Reasoning about Assembly Sequences Based on Description Logic and Rule

Yu Meng, Tianlong Gu, Liang Chang

▶ **To cite this version:**

Yu Meng, Tianlong Gu, Liang Chang. Reasoning about Assembly Sequences Based on Description Logic and Rule. 7th International Conference on Intelligent Information Processing (IIP), Oct 2012, Guilin, China. pp.131-136, 10.1007/978-3-642-32891-6_18 . hal-01524945

HAL Id: hal-01524945

https://inria.hal.science/hal-01524945

Submitted on 19 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Reasoning about Assembly Sequences Based on Description Logic and Rule

Yu Meng[1,2], Tianlong Gu[2], Liang Chang[2]

[1]School of Electronic Engineering, Xidian University, Xi'an 710071, China
[2]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

mengyucoco@163.com,cctlgu@guet.edu.cn,changl@guet.edu.cn

Abstract. Reasoning about assembly sequences is useful for identifying the feasibility of assembly sequences according to the assembly knowledge. Technologies used for reasoning about assembly sequences have crucial impacts on the efficiency and automation of assembly sequence planning. Description Logic (DL) is well-known for representing and reasoning about knowledge of static application domains; it offers considerable expressive power going far beyond propositional logic while reasoning is still decidable. In this paper, we bring the power and character of description logic into reasoning about assembly sequences. Assembly knowledge is firstly described by a description logic enhanced with some rules. Then, the feasibility of assembly operations is decided by utilizing the reasoning services provided by description logics and rules. An example has been provided to demonstrate the usefulness and executability of the proposed approach.

Keywords: assembly sequence; knowledge representation; reasoning; description logic; rule

1 Introduction

Related researches show that 40%~50% of manufacturing cost is spent on assembly, and 20%~70% of all the manufacturing work is assembly[1]. Assembly sequence is the most important part of an assembly plan. Reasoning about assembly sequences is useful for identifying the feasibility of assembly sequences according to the assembly knowledge. Technologies used for reasoning about assembly sequences have crucial impacts on the efficiency and automation of assembly sequence planning.

Bourjault[2] and De Fazio and Whitney[3] have developed the structured methodologies in which a series of Yes-or-No questions must be answered to generate the feasible sequences. In both cases, it will become a difficult and error-prone process for all but simplest assemblies. While Homem de Mello et al.[4] presented the cut-set method to finding the feasible assembly operations. This method suffers from the fact that there may be an exponential number of candidates partitioning to test, even though few satisfy physical constraints. Gottipolu and Ghosh[5] developed an algo-

rithmic procedure to generate all feasible assembly sequences by representing the contact and translation function into truth tables and then applying the Boolean algebra principles. It does not preclude that the amount of effort required also increases dramatically with the number of parts in product.

In recent years, due to the strong expression power and the decidability of reasoning, Description Logic (DL)[6] has been emphasized by more and more researchers on knowledge representation and reasoning. But it is hard for DL to define multiple, concurrent conditions rules. However, the multiple, concurrent conditions rules must be represented to implement reasoning in some application domains. For example, Fiorentini et al.[7] and Zhu et al.[8] have proposed the approaches based on DL and rule in the product assembly domain in order to satisfy design tasks such as verifying conditions for design completeness, product qualification and requirements.

In reasoning about assembly sequences, multiple constraints (e.g. connectivity constraints, precedence constraints) must be considered to verify the feasibility of assembly. In other words, the rules for judging the feasibility of assembly are multiple, concurrent conditions rules. In order to improve the level of reasoning automation about assembly sequences, this paper proposes an approach based on DL and rule. In this approach, we bring the power and character of description logic enhanced with some rules into reasoning about assembly sequences.

2 Description logic and rule representation

Description Logic (DL)[6] is a knowledge representation formalism and it is a decidable subset of first-order logic (FOL). In many of the formal methods for representing knowledge, DL has received particular attention in the recent years. The cause is mainly that it is highly effective for concept hierarchy and providing reasoning service. The knowledge base (KB) based on DL is partitioned into an assertional part (ABox) and a terminological part (TBox). In DL, a distinction is drawn between TBox and ABox. In general, TBox is a set defining concepts, relationships among concepts, and relationships among relationships, which is an axiom set describing domain structure. ABox describes which concept an individual belongs to and what relationship one individual have with another individual.

However, there is the scarcity of DL in reasoning rule representation, that is, it is hard for DL to define multiple, concurrent conditions rule. For example, the follow rule which states that x1 is the father of x2 and x3 is the brother of x1, then there exists that x3 is the uncle of x2, can't be expressed in DL.

$$\text{hasFather}(x2, x1) \wedge \text{hasBrother}(x1, x3) \rightarrow \text{hasUncle}(x2, x3)$$

To offer sophisticated representation and reasoning capabilities, the integration of DL knowledge bases and rule expression representation is necessary. It is one of the methods that the rule-based systems use vocabulary specified in DL knowledge bases.

