde sons and blonde daughters.

Definition 2.8 (Semantics of Hierarchy Specifications) *A model $\mathcal{E}$* satisfies *a hierarchy specification $\mathcal{H}$, written $\mathcal{E} \models \mathcal{H}$, iff*

- *i)* $\mathcal{E}(\varphi) \neq \emptyset$ *for each non-emptyness declaration* $\varphi \in \mathcal{H}$,
- *ii)* $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ *for each subset declaration* $\varphi \subseteq \psi$ *in* $\mathcal{H}$,
- *iii)* $\mathcal{E}(\varphi) \cap \mathcal{E}(\psi) = \emptyset$ *for each disjointness declaration* $\varphi \dagger \psi$ *in* $\mathcal{H}$,
- *iv) for each partitioning declaration* $\varphi_1, \ldots, \varphi_n \lhd \varphi$ *in* $\mathcal{H}$ *we have*
  1. $\mathcal{E}(\varphi_i) \cap \mathcal{E}(\varphi_j) = \emptyset$ *for all* $i, j = 1, \ldots n$ *with* $i \neq j$, *and*
  2. $\mathcal{E}(\varphi_1) \cup \ldots \cup \mathcal{E}(\varphi_n) = \mathcal{E}(\varphi)$.

The first algorithm for generating a decomposition function for a hierarchy specification works by generating all atoms first and then it successively eliminates unnecessary atoms.

Definition 2.9 (Decrementing Algorithm) *Given a hierarchy specification $\mathcal{H}$ over a set $\mathcal{S}$ of terms, we define a decomposition function $\alpha^D_{\mathcal{H}}$ inductively:*

- *i)* $\alpha^D_{\emptyset} \stackrel{\text{def}}{=} \alpha_{\mathcal{S}}$.
- *ii)* $\alpha^D_{\mathcal{H}\cup\{r\}} \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H}}$ *for a non-emptyness declaration* r.
- *iii)* $\alpha^D_{\mathcal{H}\cup\{\varphi\subseteq\psi\}}(s) \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H}}(s) \setminus [\alpha^D_{\mathcal{H}}(\varphi) \setminus \alpha^D_{\mathcal{H}}(\psi)]$ *for a subset declaration* $\varphi \subseteq \psi$.
- *iv)* $\alpha^D_{\mathcal{H}\cup\{\varphi\dagger\psi\}}(s) \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H}}(s) \setminus \alpha^D_{\mathcal{H}}(s \cap \varphi \cap \psi)$ *for a disjointness declaration* $\varphi \dagger \psi$.
- *v)* $\alpha^D_{\mathcal{H}\cup\{\varphi_1,\ldots,\varphi_n\lhd\varphi\}}(s) \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H}}(s) \setminus \big[\bigcup_{i\neq j} \alpha^D_{\mathcal{H}}(\varphi_i \cap \varphi_j) \cup [\alpha^D_{\mathcal{H}}(\varphi_1 \cup \ldots \cup \varphi_n) \setminus \alpha^D_{\mathcal{H}}(\varphi)] \cup [\alpha^D_{\mathcal{H}}(\varphi) \setminus \alpha^D_{\mathcal{H}}(\varphi_1 \cup \ldots \cup \varphi_n)]\big]$
  for a partitioning declaration $\varphi_1, \ldots, \varphi_n \lhd \varphi$.

■

For the following hierarchy specification example

$$\begin{aligned}\mathcal{H} \quad = \quad & \{daughters, sons \lhd children,\\ & favourite\text{-}children \subseteq children\}\end{aligned} \tag{7}$$

the decrementing algorithm works as follows: since $\mathcal{S}$ consists of the four terms

$$\{\mathit{children}, \mathit{sons}, \mathit{daughters}, \mathit{favourite\text{-}children}\}$$

we start with the full decomposition:

$$\alpha^D_\emptyset(\top) \quad = \quad \{\emptyset, c, s, d, f, cs, cd, cf, sd, sf, df, csd, csf, cdf, sdf, csdf\}$$

The string c abbreviates $\{\mathit{children}\}$ and similar abbreviations were chosen for the other atoms. First, we process the partitioning declaration $\mathit{daughters}, \mathit{sons} \lhd \mathit{children}$.
According to Definition 2.9,v, the following atoms are to be eliminated in order to obtain $\alpha^D_1 \stackrel{\text{def}}{=} \alpha^D_{\{\mathit{daughters},\mathit{sons}\lhd\mathit{children}\}}$:

a) $\alpha^D_\emptyset(\mathit{sons} \cap \mathit{daughters}) = \{sd, csd, sdf, csdf\}$

b) $\alpha^D_\emptyset(\mathit{sons} \cup \mathit{daughters}) \setminus \alpha^D_\emptyset(\mathit{children})$
$= \{s, d, cs, cd, sd, sf, df, csd, csf, cdf, sdf, csdf\} \setminus \ \{c, cs, cd, cf, csd, csf, cdf, sdf, csdf\}$
$= \{s, d, sf, df\}$.

c) $\alpha^D_\emptyset(\mathit{children}) \setminus \alpha^D_\emptyset(\mathit{sons} \cup \mathit{daughters})$
$= \{c, cs, cd, cf, csd, csf, cdf, csdf\} \ \setminus \{s, d, cs, cd, sd, sf, df, csd, csf, cdf, sdf, csdf\}$
$= \{c, cf\}$.

What remains is: $\alpha^D_1(\top) = \{\emptyset, f, cs, cd, csf, cdf\}$.

Now we process the subset declaration $\mathit{favourite\text{-}children} \subseteq \mathit{children}$. According to Definition 2.9,iii, the following atoms are to be eliminated in order to obtain $\alpha^D_2 \stackrel{\text{def}}{=} \alpha^D_{\{\mathit{daughters},\mathit{sons}\lhd\mathit{children},\mathit{favourite\text{-}children}\subseteq\mathit{children}\}}$:

$\alpha^D_1(\mathit{favourite\text{-}children}) \setminus \alpha^D_1(\mathit{children}) = \{f, csf, cdf\} \setminus \{cs, cd, csf, cdf\} = \{f\}$.

Finally we obtain

$$\begin{array}{lcl} \alpha^D_2(\top) & = & \{\emptyset, cs, cd, csf, cdf\} \\ \alpha^D_2(\mathit{children}) & = & \{cs, cd, csf, cdf\} \\ \alpha^D_2(\mathit{sons}) & = & \{cs, csf\} \\ \alpha^D_2(\mathit{daughters}) & = & \{cd, cdf\} \\ \alpha^D_2(\mathit{favourite\text{-}children}) & = & \{csf, cdf\}. \end{array}$$

This is what is to be expected, but the way it is obtained is not satisfactory because we still start with an exponential number of atoms. A better algorithm would not generate the superfluous atoms at all. Nevertheless, the algorithm is sound and complete, i.e. we have not removed too many atoms, and all atoms which must denote empty sets if the hierarchy specification is to be respected, have actually been removed.

Proposition 2.10 (Soundness of the Decrementing Algorithm)
For a given hierarchy specification $\mathcal{H}$ over a set $\mathcal{S}$ of terms and for every model $\mathcal{E} \models \mathcal{H}$ and every set-term φ we have $\mathcal{E}(\varphi) = \mathcal{E}(\alpha^D_\mathcal{H}(\varphi))$, i.e. not too many atoms have been eliminated.

Proof: It is sufficient to show the statement for the basic set-terms $s \in \mathcal{S}$. The proposition then follows by induction on the structure of set terms.
We perform an induction on the number of declarations in $\mathcal{H}$. The case $\mathcal{H} = \emptyset$, where $\alpha^D_\mathcal{H}(\top)$ computes the full power-set, is actually the statement of Corollary 2.6,i. In the induction step, we have to consider the two non-trivial cases where a subset declaration and a partitioning declaration is processed. Both cases follow from the semantics of the declarations in Definition 2.8 and the deletion rules.
Case 1: a subset declaration $\varphi \subseteq \psi$ is processed. Let $\alpha^D_\mathcal{H}(s) = \{w_1, \ldots, w_k\}$. By the induction hypothesis we have $\mathcal{E}(s) = \mathcal{E}(w_1) \cup \ldots \cup \mathcal{E}(w_k)$. Suppose for some $w_i \in \{w_1, \ldots, w_k\}$, $w_i \notin \alpha^D_{\mathcal{H}\cup\{\varphi\subseteq\psi\}}$. w_i has been eliminated because $w_i \in \alpha^D_\mathcal{H}(\varphi)$ and $w_i \notin \alpha^D_\mathcal{H}(\psi)$. Since $\mathcal{E} \models (\varphi \subseteq \psi)$, i.e. $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$

holds, it must be $\mathcal{E}(w_i) = \emptyset$. Thus, $\mathcal{E}(w_1) \cup \ldots \cup \mathcal{E}(w_k) = \mathcal{E}(w_1) \cup \ldots \cup \mathcal{E}(w_{i-1}) \cup \mathcal{E}(w_{i+1}) \cup \ldots \cup \mathcal{E}(w_k)$. This is true for all eliminated atoms. Therefore $\mathcal{E}(s) = \mathcal{E}(\alpha^D_{\mathcal{H}\cup\{\varphi\subseteq\psi\}}(s))$.
The second case where a partitioning declaration is processed, is analogous. ∎

Proposition 2.11 (Completeness of the Decrementing Algorithm)
For a given hierarchy specification $\mathcal{H}$ over a set $\mathcal{S}$ of terms let $W = \alpha_\mathcal{S}(\top) \setminus \alpha^D_\mathcal{H}(\top)$ be the set of deleted atoms. Every extended model $\mathcal{E}$ for $\mathcal{S}$ with $\mathcal{E}(W) = \emptyset$ satisfies $\mathcal{H}$, i.e. enough atoms have been deleted.

Proof: We prove the statement again by induction on the number of declarations in $\mathcal{H}$. For the base case, $\mathcal{H} = \emptyset$, nothing needs to be proved.
For the induction step, let ξ be the subset declaration or the partitioning declaration to be processed. Let $W_\mathcal{H} \stackrel{\text{def}}{=} \alpha_\mathcal{S}(\top) \setminus \alpha^D_\mathcal{H}(\top)$ and $W_{\mathcal{H}\cup\{\xi\}} \stackrel{\text{def}}{=} \alpha_\mathcal{S}(\top) \setminus \alpha^D_{\mathcal{H}\cup\{\xi\}}(\top)$. $W_{\mathcal{H}\cup\{\xi\}} \supseteq W_\mathcal{H}$ because more atoms are deleted when processing ξ. Suppose $\mathcal{E}(W_{\mathcal{H}\cup\{\xi\}}) = \emptyset$. This means, $\mathcal{E}(W_\mathcal{H}) = \emptyset$ as well, and by the induction hypothesis, $\mathcal{E} \models \mathcal{H}$. It remains to be shown that $\mathcal{E} \models \xi$.
<u>Case 1</u>: ξ is a subset declaration $\varphi \subseteq \psi$. $W_{\mathcal{H}\cup\{\xi\}}$ contains all the removed atoms in $\alpha^D_\mathcal{H}(\varphi)$ which are not in $\alpha^D_\mathcal{H}(\psi)$. These are mapped to the empty set by $\mathcal{E}$. As a consequence of Corollary 2.6 we can therefore conclude $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$.
The case where ξ is a partitioning declaration is analogous. ∎

Instead of starting with the full decomposition of a set $\mathcal{S}$, and then deleting superfluous atoms, as the $\alpha^D_\mathcal{H}$ algorithm does, one can try to build up the decomposition incrementally, adding only those atoms which are necessary. For general hierarchy specifications as defined in Definition 2.7, we did not succeed in developing such an algorithm. But for a restricted class of *incremental* hierarchy specifications, this is possible.

Definition 2.12 (Incremental Hierarchy Specifications) *An* incremental hierarchy specification *is a hierarchy specification where*

i) each non-emptyness declaration is a new basic *set-term, i.e. it has not occurred in previous declarations (recall that hierarchy specifications are* ordered *sets),*
ii) for each subset declaration $\varphi \subseteq \psi$:
  a) φ is a new basic *set-term and*
  b) all basic set-terms occurring in ψ have been introduced before,
iii) for each partitioning declaration $\varphi_1, \ldots, \varphi_n \lhd \varphi$:
  a) $\varphi_1, \ldots, \varphi_n$ are new basic *set-terms and*
  b) all basic set-terms occurring in φ have been introduced before, and
iv) there is no disjointness declaration in $\mathcal{H}$. ∎

The example hierarchy specification (6) we had before is in fact incremental. It meets all the requirements. The second version (7) is not incremental, because in the partitioning declaration, *children* is new, which violates condition *iii.*

The incrementing decomposition algorithm we are going to define now has rules for each type of declaration. The way these rules work are best illustrated with some pictures. The first rule computes from a given decomposition, a new one for a single non-emptyness declaration r. To illustrate this, suppose the given decomposition decomposes a set s into three parts s_1, s_2, and s_3, cf. Figure 3. The new set r just splits these three parts into a non-r-part and a part intersecting with r. Thus, we keep s_1, s_2, and s_3 as the non-r-parts and add s_1r, s_2r and s_3r as the intersecting parts.

A subset declaration $r \subseteq \varphi$ is processed by splitting the overlapping parts of a decomposition for a term s with the decomposition of φ again into a non-r-part and into an r-part, cf. Figure 4.

Finally, for a partitioning declaration $r_1, \ldots, r_n \lhd \varphi$, the overlapping part of some s with φ is split into the n components $r_1, \ldots, r_n$, cf. Figure 5.

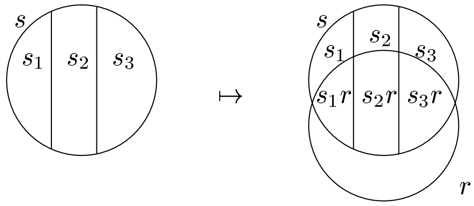


Figure 3: Incrementing algorithm for a non-emptyness declaration r.

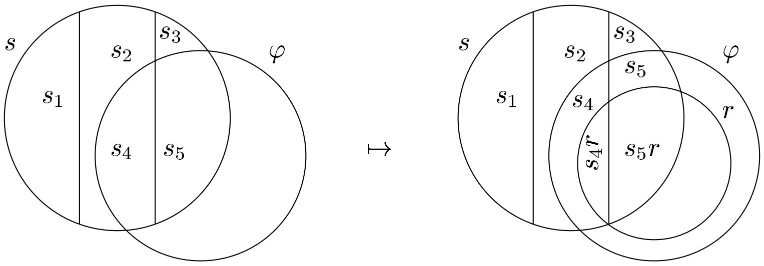


Figure 4: Incrementing algorithm for a subset declaration $r \subseteq \varphi$.

