Model Construction for Configuration Design

Michael Eisfeld
Chair for Applied Computer Science in Civil Engineering,
Technical University Dresden, Germany
Email: michael.eisfeld@cib.bau.tu-dresden.de

Abstract

In this paper, we motivate the use of the expressive description logic $\mathcal{ALCQF}(\mathcal{D})$ with acyclic TBoxes for configuration design of structural systems. A solution to the design problem is automatically constructed from problem-specific constraints given in the ABox and the persistent conceptual description about the structure and its behaviour in the TBox. Furthermore, we argue that the configuration design problem, for which a solution is defined to be a canonical model of the constructed knowledge base, is decidable because the applied DL has the finite model property. We give an example of our approach by a simple routine design problem from practice.

1 Introduction

Conceptual design requires to construct configurations of structural elements, which channel the applied loads safely to the ground. Suitable alternatives have to be chosen according to the design constraints stated in the design brief. However, almost two-thirds of the solutions designed by engineers do not comply with the structural requirements and the given design constraints in practice. Therefore, structural engineers should be supported by knowledge-based problem solving methods. Their application reduces errors by exploiting persistent knowledge for similar design problem instances at the conceptual design stage. Conceptual design of simple structures shares many commonalities with configuration design because known parameterized elements have to be configured into a structural system with a valid topology. Consequently, problem solving methods for configuration design can be applied to conceptual design of such structural systems and thereby avoid the aforementioned deficiencies.

Configuration design is the task of searching for an assembly of predefined components as solution, which satisfies a set of requirements and obeys a set of constraints [18, 7, 8]. The problem of conceptually designing a structural frame from given design constraints by means of persistent structural knowledge about mereology and topology illustrates our chosen approach in the paper.

We focus on logic-based approaches because we have to employ implicitly represented knowledge. Different logic-based methods have been developed for configuration design and proposed for further research, see [17, 9, 4, 16]. Some of them have been even deployed in industrial settings [14]. In addition, hybrid approaches have been used, which apply different techniques for solving routine design tasks, in order to cover the complexity of given application domains [11]. However, they have a few drawbacks. Either the techniques are undecidable like those proposed in [10, 11], because they employ very expressive languages as first order predicate logic and a configuration language called BHIBS, or they have no declarative semantics for the used DL as in [14], which combines instance checking with procedural rules on different hierarchical levels. We propose a model construction approach for the chosen description logic $\mathcal{ALCQF}(\mathcal{D})$, which possesses the finite model property and was first described in [1] and further developed in [9].

In our approach we represent conceptual knowledge about the structure and its requirements in the terminological component with acyclic TBoxes and knowledge about the behaviour and component attributes in distinct concrete domains, if they have different structural values. We use a arithmetical concrete domain for equilibrium statics and distinct concrete domains for numeric attribute values in order to represent properly the structural domain. We can thus describe that a component is in stable equilibrium because the sum of all terminal forces is zero. For example, the following concept describes that a local equilibrium of forces exists for a beam with an attribute of the interval $depth - range$ from the concrete domain for the dimension length:

$$Component \sqcap \exists beamDepth.depth{-}range \sqcap \\ \exists hasTerminal\ reaction1, hasTerminal\ reaction2, \\ hasTerminal\ action. + .$$

By means of the feature agreement we can define that a component and a system share a single terminal to their outer environment.

$$System \sqcap (InTerminal \downarrow Component\, InTerminal)$$

This is of particular interest for representing correctly the topology of a system. Problem-specific constraints and an actual configuration of the structure are represented in an ABox. The structural engineer starts with the specification of the design constraints in the ABox. Afterwards the algorithm tries automatically to construct a canonical model by adding new assertions according to the defined knowledge base, in which a structural engineering expert afore has deschribed persistent conceptual knowledge about the structure's topology and mereology in the TBox. If a model can be constructed it is a solution to the configuration problem. The solution includes all necessary parts, completely instantiated components and their connected terminals, which secure valid load-paths through the structure. Different models can be built due to the specific constraints given in the ABox prior to the model construction process. We assume that the configuration problem is decidable if the inference problem of consistency testing of a TBox together with an ABox is decidable in our chosen DL. The description logic $\mathcal{ALCQF}(\mathcal{D})$ is decidable because the fusion of the two decidable DLs $\mathcal{ALCF}(\mathcal{D})$ [12] and $\mathcal{ALCQ}$ [6] remains decidable according to [5].

