A Concept Language for an engineering application with part–whole relations

Ulrike Sattler*
RWTH – Aachen, uli@cantor.informatik.rwth-aachen.de

Introduction

Terminological Knowledge Representation (TKR) Systems are powerful means to represent unambiguous knowledge – like knowledge in technical domains. We investigate how TKR Systems can be used in process modeling, a field dealing with modeling huge chemical plants. As these plants are very complex, support of top–down modeling is a quite ambitious, but useful task for TKR Systems. An interesting problem to solve in this context is the handling of composite objects: An appropriate TKR System for this application should be able to

- handle different part–whole relations, for example Component–Composite or Stuff–Object (most of which are transitive),
- represent inverses of these part–whole relations,
- model transitivity–like interactions between part–whole relations since some chains of different part–whole relations imply further part–whole relations. These implicit relations permit reasoning about objects having parts, which have parts, which... which have certain properties.
- represent special characteristics of composite objects, for example parts belonging exclusively to a single whole.

Hence the application calls for a concept language with powerful role forming operators.

In this paper, results of an investigation of part–whole relations and their relevance for a process–modeling application are given and a concept–language $\mathcal{P}$ with appropriate expressive power is defined. Satisfiability of concept terms in $\mathcal{P}$ is undecidable, hence it is necessary to drop some (but not many) of the demands made for the benefit of decidability. Several ways to handle the high complexity of inference algorithms of $\mathcal{P}$ are discussed.

The application

Process modeling plays an important role in process engineering, for planning as well as for optimization and controlling. To model chemical plants, they are decomposed into components like distillation columns, tubular reactors, valves, mixed phases, signal transformers, etc. These components are again decomposed into components which are again decomposed, etc., until components are obtained whose physico–chemical characteristics can be described via differential, algebraic or integro–algebraic equations. Hence support of modeling with varying granularity and standard building blocks is indispensable. Furthermore, there is a large number of standard building blocks, which have to be specialized or modified for each concrete model. It is very difficult to define these numerous blocks such that the implicit taxonomy is the same as the explicitly stated and intended one, and a system able to infer implicit subsumption relations could help the model builder to verify his/her definitions. A TKR System able to handle part–whole relations could give this support.

Demands made by the application

Classifications of part–whole relations can be found in [Winston *et al.*,1987; Gerstl and Pribbenow,1993] and a fusion of these classifications seems to be adequate for the given application. Integration of part–whole relations into TKR Systems is treated in [Padgham and Lambrix,1994; Artale *et al.*,1994; Franconi *et al.*,1994], but the application asks for an integration with more expressive power since reasoning in this application needs for example transitive part–whole relations and consequences of transitivity–like interactions between these relations.

A widely held opinion is that *the* part–whole relation is not transitive in general. However, the non–transitive counter–examples come from mixing up different types of part–whole relations. For example: This arm is part of Herbert. Herbert is part of this orchestra. To conclude "This arm is part of the orchestra." does not make much sense – at least it sounds odd. In this case, Component–Object and Member–Collection relations have been mixed.

The next table shows all types of part–whole relations relevant for the given technical application, how they interact and, in the diagonal, whether they are transitive or not. Examples are given to illustrate these relati-

*The author is sponsored by the Deutsche Forschungsgemeinschaft under Grant No. Sp 230\ 6–6

Relation, Example	aCC$b\wedge$ $bXc \Rightarrow$	aMCb $bXc \Rightarrow$	aSEb $bXc \Rightarrow$	aSOb $bXc \Rightarrow$	aIOb $bXc \Rightarrow$	Remarks
Component – **C**omposite `motor-car`	aCCc	aCCc	aCCc	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	Component inherent part-whole boundaries, whole is inhomogeneous
Member – **C**ollection `grain-2 g salt`	Nothing[1]	Nothing	Nothing	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	parts defined through whole, parts not locally fixed
Segment – **E**ntity `front car-car`	Impos.[2]	aMCc	aSEc	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	Arbitrary part-whole boundaries
Quantity – **M**ass `2 kg steel-steel`	Impos.	aMCc	aQMc	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	Whole has no boundaries, no volume; part is bounded
Stuff – **O**bject `steel-bike`	Impos.	Nothing	Impos.	aSOc	Impos.	Part is a unbounded mass
Ingredient – **O**bject `2 kg steel-bike`	Impos.	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	aSOc	$\exists d : a$QMd $\wedge\ d$SOc	Part and whole are spatially inseparable

Figure 1: A survey of different part–whole relations and their transitivity–like interactions

ons and their domains, exact definitions cannot be given here. Please note that the three relations "Quantity-Mass", "Stuff-Object" and "Ingredient-Object" are necessary to reason about objects made of some ingredients if we can not calculate: If we know that this bike has a frame and that its frame is made of 6 kg aluminium — what is to infer for the whole bike beside the fact that it is made of aluminium?