Definition 2.13 (Incrementing Algorithm) *We define an algorithm which computes a decomposition function $\alpha^I_{\mathcal{H}}$ and the set of basic set-terms $\mathcal{S}_{\mathcal{H}}$ from an incremental hierarchy specification $\mathcal{H}$. The definition is again recursive on the length of $\mathcal{H}$.*

i) $\alpha^I_{\emptyset}(\top) = \{\emptyset\}$, $\mathcal{S}_{\emptyset} = \emptyset$.

ii) $\alpha^I_{\mathcal{H}\cup\{r\}}(s) = \begin{cases} \alpha^I_{\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^I_{\mathcal{H}}(s)\} & \textit{for } s \neq r \\ \{wr \mid w \in \alpha^I_{\mathcal{H}}(\mathcal{S}_{\mathcal{H}})\} & \textit{for } s = r \end{cases}$

$\mathcal{S}_{\mathcal{H}\cup\{r\}} = \mathcal{S}_{\mathcal{H}} \cup \{r\}$ *for a non-emptyness declaration* r.

iii) $\alpha^I_{\mathcal{H}\cup\{r\subseteq\varphi\}}(s) = \begin{cases} \alpha^I_{\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^I_{\mathcal{H}}(s) \cap \alpha^I_{\mathcal{H}}(\varphi)\} & \textit{for } s \neq r \\ \{wr \mid w \in \alpha^I_{\mathcal{H}}(\varphi)\} & \textit{for } s = r \end{cases}$

$\mathcal{S}_{\mathcal{H}\cup\{r\subseteq\varphi\}} = \mathcal{S}_{\mathcal{H}} \cup \{r\}$ *for a subset declaration* $r \subseteq \varphi$.

iv) $\alpha^I_{\mathcal{H}\cup\{r_1,\ldots,r_n\lhd\varphi\}}(s) = \begin{cases} [\alpha^I_{\mathcal{H}}(s) \setminus \alpha^I_{\mathcal{H}}(\varphi)] \cup \bigcup_{w\in\alpha^I_{\mathcal{H}}(s\cap\varphi)}\{wr_1,\ldots,wr_n\} & \textit{for } s \neq r_i \\ \{wr_i \mid w \in \alpha^I_{\mathcal{H}}(\varphi)\} & \textit{for } s = r_i \end{cases}$

$\mathcal{S}_{\mathcal{H}\cup\{r_1,\ldots,r_n\lhd\varphi\}} = \mathcal{S}_{\mathcal{H}} \cup \{r_1,\ldots,r_n\}$ *for a partitioning declaration* $r_1,\ldots,r_n \lhd \varphi$. ■

As an example, consider again hierarchy specification (6):

$$\begin{aligned} \mathcal{H} \;=\; \{&children, \\ &daughters, sons \lhd children, \\ &favourite\text{-}children \subseteq children\} \end{aligned}$$

We abbreviate *children* with c and do the same with the other terms. The algorithm performs the following steps:

Start:

$$\begin{aligned} \mathcal{S}_0 \stackrel{\text{def}}{=} \mathcal{S}_{\emptyset} &= \emptyset \\ \alpha^I_0(\top) \stackrel{\text{def}}{=} \alpha^I_{\emptyset}(\top) &= \{\emptyset\}. \end{aligned}$$

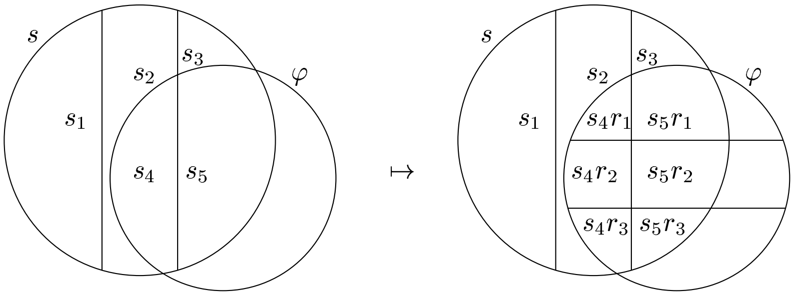


Figure 5: Incrementing algorithm for a partitioning declaration $r_1, r_2, r_3 \lhd \varphi$.

Processing of the non-emptyness declaration *children*:

$$\begin{aligned} \mathcal{S}_1 \stackrel{\text{def}}{=} \mathcal{S}_{\{children\}} &= \{children\} \\ \alpha_1^I(children) \stackrel{\text{def}}{=} \alpha^I_{\{children\}}(children) &= \{c\} \\ \alpha_1^I(\top) &= \{\emptyset, c\}. \end{aligned}$$

Processing of the partitioning declaration $daughters, sons \lhd children$:

$$\begin{aligned} \mathcal{S}_2 &= \{children, sons, daughters\} \\ \alpha_2^I(children) &\stackrel{\text{def}}{=} \alpha_1^I(children) \setminus \alpha_1^I(children) \cup \bigcup_{w \in \alpha_1^I(children \cap children)} \{ws, wd\} \\ &= \{cs, cd\} \\ \alpha_2^I(sons) &= \{cs\} \\ \alpha_2^I(daughters) &= \{cd\} \\ \alpha_2^I(\top) &= \{\emptyset, cs, cd\}. \end{aligned}$$

Processing of $favourite\text{-}children \subseteq children$:

$$\begin{aligned} \mathcal{S}_3 &= \{children, sons, daughters, favourite\text{-}children\} \\ \alpha_3^I(children) &\stackrel{\text{def}}{=} \alpha_2^I(children) \cup \{wf \mid w \in \alpha_2^I(children)\} \\ &= \{cs, cd, csf, cdf\} \\ \alpha_3^I(sons) &\stackrel{\text{def}}{=} \alpha_2^I(sons) \cup \{wf \mid w \in \alpha_2^I(sons)\} \\ &= \{cs, csf\} \\ \alpha_3^I(daughters) &\stackrel{\text{def}}{=} \alpha_2^I(daughters) \cup \{wf \mid w \in \alpha_2^I(daughters)\} \\ &= \{cd, cdf\} \\ \alpha_3^I(\top) &= \{\emptyset, cs, cd, csf, cdf\}. \end{aligned}$$

Proposition 2.14 (Equivalence of the Algorithms) *For every incremental hierarchy specification $\mathcal{H}$ and for $\mathcal{S} = \mathcal{S}_\mathcal{H}$ we have $\alpha^D_\mathcal{H} = \alpha^I_\mathcal{H}$.*

Proof: The decrementing algorithm not only depends on the hierarchy specification $\mathcal{H}$, but also on the set $\mathcal{S}$ of basic set-terms. For the incrementing algorithm we have not considered a fixed $\mathcal{S}$ but took $\mathcal{S}_\mathcal{H}$ from $\mathcal{H}$ itself. In order to compare both algorithms, we therefore need to make the dependence of $\mathcal{S}$ explicit for the decrementing algorithm. To this end we write $\alpha^D_{\mathcal{H},\mathcal{S}}$ instead of just

$\alpha^D_{\mathcal{H}}$. It is easy to prove by induction[4] on the number of declarations in $\mathcal{H}$, that for a basic set-term r not occurring in $\mathcal{H}$ and for s not containing r:

$$\alpha^D_{\mathcal{H},\mathcal{S}\cup\{r\}}(s) = \alpha^D_{\mathcal{H},\mathcal{S}}(s) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}}(s)\} \tag{8}$$

and more general[5], for basic set-terms $R = \{r_1, \ldots, r_n\}$ not occurring in $\mathcal{H}$ and for s not containing terms in R:

$$\alpha^D_{\mathcal{H},\mathcal{S}\cup R}(s) = \alpha^D_{\mathcal{H},\mathcal{S}}(s) \cup \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}}(s), u \subseteq \{r_1, \ldots, r_n\}\} \tag{9}$$

By induction on the number of declarations in $\mathcal{H}$ we prove $\alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s) = \alpha^I_{\mathcal{H}}(s)$ for all basic set-terms $s \in \mathcal{S}_\mathcal{H}$. The proposition then follows by induction on the structure of set-terms.
The base case, $\mathcal{H} = \emptyset$, is trivial because $\mathcal{S}_\emptyset = \emptyset$ and therefore $\alpha^D_{\emptyset,\mathcal{S}_\emptyset}(\top) = \{\emptyset\} = \alpha^I_\emptyset(\top)$. For the induction step we consider the different types ξ of declarations. In the sequel let $\mathcal{H}' \stackrel{\text{def}}{=} \mathcal{H} \cup \{\xi\}$.

<u>Case</u> $\xi = r$ is a non-emptyness declaration.

Subcase $s \neq r$:

$$\begin{aligned}
&\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(s)\\
&= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(s) && \text{(Definition 2.9,}\textit{ii}\text{)}\\
&= \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s)\} && \text{(cf. (8))}\\
&= \alpha^I_\mathcal{H}(s) \cup \{wr \mid w \in \alpha^I_\mathcal{H}(s)\} && \text{(induction hypothesis)}\\
&= \alpha^I_{\mathcal{H}'}(s). && \text{(Definition 2.13,}\textit{ii}\text{)}
\end{aligned}$$

Subcase $s = r$:

$$\begin{aligned}
&\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(r)\\
&= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) && \text{(Definition 2.9,}\textit{ii}\text{)}\\
&= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\top)\} && \text{(like (8), using } r \notin \mathcal{S}_\mathcal{H}\text{)}\\
&= \{wr \mid w \in \alpha^I_\mathcal{H}(\top)\} && \text{(induction hypothesis)}\\
&= \alpha^I_{\mathcal{H}'}(r). && \text{(Definition 2.13,}\textit{ii}\text{)}
\end{aligned}$$

<u>Case</u> $\xi = r \subseteq \varphi$ is a subset declaration.

Subcase $s \neq r$:

$$\begin{aligned}
&\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(s)\\
&= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(s) \setminus [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi)] && \text{(Definition 2.9,}\textit{iii}\text{)}\\
&= [\alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s)\}] && \text{(cf. (8))}\\
&\quad \setminus [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\top)\} && \text{(like (8), using } r \notin \mathcal{S}_\mathcal{H}\text{)}\\
&\qquad \setminus [\alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\varphi) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\varphi)\}]] && \text{(cf. (8))}\\
&= [\alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(s)\}]\\
&\quad \setminus [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\top)\} \setminus \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\varphi)\}] && \text{(see footnote[6])}
\end{aligned}$$

---

[4]The main argument is that the initial decomposition of a set $\mathcal{S}$ essentially yields the power-set of $\mathcal{S}$. We therefore have $\alpha_{\mathcal{S}\cup\{r\}}(\mathcal{S} \cup \{r\}) = 2^{\mathcal{S}\cup\{r\}} = 2^\mathcal{S} \cup \{wr \mid w \subseteq \mathcal{S}\} = 2^\mathcal{S} \cup \{wr \mid w \in \alpha_\mathcal{S}(\top)\}$. If r does not occur in a hierarchy specification, then for each atom w deleted by the decrementing algorithm, the corresponding element wr is deleted as well. Conversely, for each atom w without r there is still a corresponding element wr in $\alpha^D_{\mathcal{H}\cup\{r\}}(s)$.

[5]The proof is a generalization of the proof for (8).

[6]If we name $a \stackrel{\text{def}}{=} \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\top)\}$, $b \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\varphi)$ and $c \stackrel{\text{def}}{=} \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_\mathcal{H}}(\varphi)\}$ then we have $a \cap b = \emptyset$ and $c \subseteq a$. Therefore by elementary set theory: $a \setminus (b \cup c) = a \setminus c$.

$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \cup [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s)\} \cap \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\}] \qquad \text{(see footnote}^7\text{)}$$
$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \cup [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \cap \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\}]$$
$$= \alpha^I_{\mathcal{H}}(s) \cup \{wr \mid w \in \alpha^I_{\mathcal{H}}(s) \cap \alpha^I_{\mathcal{H}}(\varphi)\} \qquad \text{(induction hypothesis)}$$
$$= \alpha^I_{\mathcal{H}'}(s). \qquad \text{(Definition 2.13,}\mathit{iii}\text{)}$$

Subcase $s = r$:

$$\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(r)$$
$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) \setminus [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi)] \qquad \text{(Definition 2.9,}\mathit{iii}\text{)}$$
$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) \cap \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi) \qquad \text{(set theory: } a \setminus (a \setminus b) = a \cap b\text{)}$$
$$= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} \qquad \text{(like (8), using } r \notin \mathcal{S}_{\mathcal{H}}\text{)}$$
$$\quad \cap [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\}] \qquad \text{(cf. (8))}$$
$$= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\} \qquad \text{(see footnote}^8\text{)}$$
$$= \{wr \mid w \in \alpha^I_{\mathcal{H}}(\varphi)\} \qquad \text{(induction hypothesis)}$$
$$= \alpha^I_{\mathcal{H}'}(r). \qquad \text{(Definition 2.13,}\mathit{iii}\text{)}$$

<u>Case</u> $\xi = r_1, \ldots, r_n \lhd \varphi$. Let $R \stackrel{\text{def}}{=} \{r_1, \ldots, r_n\}$.

Subcase $s \neq r_i$, $i = 1, \ldots, n$.

$$\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(s)$$
$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(s) \setminus [\textstyle\bigcup_{i\neq j} \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r_i) \cap \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r_j)$$
$$\qquad \cup\ [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi)] \cup\ [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R)]] \qquad \text{(Definition 2.9,}v\text{)}$$
$$= [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \cup \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s), u \subseteq R\}] \qquad \text{(cf. (9))}$$
$$\quad \setminus [\textstyle\bigcup_{i\neq j}\{w \mid \{r_i, r_j\} \subseteq w, w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\top)\} \qquad \text{(like (9), using } r_i, r_j \notin \mathcal{S}_{\mathcal{H}}\text{)}$$
$$\qquad \cup [\{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\top), u \subseteq R\} \qquad \text{(like (9), using } r_i, r_j \notin \mathcal{S}_{\mathcal{H}}\text{)}$$
$$\qquad\quad \setminus [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi) \cup \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), u \subseteq R\}]] \qquad \text{(cf. (9))}$$
$$\qquad \cup [[\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi) \cup \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), u \subseteq R\}]$$
$$\qquad\quad \setminus \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top), u \subseteq R\}]$$
$$= [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \cup \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s), r \in R\}] \qquad \text{(exploiting disjointness of } r_i, r_j\text{)}$$
$$\quad \setminus [\{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top), w \notin \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), u \subseteq R\} \cup\ \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)] \qquad \text{(see footnote}^9\text{)}$$
$$= [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)] \qquad \text{(disjointness of } \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \text{ with the sets involving } R\text{)}$$
$$\quad \cup [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s), r \in R\} \cap \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), r \in R\}] \qquad \text{(set theory)}$$
$$= [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)] \cup \textstyle\bigcup_{w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s \cap \varphi)}\{wr_1, \ldots, wr_n\}$$
$$= [\alpha^I_{\mathcal{H}}(s) \setminus \alpha^I_{\mathcal{H}}(\varphi)] \cup \textstyle\bigcup_{w \in \alpha^I_{\mathcal{H}}(s \cap \varphi)}\{wr_1, \ldots, wr_n\} \qquad \text{(induction hypothesis)}$$
$$= \alpha^I_{\mathcal{H}'}(s). \qquad \text{(Definition 2.13,}\mathit{iv}\text{)}$$

Subcase $s = r \in \{r_1, \ldots, r_n\}$

$$\alpha^D_{\mathcal{H}',\mathcal{S}_{\mathcal{H}'}}(r)$$
$$= \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r) \setminus [[\textstyle\bigcup_{i\neq j} \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r_i) \cap \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(r_j)\ ]$$
$$\qquad \cup\ [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi)]\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(s) \setminus (\ \cup [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R)]] \qquad \text{(Definition 2.9,}v\text{)}$$

---

[7] If we name $a \stackrel{\text{def}}{=} \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(s)\}$, $b \stackrel{\text{def}}{=} \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\}$ and $c \stackrel{\text{def}}{=} \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\}$ then we have $a \subseteq b$ and $c \subseteq b$ and therefore by elementary set theory: $a \setminus (b \setminus c) = a \cap c$.

[8] because $\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} \cap \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi) = \emptyset$ and $\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} \supseteq \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\}$.

[9] If we name $a \stackrel{\text{def}}{=} \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\top), u \subseteq R\}$, $b \stackrel{\text{def}}{=} \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)$ and $c \stackrel{\text{def}}{=} \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi), u \subseteq R\}$, then we have $a \cap b = \emptyset$ and $c \subseteq a$. Therefore by elementary set theory: $a \setminus (b \cup c) = a \setminus c$ and $(b \cup c) \setminus a = b$.

$$
\begin{aligned}
&= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\top)\} && \text{(like (8) using } r \notin \mathcal{S}_{\mathcal{H}}) \\
&\quad \setminus [\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R) \setminus \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(\varphi)] && \text{(the rest is disjoint)} \\
&= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} \\
&\quad \setminus [\{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} && \text{(the rest is also disjoint)} \\
&\qquad \setminus \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), u \subseteq R\}] && \text{((9) and disjointness of } \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}'}}(R)) \\
&= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)\} \cap \{w \cup u \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi), u \subseteq R\} && \text{(set theory)} \\
&= \{wr \mid w \in \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi)\} && (\alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\varphi) \subseteq \alpha^D_{\mathcal{H},\mathcal{S}_{\mathcal{H}}}(\top)) \\
&= \{wr \mid w \in \alpha^I_{\mathcal{H}}(\varphi)\} && \text{(induction hypothesis)} \\
&= \alpha^I_{\mathcal{H}'}(s). && \text{(Definition 2.13,} iv)
\end{aligned}
$$

■

Since both algorithms yield the same result for incremental hierarchy specifications, one can use the most appropriate one for a particular $\mathcal{H}$. Moreover, one can first use the incremental algorithm for the incrementing part of a hierarchy specification and then delete the superfluous atoms corresponding to the rest of the specification with the decrementing algorithm. In the sequel we shall only write $\alpha_{\mathcal{H}}$ without assuming a particular algorithm.

