Knowledge representation in process engineering

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

Abstract

Process engineering is surely no pure configuration application, but modeling the structure of chemical processes confronts us, in the field of knowledge representation, with similar problems. First, the tasks we are concerned with in process engineering are described as well as how knowledge representation systems can support these tasks. Roughly speaking, this support consists in helping the user in the handling of an object-oriented database. Then it is argued why terminological knowledge representation systems are suitable tools for giving this support and how this support can be realized by these systems. In the last section, we describe some problems that arise because this task asks for a knowledge representation system with special expressive power.

Process engineering

Process engineering is concerned with the design and operation of chemical processes that take place in huge chemical plants. This engineering task includes activities like deciding on an appropriate flowsheet structure (e.g. reaction and separation system configurations), mathematical modeling and simulation of the process behavior (e.g. writing down mathematical equations and performing numerical simulations), sizing of components like reactors, heat exchangers etc. as well as costing and engineering economics. All these tasks are based on appropriate models of the process that is to be designed or operated. These models can be different graphical models, verbal or mathematical models. To support these engineering tasks by appropriate software tools, the development of process models has to be supported. The process models are based on standard building blocks [Marquardt,1994; Bogusch and Marquardt,1995] which are objects representing, among others,

- material entities such as reactors, pipes, control and cooling units,
- models of these entities such as device-, environment-, and connection-models,
- interfaces between these models and so-called implementations describing their behaviour,
- symbolic equations specifying these implementations and variables occuring in these equations which are related to each other as specified in the interfaces,
- ...

Our aim is to support the development of these models. The task of modeling chemical processes is surely no pure configuration task, but especially the modeling of the structure of a chemical plant confronts us with similar problems and tasks: Devices and connections are chosen, their respective interfaces are coupled, complex devices are decomposed into their components or segments, etc..

Problems we want to help with

The highly complex task of modeling chemical plants can be heavily supported by appropriate software tools such as CAD, decision support and numerical tools. In order to give this support, the domain specific knowledge is stored in a frame-based system. This frame-based system is able to store a great variety of the standard building blocks. As the user has to be able to find building blocks (s)he is looking for, standard building blocks are grouped in classes, and these classes are ordered with respect to the *is-a-specialization-of* relation (known also as the *is-a* relation) which yields the class hierarchy. This ordering has to be explicitely stated in each class definition by giving, for each class, the set of its superclasses. As the frame-based system includes powerful features such as methods and triggers, it is far too expressive to compute the implicit subsumption relation on the defined classes. On the other hand, it is flexible in that it can be extended by additional classes of building blocks. This second feature is necessary because in process engineering, the number of standard building blocks increases permanently.

In the sequel, by database we refer to the set of class definitions in the frame-based system. As the complexity of the database increases, navigation in its hierarchy becomes again difficult and modifying or extending the taxonomy becomes dangerous in the sense that they might not yield the desired changes. In fact, the user (the person building models and sometimes extending the database by new classes of standard building blocks) is confronted with the following problems:

1. Navigation in the class taxonomy will become difficult, especially in those parts of the data-

base not often used by the user. Searching for a certain class whose names is not known may take a long time of browsing the hierarchy and comparing different class definitions until the appropriate class is found.

2. Definig a new class A, the user has to arrange it into the existing taxonomy according to its intuition or common sense. (S)he knows that A is a subclass of B, but might be uncertain whether there is a more specific subclass B' of B such that A is a subclass of B'. Because of this uncertainty it is rather probable that the taxonomy gets broader than necessary—which is, on one hand, disadvantageous for the performance of the database system, and, on the other hand, makes navigation more difficult than necessary. Furthermore, it could happen that the user defines an inconsistent or unintendent class. The extension of the database by such a definition can cause needless work.
3. As the database can be modified by more than one user, it is probable that the same class is defined twice—in syntactically different terms and with different names. This does not only blow the size of the database, but is also a source of misunderstanding and trouble.

How these problems can be solved

To help the user with these problems, the database should be equipped with a system that is able to compute implicit specialization relation between defined classes and that is able to test consistency of class definitions. Unfortunately, the frame-based system has far to much expressive power to allow for this automatic reasoning, e.g., the according inference problems are undecidable. The main reason for this fact is the possibility to define triggers and powerful methods in the frame based system.