3 The DL representation of assembly knowledge

Assembly sequence planning begins with representing the assembly knowledge that

can be extracted directly from the CAD model of assembly. Gottipolu and Ghosh[5] represented the assembly knowledge as two types of uni-directional matrices, which are called the contact and the translation functions. The contact function C_{ab}= (C_1, C_2, C_3, C_4, C_5, C_6) is a 1×6 binary function representing contacts between the parts *a* and *b*. It can be defined as $C_{ab} : C_i \rightarrow \{0, 1\}$, i = 1~6, where C_i = 1 indicates presence of contact in the direction *i*, i.e. part *b* is in contact with part *a* in the direction *i*; C_i = 0 indicates absence of contact in that direction. The freedom of translation motion between two parts in an assembly can be defined by a 1×6 binary translation function T_{ab} = (T_1, T_2, T_3, T_4, T_5, T_6) or $T_{ab} : T_i \rightarrow \{0, 1\}$, i = 1~6. If the part *b* has the freedom of translation motion with respect to the part *a* in the direction *i*, then T_i = 1, else T_i = 0. Here, direction 1, 2 and 3 indicate the positive sense of X, Y and Z axes (X+, Y+ and Z+) respectively, whereas direction 4, 5 and 6 correspond to the negative sense of X, Y and Z axes (X−, Y− and Z−) respectively.

Given an assembly and its contact and translational functions, we can represent the assembly model by defining concepts, roles and creating assertions in DL. The related concepts and roles in DL are defined as follows:

(1) Each assembly is made up of several parts. The **Part** is an atomic concept, the concept **Part** will be defined to represent parts in the assembly.

(2) For the contact functions, we define six atomic roles $\mathbf{DC_1}$, $\mathbf{DC_2}$, $\mathbf{DC_3}$, $\mathbf{DC_4}$, $\mathbf{DC_5}$ and $\mathbf{DC_6}$, called contact roles, corresponding to C_1, C_2, C_3, C_4, C_5 and C_6 in C_{ab}.

(3) We represent the translation functions as six atomic roles $\mathbf{DT_1}$, $\mathbf{DT_2}$, $\mathbf{DT_3}$, $\mathbf{DT_4}$, $\mathbf{DT_5}$ and $\mathbf{DT_6}$, called translation roles.

For example, if *a* and *b* are individual names, then the assertion Part(*a*) means that *a* is a part, and Part(*b*) means that *b* is a part. For the above part *a* and part *b*, if *b* is in contact with *a* in the direction *i* (*i* = 1~6), there exist the assertion DC_i (*a*, *b*). In the same way, if *b* has the freedom of translational motion with respect to *a* in the direction *i* (*i* = 1~6), there exist the assertion DT_i(*a*, *b*).

4 The representation of reasoning rule of assembly

To verify the feasibility assembly, two types of constraints, connectivity constraints and precedence constraints, must be considered. We verify these constraints by representing the contact and translational relations into DL roles and then applying reasoning rules. These reasoning rules are expressed as multiple, concurrent conditions rules. The related inference will use vocabulary specified in DL knowledge bases.

For representing reasoning rules, some concepts and roles will be defined in DL. These concepts include **ComponentSeq**, **Subassembly**, while these roles include **FeasibleAssemblyRole**, **hasLeftComponent** and **hasRightComponent**. **ComponentSeq** and **Subassembly** denote the sequence of components and subassembly respectively. **FeasibleAssemblyRole** means that two components can be assembled. **hasLeftComponent** and **hasRightComponent** denote which components compose the sequence of components or subassembly. There are concept assertions such as

Part(*a*), Part(*b*), Part(*c*),

ComponentSeq(*s1*), hasLeftComponent (*s1*, *a*), hasRightComponent (*s1*, *b*).

stating that the individuals named *a*, *b* and *c* are parts; *s1* is a sequence of components and assembled by *a* and *b*.

According to the above concepts and roles, the reasoning rule of subassembly is presented as follow.

$$\text{ComponentSeq}(x) \wedge \text{hasLeftComponent}(x, xL) \wedge \text{hasRightComponent}(x, xR) \wedge \text{FeasibleAssemblyRole}(xL, xR) \rightarrow \text{Subassembly}(x)$$

In the rest of this section, we represent all reasoning rules for determining the feasibility of assembly sequences as multiple, concurrent conditions rules.

