Using description logic for configuration problems

Holger Wache
Universit"at Bremen
P.O. Box 330 440
28334 Bremen
wache@informatik.uni-bremen.de

Gerd Kamp
Universit"at Hamburg
Vogt-K"olln-Str.30
22527 Hamburg
kamp@informatik.uni-hamburg.de

In order to use a model–based approach for describing configuration problems the following types of knowledge have to be represented:

- the *components* of which a configuration can consist. Generally the components are collected in a *catalog*.
- the allowed/necessary *relationships* between the components, that define a valid set of components in a configuration solution.
- a *specification* of the configuration problem to solve.

The configuration problem is solved, when a set of components is selected from the catalog in such a way, that the components satisfy the specifications and do not contradict any relationships.

The main difficulty in representing configuration problems lies not in modeling the components but in describing the relationships. In comparsion to other existing model–based configuration systems like PLAKON [RC91], KONWERK [G"u95a, G"u95b], or COSMOS [Hei93], the term "relationship" is sweeping. Several kinds of relationships can be identified in configuration problems:

- *part–whole taxonomy:* components are described as composition from other components
- *functional dependencies:* components can be considered as function providers. For providing these functions they need other functions provided by other components and so on.
- *physical laws in a technical domain:* such as $U = R * I$ or $m = r * f$
- *procedures* in the mechanical engineering domain there are procedures verifying and/or calculating values like temperature in crankshaft
- ...

Therefore, to describe components and their relationships a KR system needs a rich and expressive language. In our opinion, terminological systems provide such a language.

On the inference level of solving configuration problems, one important task is the selection of appropriate components from the catalog. Normally a component is not identified by its name but by its properties. So selection of components can be seen as a classical classification task.

The difficulty in the classification task lies in the way how the properties of a component are determined. The relationships between the components impose certain requirements (i.e. their properties) and constraints on the components that have to be obeyed. Computing the relevant requirements for a component have to be taken into account during the classification task. Desciption logics provides a powerful mechanism for classification. The main advantage is that procedures for evaluating the relationships can interact with the classification mechanism.

In our oppinion description logics are useful for solving configuration problems because they provide a rich representation language for modelling the components and their relationships and a powerful classi-

fication mechanism for the selection task. As best to our knowledge there is no other knowledge representation formalism that provides this two features on the base of a clear semantics. Many representation formalism allow the object centered representation of the components but do not directly support the classification task.

A terminological system can only act as a part of a configuration system. It is designated for representing the components and relationships and supporting a few inferences (i.e. classification). Representing other knowledge (e.g. control knowledge for guiding the configuration process) or additional inferences (e.g. aggregation of components) have to be done in the other part of the configuration system. The other parts (e.g. a rule based system for aggregation) use description logic as a service provider: description logic holds the (incomplete) problem solution and offers a few inferences.

In a terminological system the components and (a few) relationships are represented as concepts and relations in the T-Box. Our description formalism is based on the non–trivial $\mathcal{ALC}$ language (including OR and NOT) [HN90]. To describe component properties the language must be extended with features. Roles can be used to represent relationships between components. For the part–whole taxonomy a role hierarchy is useful.

While modeling components and their relationships can be expressed in a T-Box, more important for a configuration task are the inferences in the A-Box. Computing the subsumption graph of the components is perhaps an interesting task during knowledge acquisition, but really needed only for a correct behaviour of the A-Box inferences. Important inferences are the consistency test checking the configuration solution wrt. the components and relationships in the T-Box, and the two kinds of individual classification (strong and weak realisation). Strong realisation computes the set of components an individual *has to* classify to; weak realisation determines the components to which an individual *can possibly* classify to, if new information is available.

Unfortunately, a few modifications and extensions of the terminological system are needed: First, not all relationships between components can be expressed in the concept language. In a technical domain it must be possible to represent physical laws like $U = R * I$. Thus algebraic (in)equalities can be handled by an algebraic constraint solver. For correct inferences in the terminological system (especially in the T-Box), a constraint solver has to be integrated into the terminological part. Baader and Hanschke [BH91] describe a generic Concrete Domain interface for integrating external decision procedures. An algebraic constraint solver can be considered as such a decision procedure. But the most available terminological systems do neither provide such a generic interface nor algebraic constraints.

Second, the A–Box inferences needs an adaptation. Normally, the consistency test returns a boolean value. In order to do so, a terminological system based on a tableaux method implicitly generates a model. In the configuration task the calculated model is of interest to guide the next configuration steps. Therefore, the consistency test should also return the computed model, which in the end is the solution to the configuration problem specification.

Returning the computed model also requires an adaption of the decision procedures integrated via a concrete domain interface. The procedures should to be capable of returning their computed model. E.g. an algebraic constraint solver should return the values or ranges of its variables. This requirement restricts the choice of the appropriate decision procedure.

Furthermore, the individual classification can be improved. Description logics are based on an open world semantic. For this reason the individual realisation algorithms are very conservative in classifying. It assumes that a new concept description could be added to the T-Box anytime. During a configuration task it can be assumed that no

new component descriptions will be added to the T-Box. Through a (time-consuming) combination of the strong and weak realisation, individuals can be classified to more specific concepts than the pure strong realisation: if the most specific concept, to which an individual i must be classified (strong realisation), is C and if there exists only one subconcept D of C, to which the individual i can be classified (weak realisation), then assume that the individual i is from concept D.

In our experience terminological systems extended by Concrete Domains offer a rich and expressive language to represent the domain knowledge (i.e. the components and relationships). An open problem is the formalisation of the specification. Normally, the specification is vague and uncertain. It is not clear, how such a specification can be transformed into a configuration solution.

Another problem are the inferences — especially the time consuming realisation. To speed up the realisation process the model generation should be shifted from a goal–driven to a data–driven inference. This means, that the system does not generate a hypothesis (can a individual x be realised to concept C) and then tests this hypothesis but computes the classification direct by using the information in the A-Box (because of that the individual x has the feature f it has to classify to the concept C). A data-driven terminological system has to analyse and compile the T-Box before any A-Box inferences will be done.

References

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. Technical Report RR-91-10, DFKI, April 1991.

[G"u95a] A. G"unter. KONWERK – Ein modulares Konfigurierungswerkzeug. In F. Maurer and M.M. Richter, editors, *Expertensysteme 95*, Proceedings in Artificial Intelligence. "infix", St. Augustin, 1995.

[G"u95b] A. G"unter, editor. *Wissensbasiertes Konfigurieren – Ergebnisse aus dem Projekt PROKON.* Dr. Ekkehard Hundt, "infix"-Verlag, St. Augustin, 1995.

[Hei93] M. Heinrich. Ressourcenorientiertes Konfigurieren. *KI*, 7(1):11–15, 1993.

[HN90] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. Technical Report RR-90-04, DFKI, Kaiserslautern, Germany, April 1990.

[RC91] A. Strecker R. Cunis, A. G"unter, editor. *PLAKON – "Ubersicht "uber das System.* Informatik-Fachberichte. Springer, 1991.