The five middle columns list pseudo–transitive consequences. Relations are abbreviated by their initials, e.g. aCCb stands for an element a being part of b with respect to the relation Component–Composite. The variable X refers to the relation in the respective row.

A column showing consequences of $a\mathrm{QM}b \wedge bXc$ (QM is short for the relation "Quantity–Mass") for part–whole relation X is missing since there are no such consequences (note that the Quantity–Mass relation is not transitive).

Since a relation X is transitive iff it is the transitive closure of some relation $Y \subseteq X$, we introduce six primitive roles `is_d_component, is_d_member`, etc., to model these direct, minimal relations Y and define six roles representing the transitive closure of the respective direct role (if it is transitive) and the above mentioned transitivity–like connections between them:

(defrole is_component (is_d_component $\sqcup$
(is_member $\circ$ is_d_component)$)^+$),
(defrole is_member
(is_d_member $\sqcup$ (is_d_member $\circ$ is_segment)
$\sqcup$ (is_d_member $\circ$ is_quantity))),
(defrole is_segment (is_d_segment)$^+$),
(defrole is_quantity
(is_d_quantity)$\sqcup$ (is_segment $\circ$ is_d_quantity)),
(defrole is_ingredient (is_d_ingredient)),
(defrole is_stuff (is_d_stuff $\sqcup$
(is_d_stuff $\circ$ (is_component $\sqcup$ is_member $\sqcup$
is_segment $\sqcup$ is_ingredient $\sqcup$ is_quantity))
$\sqcup$ (is_member $\circ$ is_ingredient)
$\sqcup$ ((is_quantity)$^{-1}$ $\circ$ is_segment $\circ$ is_ingredient)
$\sqcup$ ((is_quantity)$^{-1}$ $\circ$ is_ingredient
$\circ$ (is_ingredient $\sqcup$ is_component $\sqcup$ is_member
$\sqcup$ is_segment $\sqcup$ is_quantity)))$^+$),

Syntax and semantics defined in the usual manner, for example see [Baader and Hollunder,1991]. If we want to allow only for models where the direct part–whole roles are interpreted as pairwise disjoint relations, we have to define a superconcept `Disj` of all other concepts by:

```
(defconcept Disj (∀ Y*.
  (∀ ((is_d_component ⊓ is_d_member) ⊔
     (is_d_component ⊓ is_d_segment) ⊔···).⊥)))
```

where `Y` has to be replaced by a disjunction of all primitive roles occuring in the TBox and their respective inverses (see for example [Schild,1991] for an explanation of this universal role). By defining each concept as a subconcept of `A`, we prevent parts from being simultaneously related to one whole via two different direct part–whole relations. Hence direct part–whole roles can only be interpreted as pairwise disjoint relations.

Beside these relations, the application asks for means to express special characteristics on part–whole relations, as for example parts belonging exclusively to a whole.

Exclusive parts: A part is an exclusive part of a whole, if it has at most one role–filler with respect to one of the direct is–part–of relations. One way to express this in a terminological system is to use number re-

[1] This means that a, b, c with these relations are possible, but nothing can be concluded.

[2] Where "Impossible" means that there cannot be such b.

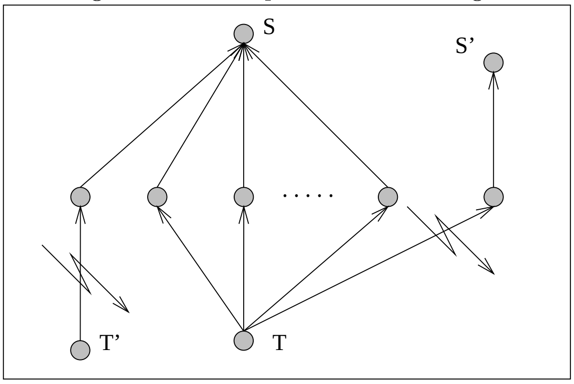
strictions on roles, for example to express that a motor is part of at most one whole:

```
(defconcept motor (≤ 1 is_d_component:
car) ⊓... ).
```

Multi–Possessed parts: In our technical applications, we have to model composite devices whose parts use a common part. In the next figure for example, a device S contains some reactors that all use the same tank T. It is not possible to view the tank as part of S, since the reactors have a tank as a necessary direct part. One way to define those multiple possessions is the following.

```
(defconcept S
   ((= 1 ((is_d_component)^-1)^2:  T)⊓...))
```

Owner–Restricted parts: These objects are parts of a set of wholes, which are characterized as being themselves parts of one single whole.