Fortunately, there is still something that can be done: The content of the database can be mirrored in a knowledge base whose reasoning services are powerful enough to help the user with the problems mentioned above. As a consequence of the above observation, this translation cannot be exact—if it were exact, the interesting problems would still be undecidable—but, by choosing an appropriate knowledge representation system, they can be sufficiently exact. An important point of this mirroring is that the taxonomy of the knowledge base has to be equivalent to the class hierarchy of the database. Even if some properties described in the database cannot be translated accordingly, this equivalence has to be assured. Then the knowledge representation system should be able to help the user with the navigation and modification of the database. It should include an intelligent browser to help finding classes, propose places in the taxonomy where to place a new class, clarify the meaning of a new class definition before the database is extended by this class, and detect semantically identical classes.

Why we chose a TKR-system

In this section, we will argue why a *terminological knowledge representation system* (TKR-system) is the appropriate representation system for the task described above. Before doing so, we will briefly describe TKR-systems.

TKR-systems differ mainly in their underlying *description language*, which are characterized by the sets of so-called concept-forming and role-forming operators. Using these operators, one can define *complex concepts* (which are interpreted as sets of elements of the interpretation domain) and *roles* (which are interpreted as binary relation on the interpretation domain) using primitive concepts and roles. Operators available in almost all implemented systems are union, intersection, negation, value restrictions, as well as restrictions on the number of role successors. A terminological knowledge base is a set of concept and role definitions stored in a so-called TBox. A small example for a TBox is given in Figure 1. In this TBox, the concepts `Material-Entity`, `Model`, and `Implementation` are defined (for a matter of space, these definitions are presented only partly). For example, a `Material-Entity` is a `Modeling-Concept` that is associated by the `is-modeled-by` relation to instances of the concept `Model` only, and by the relation `has-function` to instances of the concept `Function` only. A `Model` is, among others, associated by the relation `is-implemented-by` to exactly one `Interfaces`. The concept `Model` are further refined, for example, to concepts like `Device-Model` or `Connection-Model`, which themselves are refined, and so on.

TKR-system are suitable for this task because of the following points:

- TKR-systems can be viewed as a unified framework for class based representational formalisms [Calvanese *et al.*,1994], and are closely related to frame-based systems. The translation from a class definition in a frame-based database to a concept definition in a TBox is natural for many of the properties describable in frame-base systems, hence this translation can be performed automatically.
- For most description languages, there exist sound and complete inference algorithms for the answering of queries. In most cases, these queries are reduced to the basic inference problems such as satisfiability (the question whether a concept can ever be instantiated) or subsumption (the question whether a concept is more general than another one). Soundness and completeness of the inference algorithms implemented in a system imply that queries are always answered correctly after a finite amount of time. The advantage of TKR-systems with sound and complete inference algorithms is that, if the user explicitely describes properties of objects, then these properties are always dealt with by the algorithm—they are not simply disregarded when the algorithm reasons about these objects.
- It is possible to keep the TBox taxonomy equivalent with hierarchy of the database: There are two reasons why a concept could be placed at a different place in the (implicit) taxonomy

Material-Entity	:=	Modeling-Concept $\sqcap$ ($\forall$is-modeled-by.Model) $\sqcap$ ($\forall$has-function.Function)
Model	:=	Structural-Modeling-Concept $\sqcap$ ($\forall$possible-alternative.Model) $\sqcap$
		($\forall$active-alternative.Model) $\sqcap$
		($\forall$is-implemented-by.Implementation) $\sqcap$ ($=$ 1 is-implemented-by) $\sqcap$
		($\forall$active-interfaces.Interfaces) $\sqcap$ ($\geq$ 1 active-interfaces)
Implementation	:=	Structural-Modeling-Concept $\sqcap$
		($\forall$behaviour.Equation) $\sqcap$ ($\geq$ 1 behaviour) $\sqcap$
		($\forall$variables.Symb-vars) $\sqcap$ ($\geq$ 1 variables)

Figure 1: Example TBox

of the TKR-system than the according class in the database hierarchy: It can be (1) because of the inexactitude of the translation and (2) because the user placed the class too high in the database hierarchy. If such a mismatch occurs, the user is asked to verify which of the cases did arise. In the first case, the definition of the concept is modified such that afterwards, this concept is placed correctly. In the second case, the superclasses of the new class are modified accordingly.

- The services required for the support of the modeller in the usage of the database can be achieved by TKR-systems. Standard services provided by TKR-systems comprise the calculation of the implicit subsumption relation between two concepts, the calculation of the implicit concept taxonomy, as well as testing whether a concept is satisfiable.