Our paper is organized as follows. First, we introduce the employed description logic $\mathcal{ALCQF}(\mathcal{D})$ with acyclic TBoxes, where we define sample concrete domains for behavioural constraints and concrete attributes. Second, we give a formal problem specification for model-based configuration design, where routine is defined by the availability of conceptual knowledge and behavioural constraints to the algorithm for the construction process at the outset. Afterwards, we give a short example from our structural engineering domain and describe briefly the reasoning service for constructing a model as solution on the given problem specification. Eventually, we conclude the paper.

2 The description logic $\mathcal{ALCQF}(\mathcal{D})$

In this section we describe the decription logic $\mathcal{ALCQF}(\mathcal{D})$. We give examples, why the increased expressivity of concrete domains and feature agreements is important for modelling a structural system and its mathematical equations from statics. We use the syntax and semantics given in [2].

Definition 1 (concrete domain) A *concrete domain* $\mathcal{D}$ is a pair $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, where $\Delta_\mathcal{D}$ is a set, the domain, and $\Phi_\mathcal{D}$ is a set of predicate names over $\Delta_\mathcal{D}$. Each predicate P from $\Phi_\mathcal{D}$ is associated with an arity n and a n-ary predicate $P^\mathcal{D} \subseteq \Delta_\mathcal{D}^n$. According to [3] a concrete domain $\mathcal{D}$ is called admissible iff:

- its set of predicate names is closed under negation and contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$ and
- the satisfiability problem for finite conjunctions of predicates is decidable.

We restrict the use of concrete domains to admissible ones, for which we assume that $\bot_\mathcal{D}$ is the negation of the predicate $\top_\mathcal{D}$. Based on the former, we introduce for the equilibrium equations a concrete domain $\Delta_R := \mathbb{Q}$ over the rational numbers, which was shown to be admissible in [13], and for each component attribute type a separate concrete domain Δ_{Ai} as proposed in [17]. The static equilibrium condition at terminals or for components can be specified on the basis of the following predicates:

- unary predicate $\top_R$ with $(\top_R)^R = \mathbb{Q}$ and a unary predicate $\bot_R$ with $(\bot_R)^R = \emptyset$,
- unary predicates $\leq, =$ and a binary predicate $=$ with the usual extension and
- a ternary predicate $+$ with $(+)^R = \{(q_1, q_2, q_3) \in \mathbb{Q}^3 \mid q_1 + q_2 = q_3\}$.

Additionally, we introduce the unary predicates *depth-range* and *force-range* for separate concrete domains of different structural values.

Definition 2 ($\mathcal{ALCQF}(\mathcal{D})$ syntax) Let C, R and F be disjoint sets of concepts, role and feature names and n a nonnegative integer. We call a composition $f_1 \cdots f_n$ of features a *feature chain*. If C and D are concepts of C, R is a role of R or F, $P \in \Phi_{\mathcal{D}}$ is a predicate of arity n and $u_1, \ldots, u_n$ are feature chains, then concepts can be formed according to the following rules:

$$\begin{aligned} C, D := {} & \top \mid \bot \mid \neg C \mid \\ & C \sqcap D \mid C \sqcup D \mid \\ & \forall R.C \mid \exists R.C \mid \\ & \geq n\, R.C \mid \ \leq n\, R.C \mid \\ & \exists u_1, \ldots, u_n.P \mid u_1 \downarrow u_2 \mid u_1 \uparrow u_2, \end{aligned}$$

for which the feature chains u_i are abstract ones in the agreement and disagreement constructor and concrete feature chains for the existential predicate concept term, accordingly. We will use abbreviations for a feature chain $u = f_1 \cdots f_n$, $\exists u.C$ and $\forall u.C$ for $\exists f_1 \cdots \exists f_n.C$ and $\forall f_1 \cdots f_n.C$ respectively. We employ the usual set theoretic semantics for concepts.