3 Arithmetic Reasoning with Decomposed Set-Terms

The atomic decomposition of a set of sets yields an exhaustive and disjoint partitioning such that the cardinalities of the atoms just add up to the cardinality of the decomposed set, cf. Corollary 2.6,*ii*. In this section, we exploit this nice property for embedding *cardinality terms* $|\varphi|$ where φ is a set-term and $|\varphi|$ denotes its cardinality, into a general logic $\mathcal{L}$ with some built-in arithmetic. The integration is quite independent of the particular structure of the logic, which leaves much scope for experimenting with different systems. However, with a particular logic in mind, this section and much of the following is easier to understand. Therefore we recommend that $\mathcal{L}$ is thought of as a system of *non-negative linear Diophantine* equations and inequations. Typical such equations are[10]

$$
\begin{aligned}
2a + 3b &= 5 \\
3c - 5b &\leq 6
\end{aligned}
$$

In the case of *non-negative Diophantine* equations, the variables denote non-negative integers, and this is just what we need for dealing with cardinalities of finite sets. A solution of a set of Diophantine equations maps the variables to non-negative integers. If we generalize from this particular system, we can use the usual logical notions: a set of equations corresponds to a formula, a solution of the set of equations corresponds to an interpretation (mapping of symbols to domain elements), a set without solution corresponds to an *inconsistent* formula, and a set where *all* natural numbers are solutions corresponds to a *tautology*.

On the syntactic level, the cardinality terms are integrated into $\mathcal{L}$ by allowing them to occur at the usual term positions in $\mathcal{L}$-formulas. For example $2|a \cap b| + 3|c \setminus d| - 5e = 1$ could be a well-formed $\mathcal{L}_{\mathcal{S}}$-formula. But we must be careful. It is not allowed to mix the symbols of the two different languages. For example $2|a| + 3a = 5$ makes no sense because the a is interpreted once as set and once as a number. Therefore we require that the signatures of both languages are disjoint. This requirement makes the definition of a combined semantics of both languages very easy. Two interpretations, one of each language, can just be put together, with the cardinality function as a bridge between them.

[10] Inequations can always be eliminated by introducing so-called *slack-variables*: $s \leq n$ becomes $s + x = n$, $s \geq n$ becomes $s - x = n$. Therefore the inequations are not really needed.

Remark: In the rest of this section we shall assume that all sets under consideration are *finite*. In this case, cardinal numbers behave just like ordinary natural numbers. If arbitrary sets were allowed, then the equation $|s|+1=|s|$, for example, would have a model, namely s being an infinite set. In this case, cardinal number arithmetic is needed, which we want to avoid. In the sequel, let $\mathcal{S}$ be a given finite set of basic set-terms and let $\mathcal{H}$ be a given hierarchy specification over $\mathcal{S}$.

Definition 3.1 (Constraint Language) *Let $\mathcal{L}$ be any logic with equality and negation and with a distinguished sort* $\mathbf{N}$ *(for natural numbers) and a distinguished function symbol + (for addition) with the obvious fixed semantics for these symbols. Let $\mathcal{S}$ be a finite set of terms which are* disjoint *to the terms in $\mathcal{L}$.*

We extend $\mathcal{L}$ to $\mathcal{L}_\mathcal{S}$ by allowing cardinality terms $|\varphi|$ over set-terms $S_\mathcal{S}$ to be well-formed terms as well.

The semantics of $\mathcal{L}$ is extended in a straightforward way to a semantics of $\mathcal{L}_\mathcal{S}$: If $\mathcal{E}_\mathcal{L}$ is an interpretation for $\mathcal{L}$ and $\mathcal{E}$ is an interpretation for the set-terms which maps them to finite *sets and which in addition maps $|\ldots|$ to the cardinality function, then $\mathcal{E}_\mathcal{S} \stackrel{\text{def}}{=} \mathcal{E}_\mathcal{L} \cup \mathcal{E}$ is an interpretation for $\mathcal{L}_\mathcal{S}$.* ■

For the logic $\mathcal{L}$ we assume that the usual logical notions apply, i.e. we can speak of unsatisfiable and satisfiable formulas, tautologies, a satisfiability relation $\mathcal{E} \models \varphi$ between interpretations and formulas, and an entailment relation $\varphi \models \psi$ with the usual meaning. In the case of Diophantine equations, $\mathcal{E} \models \varphi$ means, $\mathcal{E}$ is a solution for φ, and $\varphi \models \psi$ means, all solutions for φ are also solutions for ψ.

The language $\mathcal{L}_\mathcal{S}$ is the language for formulating problems involving reasoning about cardinalities of sets. The actual process of inference, however, must take place in $\mathcal{L}$. Therefore, we replace the cardinality terms by the atomic decompositions (cf. (1) $\to$ (2) in the 'children' example in Section 1), and further by the arithmetic terms denoting the sum of their cardinalities (cf. (2) $\to$ (3) in the same example). We could have used new symbols for denoting the cardinalities of the atoms, but since atoms and their cardinalities never get confused, we use the same notation for both.

Definition 3.2 (From Set Cardinalities to Arithmetic Formulas)

i) For a set-term φ with decomposition $\alpha_\mathcal{H}(\varphi) = \{w_1, \ldots, w_k\}$ we define $\Sigma_\mathcal{H}(\varphi) \stackrel{\text{def}}{=} w_1 + \ldots + w_k$.
ii) $\Sigma_\mathcal{H}\{w_1, \ldots, w_k\}$ denotes $w_1 + \ldots + w_k$.
iii) For a $\mathcal{L}_\mathcal{S}$-formula ψ and a decomposition $\alpha_\mathcal{H}$ of set-terms let $\alpha_\mathcal{H}(\psi)$ be the result of replacing all cardinality terms $|\varphi|$ occurring in ψ with $\Sigma_\mathcal{H}(\varphi)$. ■

The translation of an $\mathcal{L}_\mathcal{S}$-formula φ into $\alpha_\mathcal{H}(\varphi)$ involves a small technical problem. For example, if we translate $\varphi \stackrel{\text{def}}{=} |r \cap s| = 2$ with, say, $\alpha_\mathcal{H}(r \cap s) = \{u, v, w\}$ into $u + v + w = 2$, then the atoms u, v, w have changed their meaning. Interpretation $\alpha_\mathcal{H}(\mathcal{E})$ interprets them as sets, whereas in the equation we want them to denote numbers. We could repair this by writing $|u|+|v|+|w| = 2$, which is properly interpreted by $\alpha_\mathcal{H}(\mathcal{E})$. But first this is too clumsy and second, this is not a syntactically correct $\mathcal{L}$-formula. Therefore, we drop the cardinality function and adapt $\alpha_\mathcal{H}(\mathcal{E})$ to deal with these arithmetic formulas properly. $\alpha_\mathcal{H}(\mathcal{E})$ gets an *arithmetic mode* of interpretation: $\alpha^A_\mathcal{H}(\mathcal{E})$ is such that $\alpha^A_\mathcal{H}(\mathcal{E})(w) \stackrel{\text{def}}{=} |\alpha_\mathcal{H}(\mathcal{E})(w)|$ for all atoms w. If, for example, $\alpha_\mathcal{H}(\mathcal{E})(w) = \{a, b, c\}$ then $\alpha^A_\mathcal{H}(\mathcal{E})(w) = 3$. In general it is clear from the context whether we are dealing with an arithmetic formula where $\alpha^A_\mathcal{H}(\mathcal{E})$ has to be used, or not. Therefore it is not necessary to distinguish these two interpretations explicitly. Notice that this is justified because for finite sets, cardinal and ordinal numbers behave in the same way.

Theorem 3.3 (Almost an Equivalence Transformation) *For any $\mathcal{L}_\mathcal{S}$-formula φ and $\mathcal{L}_\mathcal{S}$-interpretation $\mathcal{E}$: If $\mathcal{E} \models \varphi$ then $\alpha^A_\mathcal{S}(\mathcal{E}) \models \alpha_\mathcal{H}(\varphi)$.*[11]

[11] Just to illustrate this statement, suppose φ is $|r \cap s| = 4$ and for the the decomposition let $\alpha_\mathcal{H}(r \cap s) = \{w_1, w_2, w_3\}$.

Proof: By Corollary 2.6,*ii*, we find for each cardinality term $|\psi|$ in φ with decomposition $\alpha_{\mathcal{H}}(\psi) = \{w_1, \ldots, w_k\}$: $\mathcal{E}(|\psi|) = |\mathcal{E}(\psi)| = \alpha_{\mathcal{S}}^A(\mathcal{E})(w_1) + \ldots + \alpha_{\mathcal{S}}^A(\mathcal{E})(w_k) = \alpha_{\mathcal{S}}^A(\mathcal{E})(\alpha_{\mathcal{H}}(\psi))$. Therefore, the interpretation of cardinality terms does not change. By induction on the structure of terms in φ, using cardinality terms as the base cases, one shows that the interpretation of terms does not change. A further induction on the sub-formulas in φ confirms the theorem. ■

This theorem at least gives us a soundness result. If the original formula is satisfiable then the decomposed formula is satisfiable as well. The other direction is not yet clear. In fact, the decomposition is not really an equivalence transformation because in the decomposed formulas all the information about the sets themselves is lost. Therefore, one model for a decomposed formula, where only the cardinalities of the sets are fixed, may correspond to many different models of the original formula. They are only constrained by the cardinalities of the sets, not by the choice of their elements. However, we can show that from an interpretation (solution) for the decomposed formula which assigns numbers to the atoms, sets with suitable cardinalities can be constructed such that the original formula is satisfiable.

Theorem 3.4 (Completeness) *For any $\mathcal{L}_{\mathcal{S}}$-formula φ: If $\alpha_{\mathcal{H}}(\varphi)$ is satisfiable then φ is satisfiable.*

Proof: Let $\mathcal{E} \models \alpha_{\mathcal{H}}(\varphi)$. $\mathcal{E}$ assigns natural numbers to the atoms in $\alpha_{\mathcal{H}}(\top)$. We define $\mathcal{E}'$ to be like $\mathcal{E}$, but instead of natural numbers, it assigns *mutually disjoint* subsets of the integers to the atoms such that if $\mathcal{E}(w) = n$ then $|\mathcal{E}'(w)| = n$. The interpretation of the terms in $\mathcal{S}$ is now determined: For each $s \in \mathcal{S}$: if $\alpha_{\mathcal{H}}(s) = \{w_1, \ldots, w_k\}$ we get $\mathcal{E}'(s) = \mathcal{E}(w_1) \cup \ldots \cup \mathcal{E}(w_k)$. By induction on the structure of set-terms one shows: $|\mathcal{E}'(\psi)| = \mathcal{E}(\alpha_{\mathcal{H}}(\psi))$. Thus, the $\mathcal{E}$-value of all translated cardinality terms in $\alpha_{\mathcal{H}}(\varphi)$ is again identical to the $\mathcal{E}'$-value of the corresponding original terms in φ. Again since nothing else has changed, $\mathcal{E}' \models \varphi$. ■

With the same construction as in Theorem 3.4, we can show that not only satisfiability, but also falsifiability is preserved.

Corollary 3.5 *For any $\mathcal{L}_{\mathcal{S}}$-formula φ: If $\alpha_{\mathcal{H}}(\varphi)$ is falsified by some model then there is a model falsifying φ as well.*

In the sequel we omit the ^{A}-indicator in $\alpha_{\mathcal{S}}^A$. It is always clear from the context, how to interpret atoms.

So far we have a language for talking about cardinalities of sets, and by means of atomic decompositions, we can translate the set cardinality terms into ordinary sum-terms. Any inference system which can deal with addition and with non-negative integer variables can now be used to reason about these sets. In the next section we show how this idea can be put to work. As an example for an application we investigate in detail one particular knowledge representation language. The general method, however, is not restricted to this language. Therefore this section has also a paradigmatic character.

Then $\alpha_{\mathcal{H}}(\varphi)$ is $w_1 + w_2 + w_3 = 4$. Suppose further there is a model $\mathcal{E}$ which maps $r \cap s$ to the set $\{a, b, c, d\}$. $\mathcal{E}$ obviously satisfies $|r \cap s| = 4$ because $|\{a, b, c, d\}| = 4$.

$\alpha_{\mathcal{H}}(\mathcal{E})$ is defined such that it interprets the atoms $\{w_1, w_2, w_3\}$ in a way to form a partitioning of $\{a, b, c, d\}$. For example it may be $\alpha_{\mathcal{H}}(\mathcal{E})(w_1) = \emptyset$, $\alpha_{\mathcal{H}}(\mathcal{E})(w_2) = \{a\}$, $\alpha_{\mathcal{H}}(\mathcal{E})(w_3) = \{b, c, d\}$. ¿From this we get $\alpha_{\mathcal{H}}^A(\mathcal{E})(w_1) = 0$, $\alpha_{\mathcal{H}}^A(\mathcal{E})(w_2) = 1$ and $\alpha_{\mathcal{H}}^A(\mathcal{E})(w_3) = 3$. Thus $\alpha_{\mathcal{H}}^A(\mathcal{E})$ satisfies $w_1 + w_2 + w_3 = 4$.

4 Concept Languages

Concept languages are late descendants of Minski's frames [Min90] and Brachman's KL-ONE [BS85]. They come in a variety of different versions, e.g. ALC [SSS91], CLASSIC [BPS94], KRIS [BH91], LOOM [Mac94], but common to most of them is the separation of a concept language database into a so-called *T-Box* (terminological box) and a so-called *A-Box* (assertional box). The T-Box contains specifications of so-called concept terms and role terms. For example a T-Box formula

$$parent = person \wedge atleast\, 1\; has\text{-}child \tag{10}$$

specifies the concept *parent* as the set of all persons who have at least one child. The role term *has-child* denotes a binary relation. In the concept language jargon, these relations are usually called roles, and we shall, in general, use the technical term 'roles' unless we want to emphasize that it is a binary relation by speaking of 'relations'.

The A-Box on the other hand, contains information about instances of the T-Box concepts. For example, from the A-Box entries $Henry\!: person$, and $Henry\!: has\text{-}child\; Mary$, one can conclude that $Henry$ is an instance of the concept *parent*.

On the T-Box level there are two major reasoning problems. First of all, one wants to know whether a newly introduced concept definition is consistent. For example, if the T-Box contains the two definitions

$$male = person \wedge atleast\, 1\; has\text{-}y\text{-}chromosome \tag{11}$$
$$female = person \wedge atmost\, 0\; has\text{-}y\text{-}chromosome \tag{12}$$

and we add the new definition

$$hermaphrodite = male \wedge female$$

there is no non-empty extension of *hermaphrodite*, which usually indicates errors or misconceptions in the axiomatization of a given domain.

The second inference problem is *subsumption*. If we have (10) in our database and we add

$$grandparent = person \wedge atleast\, 1\; has\text{-}child.parent \tag{13}$$

(grandparents are persons who have at least one child who is a parent) then we can, of course, conclude that all grandparents are parents as well, i.e. $grandparent \subseteq parent$. Subsumption relations are usually very useful for structuring a knowledge base. Finding out all subsumption relations between all concepts is called *classification*, and this is the basic operation of all T-Boxes.

The standard semantics of concept languages allows one to translate all T-Box and A-Box information into first-order predicate logic (FOL). Therefore concept languages are essentially fragments of FOL, and since most of them are decidable, they represent proper fragments that are more expressive than propositional logic. Much effort has been invested in recent years to explore the borderline between propositional logic and FOL by investigating various versions of concept languages, see [DLNN95] for a good summary of recent results.

In this paper, we want to emphasize the treatment of set-cardinalities. Therefore we consider a relatively weak concept language which has no negation, no disjunction, and no existential quantification, just conjunction, universal quantification and, in the basic version, number restrictions of the kind $atleast\, n\, r$ and $atmost\, n\, r$, and as abbreviation, $exactly\, n\, r$, see [Neb90]. In addition, there can be so-called *role hierarchies*. For example, $has\text{-}son \subseteq has\text{-}child$ expresses that the *has-son*-relation is a sub-relation of the *has-child*-relation. Typical T-Box declarations of this language are (11, 12). The concept term (13) is not a valid declaration because we do not yet consider qualified number restrictions of the kind $atleast\, n\, r.c$.