  Based on these services, navigation can be supported in the following way: First, the user is asked to describe—in an incomplete way—the class (s)he is looking for. Then the TKR-system gives him/her the most specific classes subsumed by this description. The user should then be able to give more information concerning the class (s)he is looking for by looking more closely at these classes. Naturally, this information can also include some of the classes proposed by the system which are more general than the one the user is looking for. By iterating this ask-and-tell procedure, the user is guided to the class (s)he is looking for.

  Before adding a new class definition to the database, the user can ask the TKR-system to arrange the according concept into the TBox taxonomy. Investigating this taxomony, we can prevent the user from unintended definitions.

  Testing satisfiability of a concept before adding its according class to the database can prevent from extensions by inconsistent classes.

- Its declarative semantics enables the user to correctly define the concepts (s)he has in mind. Rule based formalisms may seem more natural, but when characterising a class one has in mind, it is difficult to fix all rules necessary to define this class.

Which TKR-system to choose?

TKR-systems differ in the expressive power of the underlying description language, and we are now confronted with the question *which* TKR-system is the most appropriate one for the task described above. In the last decade, a great variety of TKR-systems has been investigated [Levesque and Brachman,1987; Nebel,1988; Schmidt-Schauss,1989; Patel-Schneider,1989; Hollunder *et al.*,1990; Donini *et al.*,1991; Baader and Hanschke,1993; De Giacomo and Lenzerini,1994; Calvanese *et al.*,1995]. However, there are still many open questions concerning TKR-systems, their expressivity as well as their behavior in realistic applications. It is clear that, for a given application, the description language has to be expressive enough to represent relevant properties of the objects in the application domain. Unfortunately, the more expressive a description language language is, the more time or space is needed to compute query answers[1]. Hence a compromise has to be found between computational complexity and expressive power. Furthermore, as "expressive power" is not 1-dimensional, it is difficult to tell whether the expressive power of one description language is "better" for a given application as the expressive power of another one.

This process engineering application is surely no pure configuration application. Nevertheless, the structural modeling of a plant can be seen as a configuration task: Devices and connections are chosen from a set of generic devices; they are possibly modified according to the actual construction; connections between these devices have to be defined; they are possibly decomposed into their parts in order to get a more precise model; and finally, devices are aggregated from different subdevices modeled by different users. As a consequence, in the field of knowledge representation, we are confronted with problems which occur also in configuration applications:

Part-whole relations: As the plants to be modeled are very complex, the user should be able to decompose and aggregate devices and connections of the process to be modeled (this is also important for

[1] However, driven by demands from other applications, it could be shown in [Baader *et al.*,1994] that worst-case intractable languages may behave quite well in practice.

the reuse of models as well as for distributed modeling). Hence the TKR-system has to be able to represent composite objects appropriately.

For this appropriate representation of composite objects, part-whole relations have to be treated correctly by the inference algorithms of the TKR-system. As for other applications [Gerstl and Pribbenow,1993; Franconi,1994; Artale *et al.*,1994; Pribbenow,1995], we are confronted with the question

- which part-whole relations are needed for the appropriate representation of the complex objects in our application. It turned out that objects are decomposed with respect to the component–composite, segment–entity, and member–collection relation, each of them a spezialisation of the general part-whole relation. Roughly speaking, parts with respect to the member–collection are not coupled to each other and are "of the same kind", whereas components can be coupled to each other in any way and may be quite different one from each other, and segments are from a similar kind, but coupled to each other. As the user might want to refer to a part, not knowing on which level of decomposition it can be found, we have to represent the general, transitive part-whole relation as well.
- how these relations interact: If, in the intuition of the user, the segment–entity is transitive, then it has to be represented as a transitive role. But what about a component a of a segment b of a whole c — ist a also a component of c? Questions concerning these interactions are not yet completely answered, but they have to be answered in order to handle composite objects appropriately.
- which properties concerning part-whole relations are relevant in the application: For example, the existence of a certain part can be essential for the proper definition of the whole, in contrast to parts being optional; a part can be exclusive in the sense that it might be a part of at most one object b without the possibility to be shared by other objects beside those having b as a part; a part can be functional for an object in that this object does no longer work correctly if this part is broken; and many others more [Simons,1987]. The representation of these properties is quite useful because knowledge concerning these properties is required for powerful consistency-testing procedures: It thus can be verified, for example, if all essential parts are specified and, if this is not the case, either a suitable one can be determined or the user is informed on this missing part.