Definition 3 ($\mathcal{ALCQF}(\mathcal{D})$ semantics) We use an *interpretation* $\mathcal{I} = (\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a set $\Delta_{\mathcal{I}}$, namely the abstract domain, and an interpretation function $\cdot^{\mathcal{I}}$. $\Delta_{\mathcal{I}}$ is disjoint from $\Delta_{\mathcal{D}}$. The interpretation function maps each concept name C to a subset $C^{\mathcal{I}}$ of the abstract domain, each role name R to a subset of $R^{\mathcal{I}}$ of $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$, and each feature name f to a partial function $f^{\mathcal{I}} \colon \Delta_{\mathcal{I}} \to \Delta_{\mathcal{D}} \cup \Delta_{\mathcal{I}}$, which we write in the extensional form $(a^{\mathcal{I}}, x^{\mathcal{I}}) \in f^{\mathcal{I}}$. For a feature chain u, the semantics is given by the composition of the partial functions interpreting its components, i.e. $u_1^{\mathcal{I}} = f_n^{\mathcal{I}}(\cdots f_1^{\mathcal{I}}(a^{\mathcal{I}}) \cdots)$. If the symbols are defined as in Definition 2, we extend the interpretation function to complex concept terms as follows:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}, (\neg C)^{\mathcal{I}} := \Delta_{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \exists b \in \Delta_{\mathcal{I}} : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \forall\, b : (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\geq n\, R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid | \{b \in \Delta_{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C)^{\mathcal{I}}\}| \geq n\} \\
(\leq n\, R.C)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid | \{b \in \Delta_{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C)^{\mathcal{I}}\}| \leq n\} \\
(\exists u_1, \ldots, u_n.P)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \exists x_1, \ldots, x_n \in \Delta_{\mathcal{D}} : \\
&\qquad (a, x_1) \in u_1^{\mathcal{I}} \wedge \cdots \wedge (a, x_n) \in u_n^{\mathcal{I}} \wedge (x_1, \ldots x_n) \in P^{\mathcal{D}}\} \\
(u_1 \downarrow u_2)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \exists b \in \Delta_{\mathcal{I}} \colon b = u_1^{\mathcal{I}} = u_2^{\mathcal{I}}\} \\
(u_1 \uparrow u_2)^{\mathcal{I}} &:= \{a \in \Delta_{\mathcal{I}} \mid \exists b_1, b_2 \in \Delta_{\mathcal{I}} \colon u_1^{\mathcal{I}} = b_1, u_2^{\mathcal{I}} = b_2 \wedge b_1 \neq b_2\}
\end{aligned}
$$

We use the standard notion of a *model* for a concept C and apply the standard reductions between the inference problems as can be found in [2].

We employ a TBox for specifing the conceptual model of the structure, the persistent requirements from statics and the loads on the structure. This also includes necessary parts and terminal connections.

Definition 4 (TBox) An *terminological axiom* φ is a concept definition $C \doteq D$ or a concept specialization $C \sqsubseteq D$, where C is a concept name and D is a complex concept description. A finite set of such axioms is called a *TBox* $\mathcal{T}$. We use the standard semantics for TBoxes. An interpretation $\mathcal{I}$ satisfies an axiom of the form $C \doteq D$ and $C \sqsubseteq D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ and $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, respectively. Eventually, we say that an interpretation $\mathcal{I}$ is a *model* for the the TBox $\mathcal{T}$ iff it satisfies all φ in $\mathcal{T}$. We restrict $\mathcal{T}$ to include only acyclic axioms. Thus, we can expand the given TBox before starting the reasoning process.