In spite of the restricted expressivity of the language (though subsumption is co-NP-hard [Neb90] and satisfiability is PSPACE-complete [DLNN95]), one can formalize a number of interesting features.

For example, the concept terms (11) and (12) together state that $male$ and $female$ are disjoint concepts. There are no hermaphrodites. A concept term $\forall\, has\text{-}child.(male \wedge female)$ would enforce that the *has-child*-relation is empty, which makes this term equivalent to $atmost\, 0\; has\text{-}child$.

More interesting things can be expressed with role hierarchies. If we specify both *has-son* and *has-daughter* as sub-roles of *has-child* then $atleast\ 2\ has\text{-}son \wedge atleast\ 3\ has\text{-}daughter \wedge \forall\, has\text{-}son.male \wedge \forall\, has\text{-}daughter.female$ together with (11) and (12) implies $atleast 5 has\text{-}child$. This example shows that addition of numbers is implicitly encoded in the concept terms. Subtraction may occur as well: $atmost 5 has\text{-}child \wedge atleast 2 has\text{-}son \wedge \forall\, has\text{-}son.male \wedge \forall\, has\text{-}daughter.female$ together with (11) and (12) implies $atmost\, 3\; has\text{-}daughter$.

None of the known inference procedures for these kind of languages does the inferences by evaluating arithmetical terms. The standard tableaux approach generates sets of witnessing constants, counts the length of lists, and makes a lot of case distinctions, see [OSH95, HNSS90]. For example, if it hits the number restriction in $city = place \wedge atleast 1,000,000 has\text{-}inhabitant$ it would generate one million constants, one for each inhabitant. Translating such a term into predicate logic is even worse. The FOL version of $atleastnr$ is $\exists x_1 \ldots x_n\; x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \ldots \wedge x_{n-1} \neq x_n \wedge r(a, x_1) \wedge \ldots \wedge r(a, x_n)$ (where a is the current focus variable).

The atomic decomposition approach, on the other hand, solves this problem in the most adequate way by reasoning with the numbers directly. The sets we are going to decompose are not the concepts themselves, but the so called *role fillers* or *role successors*. Consider the definition of *parent* in concept term (10). It specifies that each parent must have at least one child. Given a parent, say Jack, then there is at least one object in the *has-child*-relation, which is Jack's *role filler* for the *has-child*-relation. The set of the *has-child*-role fillers has in this case cardinality ≥ 1, i.e. $|has\text{-}child| \geq 1$ is an inequation, which we can extract from (10) and which is subject to the decomposition method.

In the following, let us demonstrate the idea with a more complicated example.

4.1 The PCM-Example

First of all we introduce three concept names P (physicists), C (chemists), and M (mathematicians). We do not say much about them, but we want to make sure that nobody can be a physicist, a chemist, and a mathematician at the same time. Two qualifications, however, are allowed. That means P, C, and M are axiomatized in such a way that $P \wedge C \wedge M$ is inconsistent. See the Appendix (Section 6) for details of the axiomatization.

We introduce four roles es (employs scientist), ep (employs physicist), ec (employs chemist), and em (employs mathematician). The role hierarchy is such that ep, ec, and em are all sub-roles of es, i.e. $\mathcal{H} = \{es, ep \subseteq es, ec \subseteq es, em \subseteq es\}$ (cf. Definition 2.7). Now we axiomatize an institute I as

$$I \stackrel{\text{def}}{=} exactly\, 3\, ep \wedge exactly\, 2\, em \wedge atmost\, 4\, es \wedge\ \forall\, ep.P \wedge \forall\, ec.C \wedge \forall\, em.M \tag{14}$$

Taking the role hierarchy into account, we find the following decomposition of the roles $\{es, ep, ec, em\}$: (see Figure 6)

$$\begin{aligned} \alpha_{\mathcal{H}}(es) &= \{s, p, c, m, pc, pm, cm, pcm\} \\ \alpha_{\mathcal{H}}(ep) &= \{p, pc, pm, pcm\} \\ \alpha_{\mathcal{H}}(ec) &= \{c, pc, cm, pcm\} \\ \alpha_{\mathcal{H}}(em) &= \{m, pm, cm, pcm\} \end{aligned}$$

In a similar way as our previous name convention, s denotes the set of all employed scientists who are neither physicists, chemists, nor mathematicians. With pm we denote the set of employed physicists who are mathematicians, but not chemists, in pcm are employed scientists who have all three of the known professions, etc.

The first thing to check is whether any of the intersections of the roles are empty. To this end we have to examine all subsets of the universal quantifications and recursively test whether

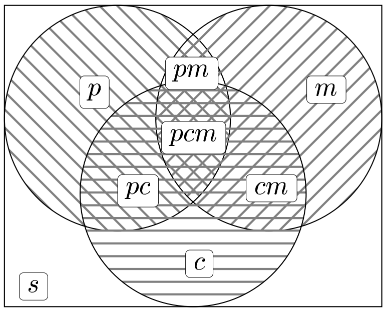


Figure 6: The decomposition of es, ep, em, ec.

the conjunction of their arguments is inconsistent. And in fact, in this case we assumed that the conjunction $P \wedge C \wedge M$ is inconsistent. Therefore the intersection of the corresponding roles $ep \cap ec \cap es$ is empty. The decomposition of this set-term is just pcm and thus we can set $pcm = 0$. The translation of the number restrictions in Definition (14) yields the following inequations:

$$\begin{aligned} p + pm + pc &= 3 \\ m + pm + cm &= 2 \\ s + p + c + m + pc + pm + cm &\leq 4 \end{aligned} \tag{15}$$

These three equations have the following solutions:

s	p	m	c	pm	pc	cm	ep	em	ec	es
0	2	1	0	1	0	0	3	2	0	4
0	2	0	0	1	0	1	3	2	1	4
0	1	0	0	2	0	0	3	2	0	3
0	1	1	0	1	1	0	3	2	1	4
0	1	0	0	1	1	1	3	2	2	4
0	0	0	0	2	1	0	3	2	1	3
0	1	0	0	1	2	0	3	2	2	4
0	0	0	0	1	2	1	3	2	3	4
1	0	0	0	2	1	0	3	2	1	4
1	1	0	0	2	0	0	3	2	1	4

The columns on the right list the number of employed physicists (ep), mathematicians (em), chemists (ec), and scientists at all (es). These numbers are computed according to the above decomposition. ¿From these solutions, we can read off for example that *atmost* $3\, ec$ must be true, i.e. there are at most three employed chemists and $c + pc + pcm \leq 3$ is therefore a consequence of (15). The formal proof can be found in the Appendix, Section 6.

4.2 The Language $\mathcal{TF}^{++}$

The concept language we have been investigating for this paper is the T-Box language $\mathcal{TF}^{++}$, which is an extension of the language $\mathcal{TF}$ [Neb90]. $\mathcal{TF}$ itself has only conjunction, the universal quantifier, the number restrictions *atleast* and *atmost*, but allows for a tree-like role hierarchy.

In $\mathcal{TF}^{++}$ we can allow for arbitrary role hierarchy specifications in the sense of Definition 2.7. Moreover, we can allow for roles denoted by set-terms according to Definition 2.4 built with $\cap$, $\cup$, and $\setminus$, which yields the first "+" in $\mathcal{TF}^{++}$.

As a second generalization yielding the second "+" in $\mathcal{TF}^{++}$, we have replaced the number restrictions *atleast* and *atmost* by general arithmetic constraints expressed in the language $\mathcal{L}_\mathcal{R}$ (Definition 3.1) where $\mathcal{R}$ is the set of role names. Since the role names are to be decomposed in our setting, $\mathcal{R}$ plays the role of $\mathcal{S}$ from the previous section. The language $\mathcal{L}_\mathcal{R}$ for expressing arithmetic constraints is actually a parameter to our system. One can make it as strong as the arithmetic solver of the system can handle.
Simple concepts, which can be expressed in $\mathcal{TF}^{++}$, but not in $\mathcal{TF}$ are

$$\textit{male-dominant-parent} \stackrel{\text{def}}{=} \textit{parent} \wedge |\textit{has-son}| \geq |\textit{has-daughter}|$$

denoting the set of all parents who have more sons than daughters, $|\textit{has-son}| \geq |\textit{has-daughter}|$ is an expression of the language $\mathcal{L}_\mathcal{R}$.
As a further extension we include so-called *functional roles* with number values. Again an example:

$$500\textit{er} \stackrel{\text{def}}{=} \textit{car} \wedge \textit{cubic-capacity} = 500 \cdot |\textit{has-cylinder}| \tag{16}$$

denotes the set of all cars who have a cubic capacity of 500 cm^3 per cylinder. The term *has-cylinder* is an ordinary role, i.e. a relation between cars and cylinders, which denotes the number of cylinders in the arithmetic context. *cubic-capacity* on the other hand is a function from cars to numbers. Number valued functional roles can only occur in the arithmetic context. For the arithmetic solver it is just another variable. Each solution of a constraint gives some value to these functional roles which is consistent with the constraints, therefore their treatment comes for free.

Definition 4.1 ($\mathcal{TF}^{++}$ Concept Terms) *Let $\mathcal{R}$ be a set of symbols (role names) and $\mathcal{C}$ another set of symbols (concept names), disjoint to $\mathcal{R}$. Let $\mathcal{L}_\mathcal{R}$ be a suitable arithmetic language according to Definition 3.1. The set of* concept terms *over $\mathcal{R}$ and $\mathcal{C}$ is the smallest set such that*

i) a concept name is a concept term,
ii) if φ and ψ are concept terms then '$\varphi \wedge \psi$' is a concept term,
iii) if $r \in S_\mathcal{R}$ is a set-term (from now on called role term*), φ is a concept term, then '$\forall r.\varphi$' is a concept term, and*
iv) all formulas of $\mathcal{L}_\mathcal{R}$ (Definition 3.1) are concept terms. ■

A T-Box is usually defined as a set of terminological axioms of the kind $c = \varphi$ or $c \subseteq \varphi$. If these definitions are cycle-free, i.e. a definition $c = \varphi$ of a new concept c does not refer directly or indirectly to itself, one can always expand the definitions. An individual consistency or subsumption test therefore need not deal with terminological axioms, but just with concept terms. Since this is possible for $\mathcal{TF}^{++}$ as well, we do not introduce terminological axioms on a formal level[12].

Notation: For two concept terms φ and ψ, $\varphi \in \psi$ means ψ contains φ as a top level conjunct, i.e. $\psi = \ldots \wedge \varphi \wedge \ldots$.

The semantics for our language $\mathcal{TF}^{++}$ is basically the usual semantics for concept languages which interprets concept terms as subsets of some domain.

Definition 4.2 (Interpretation of Concept Terms) *An interpretation $\mathcal{E} = (\mathcal{D}, \Im)$ for concept terms and a given hierarchy specification $\mathcal{H}$ over some set $\mathcal{R}$ of role names consists of a set $\mathcal{D}$ and an $\mathcal{L}$-interpretation $\Im$. $\Im$ consists of two parts, $\Im_\mathcal{L}$ which interprets the pure $\mathcal{L}$-symbols (except the functional roles), and $\Im_C$ which interprets the main $\mathcal{TF}^{++}$parts. In particular, $\Im_C$ assigns*

i) a subset of the domain $\mathcal{D}$ to each concept name in $\mathcal{C}$,

[12] A remark for modal logicians. There is a straightforward correspondence between T-Box languages like $\mathcal{TF}^{++}$ and multi-modal logic. The universal quantification $\forall r.\varphi$ corresponds to a parameterized box operator $[r]\varphi$ in a multi-modal logic like K_m. Since $\mathcal{TF}^{++}$ has no existential quantifier, which would correspond to a parameterized diamond operator, and it has neither negation nor disjunction, the non-arithmetic part of $\mathcal{TF}^{++}$ corresponds to a fragment of K_m. The arithmetic part has some correspondence with the logic of graded modalities, but our arithmetic language is different to the language used for graded modalities.

ii) disjoint binary relations on $\mathcal{D}$ to each atom in $\alpha_{\mathcal{H}}(\top)$ together with the corresponding binary relations for the role names themselves and for the role terms,

iii) a function from domain elements to numbers for each number-valued functional role, and

iv) the usual interpretation to $\cap$, $\cup$, $\setminus$ and $|...|$.

If for some $a \in \mathcal{D}$ we define $\Im_a$ to be like $\Im$, but for role names $r \in \mathcal{R}$ and atoms $a \in \alpha_{\mathcal{H}}(\top)$ we require $\Im_a(r) \stackrel{\text{def}}{=} \{b \mid (a,b) \in \Im_C(r)\}$ and for number valued functional roles f we require $\Im_a(f) = \Im_C(f)(a)$ then $\Im_a$ is turned inductively into an $\mathcal{L}_{\mathcal{R}}$ interpretation for interpreting $\mathcal{L}_{\mathcal{R}}$ formulas in the sense of Definition 3.1. Let $\mathcal{E}_a \stackrel{\text{def}}{=} (\mathcal{D}, \Im_a)$.

An interpretation is turned into an interpretation for concept terms in the usual way:

$$\begin{array}{lcll} \mathcal{E}(c) & = & \Im(c) & \textit{if } c \textit{ is a concept name} \\ \mathcal{E}(\varphi \wedge \psi) & = & \mathcal{E}(\varphi) \cap \mathcal{E}(\psi) & \\ \mathcal{E}(\forall\, r.\varphi) & = & \{a \in \mathcal{D} \mid \textit{for all } b\text{: } (a,b) \in \Im(r) \textit{ implies } b \in \mathcal{E}(\varphi)\} & \\ \mathcal{E}(\varphi) & = & \{a \in \mathcal{D} \mid |\Im_a(r)| \textit{ is finite for all } r \textit{ occurring in } \varphi \textit{ and } \Im_a \models \varphi\} & \\ & & & \textit{if } \varphi \in \mathcal{L}_{\mathcal{R}}. \end{array}$$

For two concept terms φ and ψ we define:
$\varphi \models \psi$ iff $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ for all *interpretations $\mathcal{E}$.* ■

We call an interpretation $\mathcal{E}$ a *model for* a concept term φ if $\mathcal{E}(\varphi) \neq \emptyset$. In this case φ is called *consistent* or *satisfiable.* If $\mathcal{E}(\varphi) = \emptyset$ for all interpretations $\mathcal{E}$ then φ is *inconsistent* or *unsatisfiable.*

It is important to understand the definition of $\Im_a$. For a role name r, $\Im$ maps r to a binary relation, i.e. $\Im(r) \subseteq \mathcal{D} \times \mathcal{D}$. For a domain element $a \in \mathcal{D}$, $\Im_a(r)$ is the set of *r-successors* of a (in the KL-ONE jargon also called the role fillers of a for r). Similarly for a functional role f, $\Im_a(f)$ is just the value of f on a.

So we dealt with the basic parts of the $\mathcal{L}_{\mathcal{R}}$-formulas. However, we use $\Im_a$ also for interpreting complex arithmetic formulas. To this end, we assume $\Im_a$ to be accompanied with an interpretation for $\mathcal{L}_{\mathcal{R}}$-formulas in the sense of Definition 3.1. For example if $\varphi \stackrel{\text{def}}{=} |r| + |s| = 3$ and $\Im_a$ maps r to the set $\{a\}$ and s to the set $\{b, c\}$ then $\Im_a(|r|) = 1$ and $\Im_a(|s|) = 2$. Assuming that $\Im_a$ understands the $+$ sign properly, we find $\Im_a(|r| + |s|) = 3$, and thus $\Im_a \models |r| + |s| = 3$.
To make the semantics clearer, let us consider a more complicated example (the role *age* is functional):

$$\begin{array}{lcl} \mathcal{R} & = & \{\textit{has-child}, \textit{has-daughter}, \textit{has-son}\}, \\ \mathcal{H} & = & \{\textit{has-daughter}, \textit{has-son} \lhd \textit{has-child}\}, \\ \varphi & = & \textit{person} \wedge \textit{age} \geq 30 \wedge |\textit{has-child}| \geq 2 \cdot |\textit{has-daughter}| + 1 \wedge \\ & & \forall\, \textit{has-daughter}.\textit{teacher}. \end{array}$$

We abbreviate $\alpha_{\mathcal{H}}(\top)$ with $\{d, s\}$ (d for $\{\textit{has-child}, \textit{has-daughter}\}$ and s for $\{\textit{has-child}, \textit{has-son}\}$ reflecting the semantics of a partioning declaration).