For example, we do not want any other reactor beside the ones contained in the device S to use the same tank T. This restriction of wholes to contain parts can be expressed through a restriction on parts to be contained by wholes:

```
(defconcept T
 ((= 1 (is_d_component)^2:S) ⊓
 (∀is_d_component.(∃is_d_component.S))⊓...))
```

Essential parts: The existence of an essential part is essential for the existence of a whole. This can easily be expressed using Exists Restriction

```
(defconcept human
  (∃ (is_d_component)^-1.brain))
```

Dependent parts: For a dependent part the existence of its whole is essential. The definition of dependent parts can easily be done using exists restrictions:

```
(defconcept ceiling
  (∃ is_d_component.room))
```

Finally, part–whole relations are acyclic. This demand can be realized by introducing a second superconcept Acyc of each concept which is defined as:

```
(defconcept Acyc
  (∀ Y*.(∀ (self ⊓ (is_component ⊔
    is_segment ⊔ is_d_ingredient ⊔...)).⊥)))
```

where Y is the previously described universal role and self $= id(\top)$.

Consequences of these demands

Summarizing, to represent these part–whole relations as well as their transitive–like extensions and characteristics in the suggested way, a concept language $\mathcal{P}$ is defined where role terms are built from role names using the following role–forming operators:

the identity role (self $= id(\Delta)$), inverse roles (r^{-1}), role conjunction ($r \sqcap s$), role disjunction ($r \sqcup s$), role composition ($r \circ s$) and transitive closure of roles (r^*).

Concept terms are built from role terms and concept names using the following concept–forming operators:

concept conjunction ($C \sqcap D$), primitive concept negation ($\neg A$, where A is a primitive concept), exists restriction ($\exists\, r.\, C$), value restriction ($\forall\, r.\, C$), single restriction ($\leq\, 1\, r:\, C$).

Concept terms and role terms are interpreted in the usual manner. Note that $\mathcal{P}$ includes neither concept disjunction nor a top concept.

Investigation of the decidability of the satisfiability of concept terms in $\mathcal{P}$ led us to the question whether concept disjunction in general can be expressed using role disjunction, inverse roles and composition of roles. This would mean that renouncement of concept disjunction with the goal of achieving lower complexity is pointless if we have such strong role–forming operators.

Lemma 1 Concept disjunction can be expressed in $\mathcal{P}$.

More formally: Let C be a concept term of $\mathcal{P}$ including (possibly nested) concept disjunction. Then a concept term $\hat{C}$ of $\mathcal{P}$ can be constructed with the following properties: Each interpretation I of C can be extended to an interpretation $\hat{I}$ of $\hat{C}$ such that $(*)$ holds.

$$\begin{array}{c} \mathrm{dom}(I) = \mathrm{dom}(\hat{I}) \text{ and} \\ \forall x^I \in \mathrm{dom}(I) : x^I \in C^I \text{ iff } x^{\hat{I}} \in C^{\hat{I}}. \end{array} \quad (*)$$

Vice versa, if $\hat{I}$ is an interpretation of $\hat{C}$, and I is the restriction of $\hat{I}$ to role and concept names in C, then $(*)$ is satisfied. The idea in the construction of $\hat{C}$ is to substitute concept terms of the form $D \sqcup E$ by

$$(\exists (d \sqcup e).N) \sqcap (\forall d \circ d^{-1}.D) \sqcap (\forall e \circ e^{-1}.E),$$

where d, e are new primitive roles and N is a new primitive concept.

Corollary 1 Satisfiability of concept terms in $\mathcal{P}$ is undecidable.

Using lemma 1 and observing that

1. a top concept $\top$ can be simulated by (C $\sqcup \neg$C),
2. global features (functional roles) can be expressed (let $\mathtt{r}_1, \ldots, \mathtt{r}_n$ be all role names appearing in a concept term C, then in all connected models of C $\sqcap$ ($\forall$ ($\mathtt{r}_1 \sqcup \mathtt{r}_1^{-1} \sqcup \ldots \sqcup \mathtt{r}_n^{-1})^*$.($\leq$ 1 r:$\top$)) the role r is interpreted as a feature),
3. role value maps on feature chains can be simulated, for example ($\sqsubseteq$ f g) as
   (($\leq$ 1 (f $\sqcup$ g):$\top$) $\sqcap$ (($\forall$ f.$\bot$) $\sqcup$ ($\exists$ g.$\top$)),

Corollary 1 follows from undecidability of $\mathcal{FSL}$ with role intersection [Schild,1991] or can easily be shown by a reduction of the domino problem.

Note, that in [Baader and Hanschke,1993] it was shown that the extension of $\mathcal{ALC}$ by functional roles, transitive closure of roles and integration of concrete domains leads to undecidable inference problems. This means that a knowledge representation system able to handle part–whole relations can not "calculate". Hence all constants (especially landmark–values) and their comparisons will have to be treated symbolically, and their exact treatment will be left to the user or a numerical system – which will be needed anyway to continue the modeling process.