We use an ABox for the task-specific constraints and the solution structure, which is built up by the model construction process.

Definition 5 (ABox) Let O_D and O_A be disjoint sets of so-called *concrete* and *abstract objects*. If C, R, f and P are defined as in Definiton 2, a and b are elements of O_A and $x, x_1, \ldots, x_n$ are elements of O_D, then the following expressions are an *assertional axiom* ϕ:

$$a : C,\ (a : b) : R,\ (a, x) : f,\ a \neq b,\ (x_1, \ldots, x_n) : P.$$

We call a finite set of such axioms an *ABox* $\mathcal{A}$ and extend the notion of an interpretation to the assertional component by mapping every object name from

O_A to an element of $\Delta_{\mathcal{I}}$ and every object name from O_D to an element of $\Delta_{\mathcal{D}}$, for which the unique name assumption is not imposed. An interpretation $\mathcal{I}$ satisfies an assertion

$$a : C \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{I}},$$
$$(a, b) : R \text{ iff } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}},$$
$$(a, x) : f \text{ iff } (a^{\mathcal{I}}, x^{\mathcal{I}}) \in f^{\mathcal{I}},$$
$$a \neq b) \text{ iff } a^{\mathcal{I}} \neq b^{\mathcal{I}},$$
$$(x_1, \ldots, x_n) : P \text{ iff } (x_1^{\mathcal{I}}, \ldots, x_n^{\mathcal{I}}) \in P^{\mathcal{D}}.$$

An interpretation $\mathcal{I} \models \mathcal{A}$, for which we call $\mathcal{I}$ a *model* of the ABox, iff $\mathcal{I} \models \phi$ for all $\phi \in \mathcal{A}$. $\mathcal{A}$ is *consistent* iff it has a model. The canonical model is built up from $\mathcal{A}$ by determining a logical structure from all assertions of the final ABox, which were added by rules application during the construction process. Note that we can construct different models, when we specify different task-dependent constraints in $\mathcal{A}$ at the beginning of the model finding process.

3 Configuration design problem

In this section we start with defining the configuration design problem before we give an example from our structural engineering domain. We use the notion of constructing an admissible configuration in the ABox from a conceptual description in the TBox as first defined in [1].

Definition 6 (configuration design problem) A *configuration design problem* $P_C = (\mathcal{T}, \mathcal{A}_0)$ consists of a TBox $\mathcal{T}$ and an initial ABox $\mathcal{A}_0$. In the TBox a conceptual description of possible configurations is decribed. The initial ABox specifies the design constraints, the goal object and other necessary information for the model construction process. A solution can be constructed if a *finite model* for P_C exists. This model represents an textitadmissible configuration.

Depending on the stated design constraints in the initial ABox, the algorithm computes different models for a fixed TBox by introducing new assertions into the ABox. The algorithm makes thereby all implicit knowledge from $\mathcal{T}$ explicit. It computes first a complete ABox $\mathcal{A}$. Afterwards the canonical model can be built on the domain. The task-dependent requirements stated in $\mathcal{A}_0$ reduce thereby the possible solutions, which can be generated on $\mathcal{T}$. For computing a canonical model as solution to our configuration problem P_C, we use an algorithm as described in [13], which uses concept satisfiability for testing consistency of $\mathcal{T}$ and $\mathcal{A}$.

Example Our configuration problem is a simple frame design. In $\mathcal{T}$ all requirements regarding necessary parts, the layout of the components and the allowable concrete domains, which limit structural values to certain intervals or single concrete objects, are defined. Some problem instance specific constraints on the solution together with the frame object, which has to be configured, are stated in $\mathcal{A}_0$ before the construction process starts.