We construct an interpretation for φ. First of all, we have to choose a domain $\mathcal{D}$. Say, $\mathcal{D} \stackrel{\text{def}}{=} \{Jack, George, Henry, Paul, Mary, BMW520\}$. The interpretation $\Im_{\mathcal{L}}$ interprets the arithmetic parts in the standard way:

$$\begin{array}{rcl} \Im_{\mathcal{L}}(1) & = & \text{natural number 1 etc.,} \\ \Im_{\mathcal{L}}(+) & = & \text{addition function,} \\ \Im_{\mathcal{L}}(\cdot) & = & \text{multiplication function,} \\ \Im_{\mathcal{L}}(\geq) & = & \text{greater or equal predicate,} \\ \Im_{\mathcal{L}}(|\ldots|) & = & \text{cardinality function.} \end{array}$$

We have two concept names, *person* and *teacher*. Let us choose

$$\begin{array}{rcl} \Im_C(\textit{person}) & = & \{Jack, George, Henry, Paul, Mary\} \\ \Im_C(\textit{teacher}) & = & \{Mary\} \end{array}$$

The two atoms s and d of the decomposition of the roles are interpreted as follows:

$$\begin{aligned}\Im_C(s) &= \{(Jack, George), (Jack, Henry), (George, Paul)\}\\ \Im_C(d) &= \{(Jack, Mary)\}\end{aligned}$$

¿From this we obtain

$$\Im_C(\textit{has-child}) = \{(Jack, George), (Jack, Henry), (George, Paul), (Jack, Mary)\}$$

and the interpretations of *has-son* and *has-daughter* are like s and d. Moreover we get as role fillers

$$\begin{aligned}\Im_{Jack}(s) &= \{George, Henry\}\\ \Im_{Jack}(d) &= \{Mary\}\\ \Im_{George}(s) &= \{Paul\}\\ \Im_{George}(d) &= \emptyset\end{aligned}$$

and the role fillers for all other individuals are empty as well. For the functional role *age* we choose

$$\begin{aligned}\Im_C(age) = \{&(Jack, 70), (George, 45), (Henry, 50),\\ &(Paul, 20), (Mary, 50), (BMW520, 1)\}.\end{aligned}$$

We now can evaluate $\mathcal{E}(\varphi)$
$= \mathcal{E}(person) \cap \mathcal{E}(age \geq 30) \cap \mathcal{E}(|\textit{has-child}| \geq 2 \cdot |\textit{has-daughter}| + 1) \cap$
$\mathcal{E}(\forall\, \textit{has-daughter.teacher})$
$= \{Jack, George, Henry, Paul, Mary\} \cap \{a \in \mathcal{D} \mid \Im_C(age)(a) \geq 30\} \cap$
$\{a \in \mathcal{D} \mid \Im_a \models (|\textit{has-child}| \geq 2 \cdot |\textit{has-daughter}| + 1)\} \cap$
$\{a \in \mathcal{D} \mid$ for all b: $(a, b) \in \Im(\textit{has-daughter})$ implies $b \in \mathcal{E}(teacher)\}$
$= \{Jack, George, Henry, Paul, Mary\} \cap \{Jack, George, Henry, Mary\} \cap$
$\{a \in \mathcal{D} \mid \Im_a \models (|s| + |d| \geq 2 \cdot |d| + 1)\} \cap$
$\{Jack, George, Henry, Paul, Mary, BMW520\}$[13]
$= \{Jack, George\}$
This is because $\Im_{Jack}(|s|) = 2$, $\Im_{Jack}(|d|) = 1$ and therefore $2 + 1 \geq 2 \cdot 1 + 1$ is true. Also $\Im_{George}(|s|) = 1$, $\Im_{George}(|d|) = 0$, and therefore $1 + 0 \geq 0 + 1$ is true as well.

4.3 Optimized Decomposition of Role Hierarchies

The decomposition of role hierarchies is only feasible if the number of atoms can be kept reasonably small, which so far is not the case for realistic T-Boxes with hundreds or thousands of roles. Even our incremental decomposition algorithm would generate far too many atoms except for trivial cases, cf. Definition 2.13. A significant reduction of the number of atoms could be achieved if enough information about disjointness of roles would be available. For example, in every reasonable axiomatization two roles like *has-child* and *has-car* will be disjoint, which makes all intersecting atoms empty. In fact, most pairs of roles which have no common sub- or super-role in a role hierarchy are usually disjoint. We could require the user of such a system to specify all disjointness relations for roles and use this information to compute an optimized decomposition.

But it turns out that there is a more elegant way. We can show that whenever there is a model at all for a concept term, then there is also a model where the roles, which are not in the same connected component of a role hierarchy graph[14], are disjoint. Therefore the connected components of a role

[13] $\forall$ *has-daughter.teacher* holds trivially for those who do not have daughters at all.

[14] The sub-role relation in a role hierarchy specification, given by the explicit subset declaration and the partitioning declaration can be interpreted as the (undirected) edges in a graph with role names as vertices. The connected components of this graph are meant.

hierarchy can be decomposed separately and no overlapping atoms are generated. In the extreme case, where the role hierarchy is flat, this optimized decomposition yields just one single atom per role. The number of atoms now depends primarily on the structure of the connected components in a role hierarchy, the narrower the better, and only linearly on the number of connected components.

In order to prove this statement, we must first prove the finite model property for $\mathcal{TF}^{++}$. The finite model property states that, whenever there is a model at all for a concept term, there is one with a finite domain. Since $\mathcal{TF}^{++}$ is very similar to standard modal logics, for which the finite model property holds, it is quite likely that this property holds for $\mathcal{TF}^{++}$ as well. But due to the arithmetic part in $\mathcal{TF}^{++}$, things are slightly more tricky than in modal logics. For example, if we would allow arbitrary sets as role fillers, then a concept term like $|s| + 1 = |s|$ would only be satisfiable in an infinite model. But we have excluded this case in the semantics of arithmetic formulas by requiring that the role fillers be finite sets.

Theorem 4.3 (Finite Model Property) *Given a role hierarchy $\mathcal{H}$ over a set of role names $\mathcal{R}$, each satisfiable $\mathcal{TF}^{++}$concept term φ has a model with finite domain.*

Proof: Since φ is satisfiable, there is a model $\mathcal{E} = (\mathcal{D}, \Im)$ with $x_0 \in \mathcal{E}(\varphi)$ for some $x_0 \in \mathcal{D}$. From $\mathcal{E}$ we construct a finite model using selective filtration [Gab70, Gab72], a variant of Lemmon and Scott's filtration [LS66, Che80]. Let Φ be the set of all concept-sub-terms[15] in φ. Since φ is finite, Φ is finite as well. We now define an equivalence relation over $\mathcal{D}$

$$\forall x, y \in \mathcal{D}: \ x \equiv y \text{ if and only if } \forall \psi \in \Phi : x \in \mathcal{E}(\psi) \Leftrightarrow y \in \mathcal{E}(\psi).$$

Thus, two elements of the original domain belong to the same equivalence class if and only if they satisfy exactly the same subformulas of φ. Since there are at most 2^n subsets in Φ if φ has n concept-sub-terms, we obtain finitely many equivalence classes. One can take either the set of equivalence classes as the domain of the finite model (filtration), or choose one or more representatives of each equivalence class (selective filtration). In our case we have to satisfy the numeric constraints about the number of r-successors for each role. Therefore it is necessary to choose some representatives of each equivalence class. Given an equivalence class X, let $AR(X)$ denote the set of *arithmetic* terms $\varphi_t \in \Phi$ which are satisfied by $\mathcal{E}$ for some (and thus by all) $x \in X$. If $AR(X)$ is not empty, then the number of w-role fillers for $x \in X$ is finite for all atomic roles w occurring in $AR(X)$, according to the '$\varphi \in \mathcal{L}_\mathcal{R}$' case in Definition 4.2. Therefore we can choose from each equivalence class a minimum and finite number of representatives such that the restriction of the roles to the new submodel satisfies the arithmetic constraints in $AR(X)$[16]. One of the elements to be chosen is x_0. Overall we get a finite submodel $\mathcal{E}'$ of $\mathcal{E}$.
By induction on the structure of the formulas in Φ we can easily verify that for each element in the submodel exactly the same formulae of Φ hold in the submodel as in the original model. For the concept names and the universal quantifier it is trivial, and for the arithmetic formulae it holds by construction. Since $\mathcal{E}'$ is a proper submodel of $\mathcal{E}$, it is also guaranteed that $\mathcal{E}'$ satisfies the hierarchy specification $\mathcal{H}$. Thus, $x_0 \in \mathcal{E}'(\varphi)$. ■

Theorem 4.4 (Disjointness Theorem) *Let $\mathcal{H}$ be a role hierarchy specification for some roles $\mathcal{R}$ and φ be a satisfiable concept term. If for two role names r and s, there is neither a common sub-role nor a common super-role in the role hierarchy, and neither $r \cup s$ nor $r \cap s$ occur in φ explicitly or implicitly, then there is always a 'disjoint' model, i.e. a model where r and s are disjoint relations.*

Proof: If φ is satisfiable, then there is a model $\mathcal{E} = (\mathcal{D}, \Im)$ with finite domain and $x \in \mathcal{E}(\varphi)$ for some $x \in \mathcal{D}$, according to Theorem 4.3. First of all we need to make this model irreflexive by copying the

[15] If for example φ is $|\textit{has-daughter}| \geq |\textit{has-son}|$, then *has-daughter* is a sub-term, but it is *not* a concept-sub-term. The formula $|\textit{has-daughter}| \geq |\textit{has-son}|$ itself is the only concept-sub-term.

[16] For example if there are two equivalence classes X_1 and X_2 and $AR(X_1) = \{|r| = 2\}$ then we might need to choose two elements in X_2 to get enough r-successors for a representative in X_1.

reflexive elements. That means if there is some $(X,X) \in \mathcal{E}'(r)$, we copy X once together with all outgoing edges. Schematically this might look as follows:

This operation does not change the number of role fillers. Therefore the irreflexive model still satisfies φ. We can also assume that $\mathcal{E}$ is generated by x.[17]
For the main manipulation, an algorithm $\mathsf{copy}[G,e,v]$ is defined. It takes a graph $G=(V,E)$ with a set of edges E and a set of vertices V, an edge $e=\langle x,v,r\rangle \in E$ (with x, v being the start and end vertex of edge e, respectively and r being the atomic role the interpretation of which is e), and a vertex $v \in V$ as input and constructs a graph $G'=(V',E')$ by copying the vertex v with set of in-going edges I_v and set of outgoing edges O_v in the following way:

$\mathsf{copy}[G,e,v]$:

1.	V'	$=$	$V \cup \{v_c\}$ with v_c being a copy of vertex v.
2.	O_{v_c}	$=$	$\{\langle v_c,y,r\rangle \mid \langle v,y,r\rangle \in O_v\}$, i.e. all outgoing edges of v are copied to v_c.
3.	I_{v_c}	$=$	$\{\langle x,v_c,r\rangle\}$, i.e. $e=\langle x,v,r\rangle$ is copied as the only in-going edge of v_c.
4.	E'	$:=$	$E \cup I_v \cup O_{v_c}$.

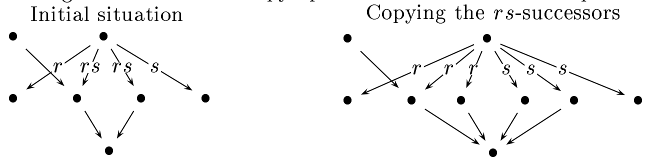
We now construct a new model $\mathcal{E}'$ by separating the common r and s successors for each domain element. Starting with x we work through the graph layer by layer. (The layers are defined by the interpretations of the roles.) Each time we meet some domain element y which has common r- and s-successors, we use the copy-operation to get a fresh copy of the common r- and s-successors. The interpretation of the role s is now changed such that the new copy is made the set of s-successors, and the original elements are no longer s-successors, but just r-successors[18]. Since $\mathcal{D}$ is finite and the copy operation introduces in layer m only finitely many new elements of an already existing layer $m+1$, it eventually terminates. In the new model $\mathcal{E}'=(\mathcal{D}',\Im')$ there are no common r- and s-successors. In $\mathcal{E}'$ we define for concept names c; $x' \in \Im'(c)$ iff $x \in \Im(c)$ and $x=x'$ or x' is a copy of x. Compared to the original model $\mathcal{E}$, in $\mathcal{E}'$ only the number of role successors for the union or intersection of r and s has changed. But according to our assumption we do not have the syntactic means for referring to this union and intersection. Therefore, for all role terms occurring in $\mathcal{H}$ and φ, the number of role successors has not changed. By induction on the structure of φ one can now easily prove $x \in \mathcal{E}'(\varphi)$. ■

This theorem has immediate implications for the consistency test for concept terms and for the subsumption test. Its contraposition states that if there is no disjoint model for a concept term, there is no model at all for it. That means only disjoint models need to the tested in the consistency test.

[17] In $\mathcal{TF}^{++}$ we have no operators moving backwards along the roles. Therefore for each model and each domain element x one can eliminate all other domain elements which cannot be accessed from x by a finite number of transitions along the role edges. This is the x-generated sub-model. The proof is essentially the same as for multi-modal logic K_m.

[18] We again illustrate the copy operation with a small example.

Initial situation | Copying the rs-successors



Notice that the incoming edge from the top left node is not copied.

The corresponding argument for the subsumption test is a bit more sophisticated. Suppose $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ is to be tested for *all* models $\mathcal{E}$, and we have verified $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ only for disjoint models. Take an arbitrary model $\mathcal{E}$ with $x \in \mathcal{E}(\varphi)$ for some x. By the construction of Theorem 4.4 we have $x \in \mathcal{E}'(\varphi)$ for the filtrated (with $\Phi = \varphi \wedge \psi$) and copied model $\mathcal{E}'$. Since $\mathcal{E}'$ is a disjoint model, $x \in \mathcal{E}(\psi)$ holds as well. Due to the construction of the filtrated and copied model, $x \in \mathcal{E}(\psi)$ can be proved with structural induction. Thus, $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ holds for all models.

This allows us to treat roles which occur in different connected components of a role hierarchy graph as disjoint roles, provided in the concept terms themselves there are no role terms connecting them in some way, either directly or indirectly via the role hierarchy. Technically we can exploit this result by decomposing connected components of the role hierarchy separately. In this way, the generation of exponentially many intersection atoms is avoided. But notice that when for example $r \cap s$ occurs in φ then the connected components of r and of s are no longer separated. They must be decomposed together. This should be a rare exception. Nothing reasonable can be expressed with a role term like *has-child* $\cap$ *has-car* for example.

4.4 Reasoning with Concept Terms

A concept term can be a tautology if it evaluates to the whole domain for all interpretations. Tautologous concept terms are usually redundant. They indicate that there is something wrong in the knowledge base. In the next theorem we show that a test for tautologies can essentiall be reduced y to a test on the decomposed arithmetic part of the concept term.

Theorem 4.5 (Tautology) *A concept formula $\varphi \stackrel{\text{def}}{=} C \wedge N \wedge U$ where C are the concept names, N are the arithmetic formulas and U are the universal quantifications, is a tautology (written $\models \varphi$), i.e. $\mathcal{E}(\varphi)$ is the whole domain for all interpretations $\mathcal{E}$, iff*
i) C is empty and
ii) $\alpha_{\mathcal{H}}(N)$ (Definition 3.2,iii) is a tautology and
iii) for all '$\forall r.\psi$' $\in U$: ψ is a tautology.