  As at least the general part-whole relation is transitive, an appropriate TKR-system has to be able to handle some kind of transitive relations. Using transitive relations, the user can refer to parts along a number of decomposition levels not known in advance or along any (finite) number of decomposition levels. Hence an interesting question is, in which ways transitive relations can be included into description languages and handled by their inference algorithms. We investigated this question for an expressive, well-known description language in [Sattler,1996].

Number restrictions: As for configuration problems, objects are often characterized by the number of objects they are related to by some relation. For example, we want to describe devices having at least 7 inputs or exactly 5 outputs. In description languages, this can be done using number restrictions as in

$$(\texttt{device} \sqcap (\geq 7\ \texttt{input})),$$
$$(\texttt{device} \sqcap (= 5\ \texttt{output})).$$

This possibility is available in almost all implemented TKR-systems, but not sufficiently expressive for our application: We wanted to describe devices having *the same* number of inputs as of outputs, as in

$$(\texttt{device} \sqcap (= \alpha\ \texttt{input}) \sqcap (= \alpha\ \texttt{output})),$$

or devices having less inputs as each of its parts have, as in

$$(\texttt{device} \sqcap (= \alpha\ \texttt{input}) \sqcap (\forall \texttt{has-part}.(> \alpha\ \texttt{input})).$$

In [Baader and Sattler,1996a], these *symbolic* number restrictions are introduced and investigated. Unfortunately, it turned out that the basic inference problems such as satisfiability or subsumption get very complex, even undecidable, if this kind of number restriction is allowed in a unrestricted way. Nevertheless, it could be shown that, if their usage is restricted, then we can reason in a sound and complete way about concepts containing symbolic number restrictions.

Furthermore, we want to restrict the number of objects that are related via a *complex* path of relations to an object. For example, we are interested in describing devices which have at most 7 parts that are components of its components, as in

$$(\texttt{device} \sqcap (\leq 7\ \texttt{has-component} \circ \texttt{has-component})),$$

or we want to describe a device where all the devices it is connected to are controlled by the same controller:

$$(\texttt{device} \sqcap (= 1\ \texttt{connected-to} \circ \texttt{controlled-by})).$$

The complexity of the basic inference algorithms depends on which operators, beside/instead of composition $\circ$ are allowed inside number restrictions. In [Baader and Sattler,1996b] it is shown that some combinations lead to undecidability of the basic inference problems whereas for other combinations, we could give sound and complete algorithms solving these problems.

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] F. Baader and P. 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 Sattler, 1996a] F. Baader and U. Sattler. Description logics with symbolic number restrictions. In *Proc. of ECAI-96*, 1996. To appear.

[Baader and Sattler, 1996b] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proc. of KR-96*. M. Kaufmann, Los Altos, 1996. To appear.

[Baader *et al.*, 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[Bogusch and Marquardt, 1995] R. Bogusch and W. Marquardt. A formal representation of process model equations. *Computers and Chemical Engineering*, 19:211–216, 1995.

[Calvanese *et al.*, 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of KR-94*, pages 109–120, Bonn, 1994. M. Kaufmann, Los Altos.

[Calvanese *et al.*, 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of DOOD-95*, volume 1013 of *LNCS*, pages 229–246, 1995.

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

[Donini *et al.*, 1991] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of KR-91*, Boston (USA), 1991.

[Franconi, 1994] E. Franconi. A treatment of plurals and plural quantifications based on a theory of collections. *Minds and Machines*, 3(4):453–474, November 1994.

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

[Hollunder *et al.*, 1990] B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.

[Levesque and Brachman, 1987] H. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Marquardt, 1994] W. Marquardt. Trends in computer-aided process modeling. In *Proc. of ICPSE '94*, pages 1–24, Kyongju, Korea, 1994.

[Nebel, 1988] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[Patel-Schneider, 1989] P. F. Patel-Schneider. Undecidability of subsumption in NIKL. *AIJ*, 39:263–272, 1989.

[Pribbenow, 1995] S. Pribbenow. Modeling physical objects: Reasoning about (different kinds of) parts. In *Time, Space, and Movement Workshop 95*, Bonas, France, 1995.

[Sattler, 1996] U. Sattler. The complexity of concept languages with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*, LNAI. Springer-Verlag, 1996. To appear.

[Schmidt-Schauss, 1989] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proc. of KR-89*, pages 421–431, Boston (USA), 1989.

[Simons, 1987] P. M. Simons. *Parts. A study in Ontology.* Oxford: Clarendon, 1987.