We begin with the definition of the system and descend the hierarchy top-down from the frame to the load definition. All frames have some common components. Two overall frame solutions are possible depending on the constraint in $\mathcal{A}_0$. The structural engineer is mainly interested in a valid topology. Numeric values are usually not considered at the conceptual design stage, unless they violate stated geometrical or structural constraints. If this is the case, the conceptual description of single component parameters or terminals has to be rewritten in $\mathcal{T}$. We do not take this case into account because we assume a fixed conceptual description during the solution process.

The following concept definitions draw also on the aforehand defined concrete domains from definition 1. The equilibrium of the components' terminals is specified by the predicates $+, =$ and $\leq$ over the concrete domain for the dimension force, while the predicate $\leq$ is defined over a separate concrete domain for the dimension length. The predicate *depth-range* and *force-range* denote unary predicates that restricts the numeric value for the dimension length and force to be in the interval of $I_1 = [0.4, 0.7]$ and $I_2 = [5, 20]$. The *force-range* predicate restricts the applicability of the solution method to reasonable loading levels. The lower bound of the *depth-range* constraint is stipulated in the code and the upper bounded by economical considerations.

$$
\begin{aligned}
StructuralSystem \doteq\ & Frame \sqcap \\
& (InTerminal \downarrow \\
& Beam1\ InTerminal) \sqcap \\
& (OutTerminal \downarrow \\
& Column1\ OutTerminal) \sqcap \\
& (OutTerminal \downarrow \\
& Column2\ OutTerminal) \sqcap \\
& (= 1\ loadedBy.Load) \\
Frame \doteq\ & ((\neg Slender \sqcup RigidFrame) \sqcup \\
& (\neg Cheap \sqcup SimpleFrame)) \sqcap \\
& (= 1\ hasPart.Beam) \sqcap \\
& (= 2\ hasPart.Column) \sqcap \\
& (= 2\ hasPart.Support) \sqcap \\
& (= 4\ hasPart.Connection)
\end{aligned}
$$

$$
\begin{aligned}
RigidFrame \doteq\ & (= 4\ hasPart.Rigid) \sqcap \\
& (= 2\ hasPart.Fixed) \\
SimpleFrame \doteq\ & (= 2\ hasPart.Rigid) \sqcap \\
& (= 2\ hasPart.Hinged) \sqcap \\
& (= 2\ hasPart.Simple) \\
Component \sqsubseteq\ & VerticalMember \sqcup Beam \\
VerticalMember \doteq\ & (Column \sqcup Connection \sqcup Support) \sqcap \\
& \exists hasTerminal\ action, \\
& hasTerminal\ reaction\ . = \\
Beam \doteq\ & Bending \sqcap \\
& \exists beamDepth.depth{-}range \sqcap \\
& \exists beamLength.\ \leq_{10}\ \sqcap \\
& \exists hasTerminal\ reaction1, \\
& hasTerminal\ reaction2, \\
& hasTerminal\ action.+ \\
Column \doteq\ & Compression \sqcap \\
& \exists columnDepth.depth{-}range \sqcap \\
& \exists columnLength.\ \leq_{4}\ \sqcap \\
Support \doteq\ & Simple \sqcup Fixed \\
Connection \doteq\ & Hinged \sqcup Rigid \\
Terminal \sqsubseteq\ & \neg Component \sqcap \neg StructuralSystem \sqcap \neg Load \sqcap \\
& (InputTerminal \sqcup OutputTerminal) \\
InputTerminal \doteq\ & \exists action.\ \geq_0 \\
OutputTerminal \doteq\ & \exists reaction.\ \geq_0\ \sqcap \\
& (= 1\ connected.InTerminal) \\
Load \doteq\ & \exists concentratedForce.force{-}range \sqcap \\
& \exists hasTerminal\ action.\ \geq_0
\end{aligned}
$$