Proof: If C is not empty we can certainly find an interpretation where C is a proper subset of the domain. If $\alpha_{\mathcal{H}}(N)$ is not a tautology, there is a falsifying model for $\alpha_{\mathcal{H}}(N)$. By Corollary 3.5 there is a falsifying interpretation $\mathcal{E}'$ for N as well. We now construct an interpretation $\mathcal{E} = (\mathcal{D}, \Im)$ for φ where $\mathcal{D} = \{a\} \cup \{\Im'(w) \mid w \in \alpha_{\mathcal{H}}(\top)\}$ for some arbitrary element a and for all atoms w: $\Im(w) \stackrel{\text{def}}{=} \{(a, b) \mid b \in \mathcal{E}'(w)\}$. This means the w-successors of a are just those elements assigned to w by $\mathcal{E}'$. This way we get an interpretation for concept terms such that $\Im_a = \mathcal{E}'$ and $\Im_a$ falsifies N. Therefore $a \notin \mathcal{E}(\varphi)$ and φ is not a tautology.
If for some '$\forall r.\psi$' $\in U$: ψ is not a tautology, there is an interpretation $\mathcal{E}$ and some domain element $b \notin \mathcal{E}(\psi)$. In the same way as in the previous case we construct an interpretation $\mathcal{E}'$ with b as r-successor of some new element a. Then it is not the case that for all r-successors of a, ψ holds, which means that φ is not a tautology. ∎

For both, the consistency check and the subsumption test, we need a certain normal-form for the concept terms. In this normal-form the universal quantifications are decomposed into atomic components as well. We can illustrate this with the PCM example from Section 4.1. There we had the roles *ep* (employed physicist), *ec* (employed mathematician), and *ec* (employed chemist). The decomposition was $\alpha_{\mathcal{H}}(ep) = \{p, pc, pm, pcm\}$, $\alpha_{\mathcal{H}}(ec) = \{c, pc, cm, pcm\}$, and $\alpha_{\mathcal{H}}(em) = \{m, pm, cm, pcm\}$. In Definition (14) we had the quantifications $\forall ep.P \wedge \forall ec.C \wedge \forall em.M$. Since the decomposition splits the roles into mutually disjoint parts, we can split the quantifications into separate parts as well. In the case of the PCM-example we get

$$\begin{array}{ll} \forall ep.P & \text{becomes } \forall p.P \wedge \forall pc.P \wedge \forall pm.P \wedge \forall pcm.P \\ \forall ec.C & \text{becomes } \forall c.C \wedge \forall pc.C \wedge \forall cm.C \wedge \forall pcm.C \\ \forall em.M & \text{becomes } \forall m.M \wedge \forall pm.M \wedge \forall cm.M \wedge \forall pcm.M \end{array}$$

Now we collect the quantifications over the same atomic component into one term. For example $\forall\, pc.P$ and $\forall\, pc.C$ is pulled together into $\forall\, pc.(P \wedge C)$. For the three quantifications $\varphi = \forall\, ep.P \wedge \forall\, ec.C \wedge \forall\, em.M$ we obtain through this rearrangement: $\forall\, p.P \wedge \forall\, c.C \wedge \forall\, m.M \wedge \forall\, pc.(P \wedge C) \wedge \forall\, pm.(P \wedge M) \wedge \forall\, cm.(C \wedge M) \wedge \forall\, pcm.(P \wedge C \wedge M)$.

The advantage of this decomposition is that the atomic parts are now independent of each other. One atomic component like $\forall\, pc.(P \wedge C)$ has no implications to any other role successors. This is not the case for the original terms. A quantification like $\forall\, ep.P$ may well be relevant for some employed chemist, who happens to be an employed physicist as well. The precise definition of the decomposition is given below.

Definition 4.6 (Decomposition of Universal Quantifications)
If $\varphi = \bigwedge_{r \in R} \forall\, r.\varphi_r$ *is a* universal concept term *where* R *is a set of role terms we define*

$$\alpha_{\mathcal{H}}(\varphi) \stackrel{\text{def}}{=} \bigwedge_{w \in \bigcup_{r\in R} \alpha_{\mathcal{H}}(r)} \forall\, w.(\bigwedge_{w \in \alpha_{\mathcal{H}}(r)} \varphi_r).$$

For an atom w *let*

$$\alpha_{\mathcal{H}w}(\varphi) \stackrel{\text{def}}{=} \bigwedge_{w \in \alpha_{\mathcal{H}}(r)} \varphi_r.$$

■

In the PCM example above we would have for the atom pm: $\alpha_{\mathcal{H}pm}(\varphi) = P \wedge M$.

The next theorem shows that the decomposition of quantifications is in fact an equivalence transformation, given the proper interpretation of the atoms.

Lemma 4.7 (Soundness and Completeness for Definition 4.6) *For all role terms* r*, universal concept terms* $\varphi = \bigwedge_{r \in R} \forall\, r.\varphi_r$ *and interpretations* $\mathcal{E} = (\mathcal{D}, \Im)$*:*

$$\mathcal{E}(\varphi) = \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(\varphi)).$$

Proof: "$\subseteq$": Let $a \in \mathcal{E}(\varphi)$. Then $a \in \mathcal{E}(\forall\, r.\varphi_r)$ for all $r \in R$.
Let $(a, b) \in \alpha_{\mathcal{H}}(\Im)(w)$ for some $w \in \alpha_{\mathcal{H}}(\top)$.
$\succ$ for all $r \in R$ with $w \in \alpha_{\mathcal{H}}(r)$: $(a, b) \in \Im(r)$
$\succ b \in \mathcal{E}(\varphi_r)$
$\succ b \in \mathcal{E}(\bigwedge_{w \in \alpha_{\mathcal{H}}(r)} \varphi_r)$
$\succ a \in \alpha_{\mathcal{H}}(\mathcal{E})(\forall\, w. \bigwedge_{w \in \alpha_{\mathcal{H}}(r)} \varphi_r)$
$\succ a \in \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(\varphi))$.

"$\supseteq$": Let $a \in \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(\varphi))$.
Let $b \in \Im_a(r)$ for some $r \in R$.
$\succ b \in \alpha_{\mathcal{H}}(\Im)_a(w)$ for exactly one $w \in \alpha_{\mathcal{H}}(r)$
$\succ b \in \mathcal{E}(\varphi_r)$ since $b \in \mathcal{E}(\bigwedge_{w \in \alpha_{\mathcal{H}}(r)} \varphi_r)$
$\succ a \in \mathcal{E}(\forall\, r.\varphi_r)$
$\succ a \in \mathcal{E}(\varphi)$. ■

4.4.1 A Normal-Form and the Consistency Check

The normal-form for concept terms, which we are going to define now, is not just a simple syntactic standardization. It makes implicit information explicit and it even makes the universal quantifications redundant for the purpose of the consistency check. All explicit and implicit arithmetic information is comprised in the constraint part, and the decomposed quantifications are collected in the quantification part. The constraint part is turned into a proper $\mathcal{L}$-formula by replacing the role terms r with their decomposed arithmetic sums $\Sigma_{\mathcal{H}}(r)$.

The tricky part of the normal-form generation algorithm is the collection of the implicit arithmetic information. Here the system must check which of the role term's atomic components are actually empty. In the PCM example (Section 4.1) we had the situation that $P \wedge C \wedge M$ was inconsistent and therefore the *pcm* atom was empty. This is what has to be checked and made explicit by adding $pcm = 0$ to the constraint part.

Already in this phase of the normal-forming algorithm one must do a consistency check, but only for sub-formulas of smaller size. Therefore the definition of the normal-form and the consistency check are intertwined. As we shall see, one can extract enough implicit arithmetic information this way such that it suffices to check the constraint part for consistency. The quantification part is no longer necessary. (But it is still necessary for the subsumption test.)

Extracting the implicit information can be done in two ways. If the quantification part of the concept term is $\varphi = \forall r_1.\varphi_1 \wedge \ldots \wedge \forall r_k.\varphi_k$, Method 1 works as follows: we map through all subsets $\{\varphi_i, \ldots, \varphi_j\}$ of the quantifications and check whether $\varphi_i \wedge \ldots \wedge \varphi_j$ is consistent. If it is inconsistent, all atoms of $\alpha_{\mathcal{H}}(r_i \cap \ldots \cap r_j)$ are set to zero. Method 2 first decomposes the universal quantifiers (Definition 4.6) and then tests for each atom w, whether $\alpha_{\mathcal{H}w}(\varphi)$ is inconsistent. In this case '$w = 0$' is added to the constraint part. It is a matter of strategy whether to choose Method 1 or 2. In Definition 4.8 we used Method 2. In an implementation, Method 1 might be more efficient. But in both cases some bookkeeping is necessary to avoid redundant checks. If for example $\varphi_1 \wedge \varphi_2$ is inconsistent, then $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ must be inconsistent as well and therefore need not to be checked again.

Definition 4.8 (Normal-Form of Concept Terms) *Let $\varphi = C \wedge N \wedge U$ be a concept term where C is a conjunction of concept names, N is a conjunction of $\mathcal{L}_{\mathcal{R}}$-formulas and U is a conjunction of universal quantifications (all conjunctions may be empty).*
The normal-form $\eta(\varphi)$ of φ is

$$\eta(\varphi) \stackrel{\text{def}}{=} \begin{cases} C \wedge \mathcal{A}_{\mathcal{H}}(\varphi) \wedge \alpha'_{\mathcal{R}}(U) & \textit{if } \mathcal{A}_{\mathcal{H}}(\varphi) \textit{ is consistent} \\ \textit{`inconsistent'} & \textit{otherwise} \end{cases}$$

where

$$\mathcal{A}_{\mathcal{H}}(\varphi) \stackrel{\text{def}}{=} \alpha_{\mathcal{H}}(N) \wedge \bigwedge_{w \in \alpha_{\mathcal{H}}(\top), \mathcal{A}_{\mathcal{H}}(\alpha_{\mathcal{H}w}(\varphi)) \textit{ is inconsistent}} w = 0$$

and

$$\alpha'_{\mathcal{R}}(U) \stackrel{\text{def}}{=} \bigwedge_{\text{`}\forall w.\psi\text{'} \in \alpha_{\mathcal{H}}(U), \mathcal{A}_{\mathcal{H}}(\varphi) \not\models w=0 \textit{ and } \not\models \psi} \forall w.\psi.$$

■

$\mathcal{A}_{\mathcal{H}}(\varphi)$ in the normal-form of $\eta(\varphi)$ is the arithmetic constraint part. It consists of the original constraint part $\alpha_{\mathcal{H}}(N)$ where the cardinality terms have been replaced by the corresponding sum-terms (Definition 3.2,*iii*), plus some equations $w = 0$. These equations come from quantifications $\forall w.\psi$ where ψ is inconsistent, and therefore there cannot be any w-successors. $\alpha'_{\mathcal{R}}(U)$ is the decomposed and reduced quantification part where all quantifications over empty atomic role components and all tautologies have been eliminated.
The normal-form of concept term (14)

$$\textit{exactly}\ 3\ ep \wedge \textit{exactly}\ 2\ em \wedge \textit{atmost}\ 4\ es \wedge\ \forall ep.P \wedge \forall ec.C \wedge \forall em.M$$

in the PCM example is

$$\begin{array}{rcll} p + pc + pm + pcm & = & 3 & \wedge \\ m + pm + cm + pcm & = & 2 & \wedge \\ s + p + c + m + pc + pm + cm + pcm & \leq & 4 & \wedge \\ pcm & = & 0 & \wedge \end{array}$$
$$\forall p.P \wedge \forall c.C \wedge \forall m.M \wedge \forall pc.(P \wedge C) \wedge \forall pm.(P \wedge M) \wedge \forall cm.(C \wedge M) \wedge \forall pcm.(P \wedge C \wedge M)$$

We shall prove that a concept term φ is consistent if and only if the arithmetic constraint part $\mathcal{A}_{\mathcal{H}}(\varphi)$ has a solution. This allows us to use an arithmetic equation solver as the primary inference engine for $\mathcal{TF}^{++}$.

Theorem 4.9 (Almost an Equivalence Transformation) *For any concept term φ and interpretation $\mathcal{E}$: $\mathcal{E}(\varphi) = \alpha_{\mathcal{H}}(\mathcal{E})(\eta(\varphi))$.*

Proof: Again let $\varphi = C \wedge N \wedge U$ and $\eta(\varphi) = C \wedge \mathcal{A}_{\mathcal{H}}(\varphi) \wedge \alpha'_{\mathcal{R}}(U)$ as in Definition 4.8. We prove $\mathcal{E}(\varphi) \subseteq \alpha_{\mathcal{H}}(\mathcal{E})(\eta(\varphi))$ and $\mathcal{E}(\varphi) \supseteq \alpha_{\mathcal{H}}(\mathcal{E})(\eta(\varphi))$ separately. Since nothing changes for concept names C, there is nothing to prove for these cases.

Case '$\subseteq$': From Theorem 3.3 we obtain $\mathcal{E}(N) = \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(N))$. For the arithmetic part of φ we therefore only need to show: if $a \in \mathcal{E}(\varphi)$ then $a \in \alpha_{\mathcal{H}}(\mathcal{E})(w = 0)$ where '$\forall w.\psi$' $\in \alpha_{\mathcal{H}}(U)$ is inconsistent. Applying Lemma 4.7 we find that if ψ is inconsistent, there cannot be a w-successor of a. Therefore $w = 0$ must be true in $\alpha_{\mathcal{H}}(\mathcal{E})$.
For the quantification part U we also use Lemma 4.7: $\mathcal{E}(\varphi) = \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(\varphi))$. $\alpha'_{\mathcal{R}}(\varphi)$ differs from $\alpha_{\mathcal{H}}(\varphi)$ in that some tautologies get removed. Therefore the '$\subseteq$'-part of this proof is trivial.

Case '$\supseteq$': The arithmetic part of the proof is a direct consequence of $\mathcal{E}(N) = \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(N))$. For the quantification part we have to show that the removed tautologies hold in $\mathcal{E}(\varphi)$. Suppose $a \in \alpha_{\mathcal{H}}(\mathcal{E})(\eta(\varphi))$, i.e. in particular $a \in \alpha_{\mathcal{H}}(\mathcal{E})(\mathcal{A}_{\mathcal{H}}(\varphi))$. For a removed quantification $\forall w.\psi$ for which $\mathcal{A}_{\mathcal{H}}(\varphi) \models w = 0$ holds, there cannot be a w-successor of a in $\alpha_{\mathcal{H}}(\mathcal{E})$. Therefore $\forall w.\psi$ holds. A removed formula $\forall w.\psi$ in which ψ is a tautology holds anyway. Thus, $a \in \alpha_{\mathcal{H}}(\mathcal{E})(\alpha'_{\mathcal{R}}(U)) = \alpha_{\mathcal{H}}(\mathcal{E})(\alpha_{\mathcal{H}}(U)) = \mathcal{E}(U)$ according to Lemma 4.7. ■

The theorem states in particular that whenever φ is satisfiable, then the arithmetic part of $\eta(\varphi)$ is satisfiable. This weaker statement is sufficient for the soundness of the consistency check.

The theorem above is quite useful, but not yet strong enough for a completeness proof of a consistency check, which tests only the arithmetic parts in the normalized concept term. The difficult part in the completeness proof is the construction of a model for φ from a solution for $\mathcal{A}_{\mathcal{H}}(\varphi)$. We construct a tree model by working from the leaf nodes up to the root node. This corresponds to induction on the structure of φ from the base case, the concept names, up to φ itself.

In this construction, it is necessary to join different sub-trees under a new root node, and sometimes to copy a sub-tree several times and hang the copies under a new root node. We define the *disjoint union* of two interpretations as a generalization of an operation that puts two interpretations together and a copy operation. Then we show that the disjoint union of two models again is a model. This is in a sense the inverse of Segerberg's 'bulldozing theorem' for modal logic, which says that generated sub-models are models again [Seg71].

Definition 4.10 (Disjoint Union of Interpretations) *Let $\mathcal{E}_i = (\mathcal{D}_i, \Im_i)$, $i = 1, 2$ be two interpretations for our concept language such that $\Im_1 = (\Im_{\mathcal{L}}, \Im_{C,1})$ and $\Im_2 = (\Im_{\mathcal{L}}, \Im_{C,2})$, i.e. the $\mathcal{L}$-part is identical.*

We define the disjoint union $\mathcal{E}_1 \uplus \mathcal{E}_2 \stackrel{\text{def}}{=} (\mathcal{D}_1 \uplus \mathcal{D}_2, \Im_1 \uplus \Im_2)$ *where $\mathcal{D}_1 \uplus \mathcal{D}_2$ is the usual disjoint union of two sets, the common $\Im_{\mathcal{L}}$-part is unchanged and $\Im_{C,1} \uplus \Im_{C,2}$ is defined:*

i) $(\Im_{C,1} \uplus \Im_{C,2})(s) \stackrel{\text{def}}{=} \Im_{C,1}(s) \uplus \Im_{C,2}(s)$ *where s is any role or concept name.*

ii) $(\Im_{C,1}(f) \uplus \Im_{C,2}(f))(a) \stackrel{\text{def}}{=} \begin{cases} (\Im_{C,1}(f))(a) & \text{if } a \in \mathcal{D}_1 \\ (\Im_{C,2}(f))(a) & \text{if } a \in \mathcal{D}_2 \end{cases}$

if f is a functional role name. ■

Proposition 4.11 *For every concept term φ and interpretations $\mathcal{E}_1$ and $\mathcal{E}_2$: $(\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi) = \mathcal{E}_1(\varphi) \uplus \mathcal{E}_2(\varphi)$.*

Proof: For the proof below notice that for all φ: $\mathcal{E}_i(\varphi) \subseteq \mathcal{D}_i$ and therefore for all φ, ψ: $\mathcal{E}_1(\varphi) \cap \mathcal{E}_2(\psi) = \emptyset$ because $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$.