Possible ways out

One way to handle undecidability (or the high complexity of a reduced language) is to use incomplete algorithms to solve satisfiabilty or subsumption problems. The aim of this knowledge base is not "just" to classify concepts and primitives in process engineering, but to support modeling of chemical plants, which means that the user has to be able to interprete answers given by his/her system. For example, if a user wants to know whether a concrete model of a chemical process contains a phase which is carcinogenic, then an incorrect answer "yes" is not dangerous. By contrast, an incorrect answer "yes" to the question whether all phases contained in this plant are eatable could be dangerous. This means that the semantic of answers given by an incomplete inference algorithm has to be known. Furthermore, as the one asking questions has to ask "good" questions in order to get no dangerous answers, users have to know about this semantic. There are two first ideas for incomplete reasoning with part–whole relations where the meaning of incompleteness is well defined.

A first approach, which can be viewed as a special case of reasoning with incomplete inference algorithms where only "yes" answers to satisfiability questions and "no" answers to subsumption questions can be incorrect, is to disallow role conjunction and to restrict single restrictions to primitive or negated primitive roles. This modification of $\mathcal{P}$ yields a concept language with decidable inference problems. As shown in [Schild,1991; De Giacomo and Lenzerini,1994b; 1994a], subsumption is decidable for this highly expressive sublanguage, even provided with full concept negation. For our application, this means that we have to allow for models where direct part–whole roles are no longer interpreted as disjoint relations and with possibly cyclic part–whole relations, whereas all part–whole relations as well as their transitive–like extensions can still be expressed. Furthermore, we can no longer express owner–restricted or multipossessed parts.

The next approach is just an idea – and might or might not work: It is to modify the language $\mathcal{P}$ in order to decrease the complexity of its inference algorithms by substituting the transitive closure of a role R – which is the smallest transitive role extending R – by a role R' which is transitive and contains R. Let us call such a role R' the transitive orbit of R. There are hints from modal logic that reasoning with transitive orbits is easier than with transitive closures. The consequences for the application are the following: A TBox $\mathcal{T}$ where R' is the transitive orbit of R has more models than the same TBox $\mathcal{T}'$, with R' beeing the transitive closure of R. Hence theorems of $\mathcal{T}$ are subsets of theorems of $\mathcal{T}'$. This means that "C subsumes D" can be true in $\mathcal{T}'$ while it is not true in $\mathcal{T}$, whereas a concept can be satisfiable in $\mathcal{T}$ and unsatisfiable in $\mathcal{T}'$. Hence, as in the above mentionned case, only "yes" answers to satisfiabilty questions and "no" answers to subsumption questions can be incorrect.

These and other possibilities will be thoroughly investigated.

Acknowledgement

I would like to thank Franz Baader, Diego Calvanese and the anonymous referees for valuable suggestions and comments.

References

[Artale *et al.*, 1994] A. Artale, F. Cesarini, E. Grazzini, F. Pippolini, and G. Soda. Modelling composition in a terminological language environment. In *Workshop Notes of the ECAI Workshop on Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology*, pages 93–101, Amsterdam, 1994.

[Baader and Hanschke, 1993] Franz Baader and Philipp Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German AI-Conference, GWAI-92*, volume 671 of *LNCS*, pages 132–143, Bonn, Deutschland, 1993. Springer-Verlag.

[Baader and Hollunder, 1991] Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, volume 567 of *LNAI*, pages 67–86. Springer-Verlag, 1991.

[De Giacomo and Lenzerini, 1994a] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics (extended abstract). In *Proc. of AAAI-94*, 1994.

[De Giacomo and Lenzerini, 1994b] Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proc. of ECAI-94*, 1994.

[Franconi *et al.*, 1994] Enrico Franconi, Alessandra Giorgi, and Fabio Pianesi. A mereological characterization of temporal and aspectual phenomena. In Carlos Martin-Vide, editor, *Current Issues in Mathematical Linguistics*, pages 269–278. Elsevier, North-Holland Linguistic Series, 1994.

[Gerstl and Pribbenow, 1993] Peter Gerstl and Simone Pribbenow. Midwinters, end games and bodyparts. In N. Guarino and R. Poli, editors, *International Workshop on Formal Ontology-93*, pages 251–260, 1993.

[Padgham and Lambrix, 1994] Lin Padgham and Patrick Lambrix. A framework for part-of hierarchies in terminological logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proc. of KR-94*, pages 485–496, 1994.

[Schild, 1991] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, 1991.

[Winston *et al.*, 1987] M.E. Winston, R. Chaffin, and D. Herrmann. A taxonomy of part whole relations. *Cognitive Science*, 11:417–444, 1987.