The task specific assertions are given in the initial ABox $\mathcal{A}_0$.

$$
\begin{aligned}
&\{FRAMEI\colon Frame, FRAMEI\colon Slender, \\
&VLOAD\colon Load, \\
&(FRAMEI, VLOAD)\colon loadedBy, \\
&(BEAMI, x_1)\colon beamLength, (x_1)\colon =_8, \\
&(BEAMI, x_2)\colon beamDepth, (x_2)\colon =_{0.5}, \\
&(COLUMN1, x_3)\colon columnLength, (x_3)\colon =_3,
\end{aligned}
$$

$$
\begin{aligned}
&(FRAMEI, TERM1)\colon \mathit{InTerminal},\\
&(VLOAD, TERM2)\colon \mathit{hasTerminal},\\
&(TERM2, TERM1)\colon \mathit{connected},\\
&(VLOAD, x_4)\colon \mathit{concentratedForce}, (x_4)\colon =_{10}\}
\end{aligned}
$$

We introduce $x_1, \ldots, x_3$ as concrete objects and x_4 as a concrete object over another concrete domain, respectively. The equality predicates restrict structural values to single objects from the concrete domains.

An algorithm for testing consistency of the configuration problem P_C can compute a complete ABox $\mathcal{A}$ [15]. If the configuration problem is inconsistent, namely not clash free, there might be two distinct cases. First, constraints were stated in $\mathcal{A}_0$, which are inconsistent on $\mathcal{T}$ without introducing new objects during the construction process at all, being the case for contradictory task-specific constraints in $\mathcal{A}_0$. Second, no solution to P_C can be found on the completed ABox $\mathcal{A}$. If the ABox is clash free, a canonical model can be constructed from the generated object names in $\mathcal{A}$. We give only some typical excerpts from the whole model due to space restriction and restrict us to one half of the frame because of symmetry. We do not specify values for the concrete objects from the domain Δ_R because the actual force distribution is not taken into account by practitioners in the conceptual design process. The real force distribution is usually afterwards calculated by structural analysis tools.

$$
\begin{aligned}
&\Delta_{\mathcal{I}} = \{FRAMEI, BEAMI, COLUMN1, FIXED1, RIGID_i, TERM_j, \ldots\},\\
&\Delta_R = \{force1, \ldots, force20\}, \Delta_{A1} = \{10\}, \Delta_{A2} = \{0.5, 3, 8\},\\
&\mathit{StructuralSystem}^{\mathcal{I}}, \mathit{RigidFrame}^{\mathcal{I}} = \{FRAMEI\},\\
&\mathit{Beam}^{\mathcal{I}} = \{BEAM1\}, \mathit{Column}^{\mathcal{I}} = \{COLUMN1\}, \text{etc.},\\
&\mathit{InputTerminal}^{\mathcal{I}} = \{TERM1, \ldots, TERM9\},\\
&\mathit{OutputTerminal}^{\mathcal{I}} = \{TERM2, \ldots, TERM11\},\\
&\mathit{InTerminal}^{\mathcal{I}} = \{(FRAMEI, TERMI), (BEAMI, TERMI), \ldots\}, \text{etc.},\\
&\mathit{hasPart}^{\mathcal{I}} = \{(FRAMEI, BEAMI), (FRAMEI, COLUMN1), \ldots\},\\
&\mathit{hasTerminal}^{\mathcal{I}} = \{(BEAMI, TERM1), (COLUMN, TERM3), \ldots\},\\
&\mathit{action}^{\mathcal{I}} = \{(TERM1, force1), (TERM3, force3), \ldots\},\\
&\mathit{reaction}^{\mathcal{I}} = \{(TERM2, force2), (TERM4, force4), \ldots\},\\
&\mathit{connected}^{\mathcal{I}} = \{(TERM3, TERM4), (TERM2, TERM1), \ldots\}.
\end{aligned}
$$

The constructed model $\mathcal{I}$ from the complete and clash free ABox is a solution, which describes the actual assembly in terms of the required parts being the specific components and terminals, the components attributes as well as the resulting topology.