The proof is by induction on the structure of φ. If φ is a concept name (base case) then $(\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi) = \Im_1(\varphi) \uplus \Im_2(\varphi) = \mathcal{E}_1(\varphi) \uplus \mathcal{E}_2(\varphi)$.

Case $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned}
&(\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi_1 \wedge \varphi_2)\\
= \quad & (\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi_1) \cap (\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi_2)\\
= \quad & [\mathcal{E}_1(\varphi_1) \uplus \mathcal{E}_2(\varphi_1)] \cap [\mathcal{E}_1(\varphi_2) \uplus \mathcal{E}_2(\varphi_2)]\\
= \quad & [\mathcal{E}_1(\varphi_1) \cap \mathcal{E}_1(\varphi_2)] \cup [\mathcal{E}_1(\varphi_1) \cap \mathcal{E}_2(\varphi_2)] \cup [\mathcal{E}_2(\varphi_1) \cap \mathcal{E}_1(\varphi_2)] \cup [\mathcal{E}_2(\varphi_1) \cap \mathcal{E}_2(\varphi_2)]\\
= \quad & [\mathcal{E}_1(\varphi_1) \cap \mathcal{E}_1(\varphi_2)] \cup [\mathcal{E}_2(\varphi_1) \cap \mathcal{E}_2(\varphi_2)]\\
= \quad & \mathcal{E}_1(\varphi_1 \wedge \varphi_2) \uplus \mathcal{E}_2(\varphi_1 \wedge \varphi_2).
\end{aligned}$$

Case $\varphi = \forall\, r.\psi$:

$$\begin{aligned}
&(\mathcal{E}_1 \uplus \mathcal{E}_2)(\forall\, r.\psi)\\
= \quad & \{a \mid \text{for all } b : a(\Im_1 \uplus \Im_2)(r)b \text{ implies } b \in (\mathcal{E}_1 \uplus \mathcal{E}_2)(\psi)\}\\
= \quad & \{a \mid \text{for all } b : a\Im_1(r)b \text{ implies } b \in \mathcal{E}_1(\psi)\} \uplus\\
& \{a \mid \text{for all } b : a\Im_2(r)b \text{ implies } b \in \mathcal{E}_2(\psi)\}\\
= \quad & \mathcal{E}_1(\forall\, r.\psi) \uplus \ \mathcal{E}_2(\forall\, r.\psi).
\end{aligned}$$

Case $\varphi \in \mathcal{L}_{\mathcal{R}}$:

$$\begin{aligned}
&(\mathcal{E}_1 \uplus \mathcal{E}_2)(\varphi)\\
= \quad & \{a \in \mathcal{D}_1 \uplus \mathcal{D}_2 \mid (\Im_1 \uplus \Im_2)_a \models \varphi\}\\
= \quad & \{a \in \mathcal{D}_1 \mid (\Im_1 \uplus \Im_2)_a \models \varphi\} \cup \{a \in \mathcal{D}_2 \mid (\Im_1 \uplus \Im_2)_a \models \varphi\}\\
= \quad & \{a \in \mathcal{D}_1 \mid (\Im_1)_a \models \varphi\} \cup \{a \in \mathcal{D}_2 \mid (\Im_2)_a \models \varphi\}\\
= \quad & \mathcal{E}_1(\varphi) \uplus \mathcal{E}_2(\varphi)
\end{aligned}$$

■

Now we prove that the consistency of the arithmetic constraint part of the normal-form for concept terms is sufficient to guarantee consistency of the original term.

Theorem 4.12 (Completeness of the Normal-Form) *If for a concept term φ, $\mathcal{A}_{\mathcal{H}}(\varphi)$ is satisfiable then for all $n > 0$ there is a model $\mathcal{E} = (\mathcal{D}, \Im)$ with $|\mathcal{E}(\varphi)| = n$.*

Proof: By induction on the structure of φ.

Base Case: φ is a conjunction $\varphi = c_1 \wedge \ldots \wedge c_k$ of concept names. We can choose $\mathcal{D} = \Im(c_i) = \{1, \ldots, n\}$ for $1 \leq i \leq k$. All role names are interpreted as empty sets. $|\mathcal{E}(\varphi)| = n$ obviously holds.

Induction Step: Now let $\varphi = C \wedge N \wedge U$ as in Definition 4.8 contain some nested terms and let $\Im_E$ be a model for $\mathcal{A}_{\mathcal{H}}(\varphi)$. $\Im_E$ assigns non-negative integers to all $w \in \alpha(\mathcal{R})$ and numbers to all functional role names. We construct $\mathcal{E}$ in a sequence of steps.

Step 1: For every atomic component $w = r_1 \ldots r_m \in \alpha(\mathcal{R})$ we define a model $\mathcal{E}_w = (\mathcal{D}_w, \Im_w)$:

i) if $\alpha_{\mathcal{H}w}(\varphi)$ is empty then $\mathcal{D}_w \stackrel{\text{def}}{=} \{1, \ldots, n\}$; and $\Im_w$ maps role- and concept names to the empty set;
ii) if $\mathcal{A}_{\mathcal{H}}(\alpha_{\mathcal{H}w}(\varphi))$ is inconsistent then $\mathcal{D}_w = \emptyset$;
iii) if $\mathcal{A}_{\mathcal{H}}(\alpha_{\mathcal{H}w}(\varphi))$ is consistent we choose a model $\mathcal{E}_w$ such that $|\mathcal{E}_w(\alpha_{\mathcal{H}w}(\varphi))| = \Im_E(w)$, which is possible according to the induction hypothesis.

Step 2: Now take $\mathcal{E}_0 = (\mathcal{D}_0, \Im_0)$ to be the disjoint union of all the $\mathcal{E}_w$ defined in the previous step.

Step 3: Let $\mathcal{E}_1 = (\mathcal{D}_1, \Im_1)$ be such that $\mathcal{D}_1 = \mathcal{D}_0 \cup \{a\}$ where a is a fresh new *root element*, and $\Im_1$ is like $\Im_0$ but extends the interpretation of the role names r as follows:

$$\text{if for some } w \in \alpha(\mathcal{R}) \text{ and } b \in \mathcal{E}_w(\alpha_{\mathcal{H}w}(\varphi) \text{ then } (a,b) \in \Im_1(r); \tag{17}$$

$$\text{for functional roles } f \text{ we require } \Im_w(f)(a) \stackrel{\text{def}}{=} \Im_E(a); \tag{18}$$

$$\text{furthermore for concept names } c \in \varphi \text{ we state } \Im_1(c) \stackrel{\text{def}}{=} \Im_0(c) \cup \{a\}. \tag{19}$$

Step 4: Now we choose as $\mathcal{E}_2$ among the domain elements of $\mathcal{E}_1$ a smallest generated sub-model of $\mathcal{E}_1$ that satisfies φ. Let b be the root of $\mathcal{E}_2$. ($b = a$ may well be possible.) Finally let $\mathcal{E}$ be the disjoint union of n copies of $\mathcal{E}_2$.

Using Proposition 4.11, the claim is proved by showing that $\mathcal{E}_2(\varphi) = \{b\}$ where b is the new root element introduced in Step 4. And for proving this, it is sufficient to prove $a \in \mathcal{E}_1(\varphi)$ where a is the root element introduced in Step 3.
We must show $a \in \mathcal{E}_1(\varphi_i)$, $1 \leq i \leq k$, if $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_k$.

Case φ_i is a concept name: $a \in \mathcal{E}_1(\varphi_i)$ because of (19).

Case $\varphi_i = \forall r.\psi$: Let $c \in \Im_1(r)(a)$.
$\succ c \in \Im_1(w)(a)$ for some $w \in \alpha_{\mathcal{H}}(r)$
$\succ c \in \mathcal{D}_w$ and $c \in \mathcal{E}_w(\psi)$ (construction of $\mathcal{E}_0$ and $\mathcal{E}_1$ in Steps 2 and 3)
$\succ c \in \mathcal{E}_0(\psi)$ (Proposition 4.11)
$\succ a \in \mathcal{E}_1(\forall r.\psi)$. (construction of $\mathcal{E}_1$)

Case $\varphi_i \in \mathcal{L}_{\mathcal{R}}$: The models $\mathcal{E}_w$ had been chosen in Step 1 such that $|\mathcal{E}_w(\alpha_{\mathcal{H}w}(\varphi))| = \Im_E(w)$. In Step 3, these sets were used as a-role successors for w. For a given role r, its role successors are then the disjoint union of all the role successors for its atomic components. With the same arguments as in the 'if' part of Theorem 3.4 it follows that $a \in \mathcal{E}_1(\varphi_i)$ is satisfied. ■

We can summarize the results in a corollary which gives us the basis for a consistency checker for $\mathcal{TF}^{++}$: compute $\mathcal{A}_{\mathcal{H}}(\varphi)$ and submit it to an arithmetic equation solver. If the equation solver finds a single solution, then φ is consistent, otherwise not.

Corollary 4.13 *A concept term φ is consistent if and only if the arithmetic constraint part $\mathcal{A}_{\mathcal{H}}(\varphi)$ in $\eta(\varphi)$ is consistent.*

4.4.2 The Subsumption Test

Testing subsumption means figuring out whether $\varphi \models \psi$ holds, or more precisely, whether $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ holds for two concept terms φ and ψ and for *all* interpretations $\mathcal{E}$. In our case we make use of our normal-form for concept terms where the arithmetic information is comprised in the arithmetic constraint part and the quantifications are decomposed into their atomic components. The structure of the normalized φ is $\varphi = C_\varphi \wedge N_\varphi \wedge U_\varphi$ where C_φ is just a conjunction of concept names, N_φ is the constraint part and U_φ are the decomposed quantifications. The same for $\psi = C_\psi \wedge N_\psi \wedge U_\psi$.

In order to verify $\varphi \models \psi$ we have to prove each conjunctive part in ψ from φ. The normal-form allows us to separate the problem. The C_ψ-part can only follow from the C_φ-part, which is a pure propositional problem. The constraint part N_ψ can only follow from the N_φ-part. This, we assume can be solved by a corresponding arithmetic algorithm, for example by reducing it to the consistency problem for $\varphi \wedge \neg\psi$ (the language $\mathcal{L}$ usually has negation).

Finally, the atomic components $\forall w.c'$ of ψ can only follow from corresponding $\forall w.c$ components of φ, if $c \models c'$. Here the algorithm becomes recursive. There is, however, one other possibility where

$\forall w.c'$ is a consequence of φ, namely if N_φ forces $w = 0$. Then there are no w-successors, and $\forall w.c'$ is true.

Theorem 4.14 (Subsumption Test) *Let $\varphi = C_\varphi \wedge N \wedge U$ be a* consistent *concept term with normal-form $\eta(\varphi) = C_\varphi \wedge N_\varphi \wedge U_\varphi$. Let $\psi = C_\psi \wedge N' \wedge U'$ be another concept term with normal-form $\eta(\psi) = C_\psi \wedge N_\psi \wedge U_\psi$. C_φ and C_ψ are conjunctions of concept names, N and N' are the arithmetic constraint parts, and U and U' are the universal quantifications.*

We have $\varphi \models \psi$ if and only if

i) $C_\varphi \supseteq C_\psi$,

ii) $N_\varphi \models N_\psi$ and

iii) for all '$\forall w.c'$' $\in U_\psi$:

  a) $N_\varphi \models w = 0$ or

  b) there is some '$\forall w.c$' $\in U_\varphi$ with $c \models c'$.

Proof: The 'only-if'-part (soundness) is obvious. For the 'if'-part (completeness) we show by induction on the structure of ψ that if one of *i)*, *ii)*, or *iii)* is violated, there is an interpretation $\mathcal{E}$ with $\mathcal{E}(\varphi) \not\subseteq \mathcal{E}(\psi)$. This means, if $\mathcal{E}(\varphi) \subseteq \mathcal{E}(\psi)$ for all interpretations $\mathcal{E}$, then *i)*, *ii)* and *iii)* must be true, and we can check subsumption by testing *i)*, *ii)*, and *iii)*.
In the base case of the induction, ψ consists only of concept names, which makes it a trivial propositional problem. Now let ψ have some nested terms.

Case *i)* is violated: This is again a propositional problem.

Case *ii)* is violated: Let $\Im_\varphi$ be an interpretation satisfying N_φ, but falsifying N_ψ.
By Theorem 4.12, $\Im_\varphi$ can be turned into a model $\mathcal{E} = (\mathcal{D}, \Im)$ with $\mathcal{E}(\varphi) = \{a\}$ for a suitable a.
Suppose $a \in \mathcal{E}(\psi)$. (Remember $\psi = C_\psi \wedge N' \wedge U'$)
$\succ \Im_a \models N'$ (cf. Definition 4.2 for $\Im_a$)
and by the construction of $\mathcal{E}$: $\Im_\varphi(w) = |\Im_a(w)|$ for all atoms $w \in \alpha(\mathcal{R})$
$\succ \Im_\varphi(\alpha_\mathcal{H}(r)) = |\Im_a(r)|$ for all role-terms r (by structural induction)
$\succ \Im_\varphi \models \alpha_\mathcal{H}(N')$.
N_ψ consists of the conjunction of $\alpha_\mathcal{H}(N')$ and a conjunction of equations $w = 0$ according to Definition 4.8 of the normal-form η.
Since $\Im_\varphi \models \alpha_\mathcal{H}(N')$, but $\Im_\varphi \not\models N_\psi$, it must falsify one of these equations $w = 0$. That means $\Im_\varphi$ requires the existence of at least one w-successor of a. Now, the equation $w = 0$ was inserted into N_ψ because $\mathcal{A}_\mathcal{H}(\psi)$ is inconsistent for some $\forall w.\psi$ in U_ψ (Definition 4.8 of $\mathcal{A}_\mathcal{H}(\varphi)$). Thus, there cannot be a w-successor, which contradicts the fact stated above that there is a w-successor of a. Thus, $a \in \mathcal{E}(\psi)$ was wrong and $\mathcal{E}(\varphi) \not\subseteq \mathcal{E}(\psi)$ must hold.

Case *iii)* is violated: Suppose there is some '$\forall w.c'$' $\in U_\psi$ and $N_\varphi \not\models w = 0$, i.e. there is some w-successor of a, and
a) there is no corresponding '$\forall w.c$' $\in U_\varphi$ or
b) there is some '$\forall w.c$' $\in U_\varphi$ and $c \not\models c'$.

Subcase a) Since φ is consistent, there is a model $\mathcal{E}$ with $a \in \mathcal{E}(\varphi)$ for some a. If $a \notin \mathcal{E}(\forall w.c')$ we are done. Suppose $a \in \mathcal{E}(\forall w.c')$. Since in the normal-form all '$\forall wc'$' with tautologous c' have been eliminated, we can assume there is some b with $b \notin \mathcal{E}(c')$ ($\mathcal{E}$ can be chosen appropriately). Now turn $\mathcal{E}$ into $\mathcal{E}'$ which is like $\mathcal{E}$, but b is either a new w-successor of a (in case there were no ones before), or b is exchanged with an existing w-successor (to keep the number of w-successors the same). $a \in \mathcal{E}'(\varphi)$ because nothing is said about w-successors in φ, but $a \notin \mathcal{E}'(\psi)$. Thus, $\mathcal{E}'(\varphi) \not\subseteq \mathcal{E}'(\psi)$.

Subcase b) By the induction hypothesis there is a model $\mathcal{E}'$ with $\mathcal{E}'(c) \not\subseteq \mathcal{E}'(c')$. We can assume $\mathcal{E}'$ is a generated sub-model with some b as root, i.e. $\mathcal{E}'(c) = \{b\}$, but $b \notin \mathcal{E}'(c')$. Since φ is satisfiable, there is some tree model $\mathcal{E}$ for φ with $\mathcal{E}(\varphi) = \{a\}$ for some a (again a generated sub-model) whose $\mathcal{L}$-part agrees with $\mathcal{E}'$. As in *Subcase a*, since $N_\varphi \not\models w = 0$, we can either add b (together with the whole tree $\mathcal{E}'$) as a new w-successor, or exchange it with an existing w-successor, in both cases, this is a disjoint union construction following Definition 4.10. The new interpretation $\mathcal{E}''$ still satisfies φ in a, but not ψ. Thus, $\mathcal{E}''(\varphi) \not\subseteq \mathcal{E}''(\psi)$. ∎

Let us illustrate the subsumption checking procedure with an example taken from [Neb90], page 80. The task is to show that a concept[19]

$$D \stackrel{\text{def}}{=} \mathit{atleast}\,3\,r$$

is subsumed by a concept

$$C \stackrel{\text{def}}{=} \forall\,(r\cap p).a \;\wedge\; \forall\,(r \cap q).\mathit{not\text{-}a} \;\wedge\; \mathit{atleast}\,2\,(r\cap p) \;\wedge\; \mathit{atleast}\,2\,(r\cap q)$$

where a and $\mathit{not\text{-}a}$ have been axiomatized such that their conjunction is inconsistent.[20]
To show that $C \models D$, following Theorem 4.14 we need the normal-forms $\eta(C)$ and $\eta(D)$. We begin with the decomposition of the role set $S = \{r, p, q\}$ according to Definition 2.1. We obtain

$$\begin{aligned} \alpha_{\mathcal{H}}(r) &= \{r, rp, rq, rpq\} \\ \alpha_{\mathcal{H}}(p) &= \{p, rp, pq, rpq\} \\ \alpha_{\mathcal{H}}(q) &= \{q, rq, pq, rpq\}. \end{aligned}$$

Now we are able to decompose the universal quantifiers in C following Definition 4.6 into

$$\forall\,rp.a \;\wedge\; \forall\,rpq.(a \wedge \mathit{not\text{-}a}) \;\wedge\; \forall\,rq.\mathit{not\text{-}a}$$

Since $a \wedge \mathit{not\text{-}a}$ is inconsistent we obtain

$$\alpha'_{\mathcal{R}}(U_C) \;=\; \forall\,rp.a \wedge \forall\,rq.\mathit{not\text{-}a}$$

and the first equation for $\mathcal{A}_{\mathcal{H}}(C)$ with $rpq = 0$. Using this equation to simplify the normal-form $rp + rpq \geq 2 \wedge rq + rpq \geq 2$ of $\mathit{atleast}\,2\,(r\cap p) \wedge \mathit{atleast}\,2\,(r \cap q)$ we obtain the two inequations

$$\mathcal{A}_{\mathcal{H}}(C) \;=\; rp \geq 2 \;\wedge\; rq \geq 2$$

Normalizing D leads to a single inequation

$$\eta(D) = \mathcal{A}_{\mathcal{H}}(D) \;=\; r + rp + rq + rpq \geq 3$$

Since C_C and C_D are empty there is nothing to show for concept names. Similarly for the universal quantification part because $\alpha'_{\mathcal{R}}(U_D) = \emptyset$. Thus it remains to prove that $\mathcal{A}_{\mathcal{H}}(C) \models \mathcal{A}_{\mathcal{H}}(D)$ by showing that

$$rp \geq 2 \;\wedge\; rq \geq 2 \;\Rightarrow\; r + rp + rq + rpq \geq 3$$

is valid, and this is obvious.

5 Summary and Outlook

We have presented a technique for reasoning about cardinalities of finite sets and, as an application of this technique, we have shown how it can be turned into an inference procedure for a particular concept language. The method exploits the fact that any finite collection of sets can be built up from a finite set of mutually disjoint building blocks, the atoms of a finite Boolean algebra.

Based on a syntactic representation of the atoms, we have developed algorithms for computing an optimal decomposition that takes a syntactic representation of the sets and information about set

[19] The notation has been adapted to fit into our framework.

[20] See the Appendix for an example of how such an axiomatization can be achieved.

theoretic relations between them as input. The algorithms exploit subset relationships, information about disjointness of sets, and information about partitioning of some sets into disjoint components. This helps in reducing the usually exponential number of atoms. As we could show for the language $\mathcal{TF}^{++}$, further assumptions about disjointness of sets are possible in particular applications. This reduces the number of atoms considerably.

We have embedded the set language into an arithmetic language, such that it becomes possible to reason about cardinality of finite sets. The atomic decomposition is the means to replace the set-cardinality terms in the arithmetic formulas by simple sum-terms. This way no extra mechanism is necessary on the arithmetic side. The integration of the set-terms into the arithmetic language has been done in a generic way. It works for all arithmetic languages which have at least a notion of non-negative integers and an addition function. A typical example of such a language is the language of Diophantine equation systems. The problem of solving Diophantine equation systems has come up in a number of other areas as well, such as unification of predicate logic terms with associative-commutative function symbols [PS82], constraint handling in logic programming languages [Jou94], integer programming [Sch86, Kov80], and network flow problems [Hu69]. A variety of different algorithms that solve Diophantine equation systems is presented in the literature. The first algorithms [Hue78, For83] were limited to the case where a single equation has to be solved. More recent approaches can handle systems of equations directly [GH85, CF89, CD94] or even general linear constraints involving inequations [AC95, CD94] and disequations [DT95]. Therefore we have not gone into further details of the equation solving aspects.

However, in our first application, the concept language $\mathcal{TF}^{++}$, it turned out that all reasoning problems can be reduced to consistency checks for equation systems and thus no explicit solution need be computed. This calls for optimized algorithms, which do not compute solutions, but only check consistency. First approaches have been presented in [Dom91, BC96] and we are currently exploring ways to develop a consistency checking algorithm that exactly suits the purpose of our application where equations with mixed integer and real number valued variables may occur.

The atomic decomposition method is a new approach for developing inference systems for concept languages. Traditional approaches, in particular tableaux methods, are focused on the logical parts of these languages, and cannot deal very well with arithmetic information. In contrast to this, the decomposition method focuses primarily on the arithmetic parts. The information represented by the logical connectives has to be treated on a different level. In the case of $\mathcal{TF}^{++}$, it is implicitly encoded in the normal-form generation algorithm for concept terms. This makes the treatment of the logical connectives more difficult, from a conceptual point of view, rather than for the actual inference engine. Therefore so far we have only investigated conjunction and the universal quantifier. We have a relatively clear picture of how the other connectives are to be treated, but the technical details are beyond the scope of this paper. On the arithmetic part, however, the expressivity of $\mathcal{TF}^{++}$ is only limited by the power of the equation solver at hand. But already for very simple equation solvers, $\mathcal{TF}^{++}$ goes far beyond the traditional *atleast* and *atmost* number restrictions.

There are still many open problems if we want to use the decomposition method as primary inference engine in concept logics. First of all the other logical connectives, existential quantifier, disjunction and negation have to be investigated in this context. Fortunately, once negation is available, the treatment of some of the more advanced constructs of concept logics comes for free. For example, qualified number restrictions[21] *atmost nr.c* can be translated into $|r'| \leq n \wedge \forall r'.c \wedge \forall (r \setminus r').\neg c$, where r' is a new sub-role of r. On the other hand, for the qualified number restriction *atleast n r.c*, the expressivity of $\mathcal{TF}^{++}$ is already sufficient, because it can be translated into $|r'| \geq n \wedge \forall r'.c$ with a new sub-role $r' \subseteq r$. More complicated, however, seems to be the treatment of role constructors such as role composition, inverse roles, role value maps, etc., as well as the treatment of special properties of the roles, for example transitivity. These topics are subject to further work.

In this paper we have considered only the T-Box part of $\mathcal{TF}^{++}$, and there only the consistency

[21] An example for a qualified number restriction is *atmost 2 has-child.teacher*. It denotes the set of all objects which have at most two children who are teachers.

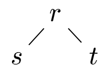
and subsumption check. Our approach to classification and the whole area of A-Box reasoning requires presentation in another paper to allow for a detailed discussion.

Our approach opens a new field of research in the knowledge representation area, where the emphasis is more on the data side. The goal is to close the gap between symbolic logic and real data in relational databases.

6 Appendix: The PCM Example

For the PCM example introduced in Section 4 we demonstrate how role hierarchies and number restrictions can be used to axiomatize that $P \wedge C \wedge M$ are inconsistent, but the concepts are pairwise consistent. The axiomatization is purely artificial. There is no intuitive meaning for the axioms.

First of all we need a small role hierarchy with roles $\{r, s, t\}$. It is depicted to the right. Furthermore we require that s and t are disjoint. The specification is $\mathcal{H} = \{r, s \subseteq r, t \subseteq r, s \dagger t\}$. The axioms are now:

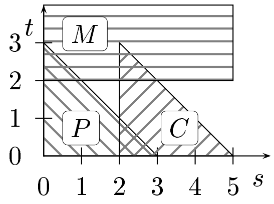


$$P = \mathit{atmost}\, 3\, r \quad (20)$$

$$C = \mathit{atmost}\, 5\, r \wedge \mathit{atleast}\, 2\, s \quad (21)$$

$$M = \mathit{atmost}\, 5\, s \wedge \mathit{atleast}\, 2\, t \wedge \mathit{atmost}\, 4\, t \quad (22)$$

The little graph on the right shows the possible values, s and t can take, given that they are disjoint and constrained by r. Obviously the intersection of all three is empty, but pairwise it is not empty.

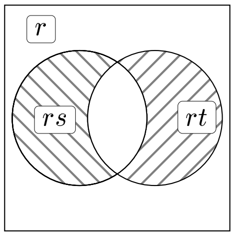


For the decomposition method it is quite easy to find out that $P \wedge C \wedge M$ is indeed inconsistent. The sets to be decomposed are $\{r, s, t\}$. Exploiting the hierarchy for r, s and t, we find the following decomposition

$$\alpha_{\mathcal{H}}(r) = \{r, rs, rt\}, \quad \alpha_{\mathcal{H}}(s) = \{rs\}, \quad \alpha_{\mathcal{H}}(t) = \{rt\}.$$

The translation of (20, 21, 22) is now

$$\begin{array}{lrcl} (20): & r + rs + rt & \leq & 3 \\ (21): & r + rs + rt & \leq & 5 \\ & rs & \geq & 2 \\ (22): & rs & \leq & 5 \\ & rt & \geq & 2 \\ & rt & \leq & 4 \end{array}$$



The inequations $rs \geq 2$ and $rt \geq 2$ contradict $r + rs + rt \leq 3$, therefore the set of inequations does not have a solution.

A Bit of Equation Solving

The main part of the PCM example has a more interesting equation solving problem. The equations (15)

$$\begin{array}{rcl} p + pm + pc & = & 3 \\ m + pm + cm & = & 2 \\ s + p + c + m + pc + pm + cm & \leq & 4 \end{array}$$

are supposed to imply that there are at most three chemists. That means $c + pc + cm \leq 3$ is a theorem, given (15). Negating the theorem and introducing slack variables, we obtain the four equations.

$$\begin{array}{lrcl} E_1: & p + pm + pc & = & 3 \\ E_2: & m + pm + cm & = & 2 \\ E_3: & s + p + c + m + pc + pm + cm + s_1 & = & 4 \\ E_4: & c + pc + cm - s_2 & = & 4. \end{array}$$

Applying a Gaussian-like elimination procedure, we find:

$$\begin{array}{lrcl} E_3 - E_2 = E_5: & s + p + c + pc + s_1 & = & 2 \\ E_4 - E_5 = E_6: & cm - s - p - s_1 - s_2 & = & 2 \\ E_2 - E_6 = E_7: & m + pm + s + p + s_2 + s_1 & = & 0. \end{array}$$

Since all variables are non-negative, E_7 has only the trivial solution: $m = 0$, $pm = 0$, $s = 0$, $p = 0$, $s_2 = 0$, $s_1 = 0$. This simplifies E_1 to $pc = 3$ and E_2 to $cm = 2$, and this in turn simplifies E_4 to $c = -1$, which is not possible. Thus, the equations are inconsistent. The theorem must be true.

References

[AC95] F. Ajili and E. Contejean. Complete solving of linear diophantine equations and inequations without adding variables. In U. Montanari and F. Rossi, editors, *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP-95).* Springer, 1995.

[BC96] A. Boudet and H. Comon. Diophantine equations, Presburger arithmetic and finite automata. In H. Kirchner, editor, *Proceedings of the Colloquium on Trees in Algebra and Programming (CAAP'96)*, Lecture Notes in Computer Science. Springer, 1996.

[BH91] F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(2):8–15, 1991.

[BPS94] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[BS85] R. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[CD94] E. Contejean and H. Devie. An efficient incremental algorithm for solving systems of linear diophantine equations. *Journal of Information and Computation*, 113:143–172, 1994.

[CF89] M. Clausen and A. Fortenbacher. Efficient solution of linear diophantine equations. *Journal of Symbolic Computation*, 8:201–216, 1989.

[Che80] B. F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, 1980.

[DLNN95] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. Research Report DFKI-RR-95-07, DFKI, 1995.

[Dom91] E. Domenjoud. Solving systems of linear diophantine equations: An algebraic approach. In A. Tarlecki, editor, *Mathematical Foundations of Computer Science 1991: Proc. of the 16th International Symposium*, volume 520 of *Lecture Notes in Computer Science*, pages 141–150. Springer, 1991.

[DT95] E. Domenjoud and A. Tomas. From Elliot-MacMahon to an algorithm for general linear constriants on naturals. In Ugo Montanari and Francesca Rossi, editors, *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP-95).* Springer, 1995.

[For83] A. Fortenbacher. Algebraische unifikation, 1983.

[Gab70] D. M. Gabbay. Selective filtration in modal logics. *Theoria*, 36:323–330, 1970.

[Gab72] D. M. Gabbay. A general filtration method for modal logics. *Journal of Philosophical Logic*, 10:135–146, 1972.

[GH85] T. Guckenbiehl and A. Herold. Solving linear diophantine equations. Technical Report SEKI-85-IV-KL, University of Kaiserslautern, 1985.

[HNSS90] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90), Stockholm, Sweden,*, 1990.

[Hu69] T. C. Hu. *Integer programming and network flows.* Addison-Wesley, Reading, Mass., 1969.

[Hue78] G. Huet. An algorithm to generate the basis of solutions to homogeneous linear diophantine equations. *Information Processing Letters*, 7(3), 1978.

[Jou94] J.-P. Jouannaud, editor. *First International Conference on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*. Springer, Berlin, New York, 1994.

[Kov80] L. B. Kovacs. *Combinatorial methods of discrete programming*, volume 2 of *Mathematical methods of operations research.* kademiai Kiado, Budapest, 1980.

[LS66] E. Lemmon and D. Scott. *Intentional Logics.* Stanford, 1966.

[Mac94] R. MacGregor. A description classifier for the predicate calculus. In *Proceedings AAAI-94*, pages 213–220. Morgan Kaufmann, San Francisco, 1994.

[Min90] M. Minsky. A framework for representing knowledge. In R. J. Brachman and H. J. Levesque, editors, *Reprinted in Readings in Kwowledge Representation*, pages 245–262. Morgan Kaufmann, San Francisco, 1990.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence 422. Springer, 1990.

[OSH95] H. J. Ohlbach, R. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. Research Report MPI-I-95-2-008, Max-Planck-Institute of Computer Science, 1995. To appear in H. Wansing (ed), Proof Theory for Modal Logic, Oxford Univ. Press.

[PS82] G. E. Peterson and M. E. Stickel. Complete sets of reductions using associative and/or commutative unification. Technical Note 269, SRI, 1982.

[Sch86] A. Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics. Wiley, Chichester, New York, 1986.

[Seg71] K. Segerberg. An essay in classical modal logic (3 vols.). Filosofiska studier, nr. 13, Uppsala Universitet, 1971.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.