4 Conclusion

We have shown that configuration design problems from the domain of structural engineering can be solved by model construction in a description logic like $\mathcal{ALCQF}(\mathcal{D})$, which possesses the finite model property. We focused on the computation of parts, namely components and terminals of the structural system on different hierachical levels, and of a layout that ensures channeling the loads within the structure safely to the ground. Especially, the topological requirements on the structure required feature aggreement in the DL. As far as we know, regular configuration languages cannot represent topological requirements yet, which are important if the structure of the configuration cannot be considered fixed prior the model construction approach.

In addition, we found that only a subset of possible concept expressions,which are defined by the syntax of $\mathcal{ALCQF}(\mathcal{D})$, was used for the definition of the structural description in the TBox. It might be possible to take such restriction into account for increasing the expressiveness of the description logic but still obtaining decidable inference problems. Hence, the model construction approach extends to subsets of even more expressive description logics with infinite models, as long as the employed subset used for the formulation of the configuration problem has the finite model property.

Because intuitively many finite solutions exist for usual configuration design problems from engineering domains, it might be a further research direction to identify more clearly a decidable subset of very expressive decription logics, which cover the representation requirements from engineering configuration domains.

References

[1] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. In *Proceedings of the German AI Conference, KI'94*, volume 861 of *Lecture Notes in Computer Science*, pages 51–62, Saarbrücken (Germany), 1994. Springer–Verlag.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P Patel-Schneider. *Description Logic Handbook*. Cambridge University Press, 2002.

[3] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. DFKI Research Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, 1991.

[4] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proceedings of the 16th German AI-*

Conference, GWAI-92, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn (Germany), 1993. Springer–Verlag.

[5] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research (JAIR)*, 16:1–58, 2002.

[6] F. Baader and U. Sattler. Expressive number restrictions in description logics. *Journal of Logic and Computation*, 9(3):319–350, 1999.

[7] D. C. Brown. Defining configuring. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12(4):301–305, March 1997.

[8] D. C. Brown and B. Chandrasekaran. *Design Problem Solving, Knowledge Structures and Control Strategies*. Morgan Kaufmann Publishers, 1989.

[9] M. Buchheit, R. Klein, and W. Nutt. Constructive problem solving: A model construction approach towards configuration. DFKI technical memo TM-95-01, Deutsches Forschungszentrum für Künstliche Intelligenz Saarbrücken, January 1995.

[10] G. Friedrich and M. Stumptner. Consistency-based configuration. In *Proceedings of the AAAI '99 Workshop on Configuration*, 1999.

[11] O. Hollmann, Th. Wagner, and A. Günter. Engcon: A flexible domain-independent configuration engine. In *Proceedings of the ECAI-Workshop Configuration, ECAI-2000*, 2000.

[12] C. Lutz. Reasoning with concrete domains. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, pages 90–95, Stockholm, Sweden, July 31 – August 6, 1999. Morgan-Kaufmann Publishers.

[13] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[14] D. McGuinness and J. Wright. An industrial strength description logic-based configurator platform. *IEEE Intelligent Systems*, 13(4):69–77, July 1998.

[15] R. Möller. Expressive Description Logics: Foundations for Practical Applications. Faculty of Computer Science, University of Hamburg, 2000. Habilitation.

[16] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 1998.

[17] C. Schröder, R. Möller, and C. Lutz. A partial logical reconstruction of PLAKON/KONWERK. In *Proceedings of the Workshop on Knowledge Representation and Configuration WRKP-96*, pages 55–64, 1996.

[18] B. Wielenga and G. Schreiber. Configuration-design problem solving. *IEEE Intelligent Systems*, 12(2):49–56, March 1997.