Firstly, the feasibility of two part subassemblies can be verified from the roles DC_i and DT_i (i = 1~6) of that pair. For any pair (a, b) of parts, at least one of DC_i (a, b)(i = 1~6) assertions must exist and at least one of DT_i (a, b)(i = 1~6) assertions must exist to make that pair (a, b) a feasible subassembly. For describing these conditions, two roles **DRC** and **DRT** will be defined in DL knowledge bases, where $DC_i \sqsubseteq$ DRC and $DT_i \sqsubseteq$ DRT (i = 1~6) hold. The reasoning rule for verifying the feasibility of two component subassemblies is described as

$$\text{DRC}(p1, p2) \wedge \text{DRT}(p1, p2) \rightarrow \text{FeasibleAssemblyRole}(p1, p2)$$

Secondly, the feasibility of subassemblies with more than two parts, both roles DC_i and DT_i (i = 1~6) of the involved ordered pairs must be used. For example, the feasibility of assembly of the subassembly (a, b) and the part c is verified as explained as follows.

Step 1 Consider the subassembly (a, b) as a set $\{a, b\}$ and the part c to be added as another set $\{c\}$. The Cartesian product of these two sets ($\{a, b\} \times \{c\}$) gives a set of ordered pairs $\{(a, c), (b, c)\}$. For the pair (a, c) and (b, c), if the assertion DRC(a, c) or DRC(b, c) hold, then there is a contact between c and (a, b). So we define the role **DRTC** in DL knowledge bases and describe this condition as rule expressions such as

$$\text{Subassembly}(s) \wedge \text{hasLeftComponent}(s, sL) \wedge \text{DRC}(sL, p) \rightarrow \text{DRTC}(s, p)$$

$$\text{Subassembly}(s) \wedge \text{hasRightComponent}(s, sR) \wedge \text{DRC}(sR, p) \rightarrow \text{DRTC}(s, p)$$

Step 2 The presence of contact above provides only the necessary condition, but it does not guarantee the feasibility of assembly operation due to the precedence constraints. To consider the precedence constraints, the translation roles DT_i (i = 1~6) of the Cartesian ordered pairs must be used.

Considering the assembly of part c to the subassembly (a, b), there are two pairs (a, c) and (b, c). Firstly, if the DT_i (a, c) and DT_i (b, c) assertions hold, then the component c has collision-free disassembly in the directions i with respect to the subassembly (a, b). Secondly, if one of the six directions is a collision-free disassembly direction of the component c with respect to the subassembly (a, b), then the subassembly (a, b, c) is feasible. In this regard, the role **DRTT** is defined in DL knowledge bases and the above precedence conditions is presented as

$$\text{Subassembly}(s) \wedge \text{hasLeftComponent}(s, sL) \wedge \text{hasRightComponent}(s, sR) \wedge \text{DT}_i(sL, p) \wedge \text{DT}_i(sR, p) \rightarrow \text{DRTT}(s, p), \quad i = 1\sim6.$$

The same logic can be applied to the subassembly including multiple components and the single component.

Step 3 If the rules of **Step 1**and **Step 2** hold, then we can obtain the following reasoning rule for judging the feasibility of assembly.

$$\text{DRTC}(s, p) \wedge \text{DRTT}(s, p) \rightarrow \text{FeasibleAssemblyRole}(s, p)$$

5 The reasoning of assembly sequence

To demonstrate the usefulness of the proposed approach, we described the assembly knowledge and reasoning rules of the assembly shown in Fig. 1 by the languages OWL DL and SWRL with the help of the editing tool Protégé, and the reasoning is carried out by the JESS reasoning engine [9]. This example assembly includes four parts, *a*, *b*, *c* and *d* are the name of those parts. The process of generating assembly sequences for the example shown in Fig. 1 is explained as follows.

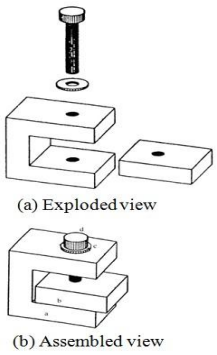

Fig. 1. An example assembly

Part(a), Part(b), Part(c), Part(d)

DC_4 (a, b), DC_5 (a, b), DT_1 (a, b), DT_3 (a, b), DT_6 (a, b)
DC_5 (a, c), DT_1 (a, c), DT_2 (a, c), DT_3 (a, c), DT_4 (a, c), DT_6 (a, c)

DC_1 (d, b), DC_2 (d, b), DC_4 (d, b), DC_6 (d, b), DT_5 (d, a)
DC_1 (d, c), DC_2 (d, c), DC_3 (d, c), DC_4 (d, c), DC_6 (d, c), DT_5 (d, c)

ComponentSeq (S_ab),
hasLeftComponent(S_ab, a), hasRightComponent(S_ab, b)
ComponentSeq (S_ac),
hasLeftComponent(S_ac, a), hasRightComponent(S_ac, c)

ComponentSeq (S_ab_c),
hasLeftComponent(S_ab_c, S_ab), hasRightComponent(S_ab_c, c)
ComponentSeq (S_ab_d),
hasLeftComponent(S_ab_d, S_ab), hasRightComponent(S_ab_d, d)

ComponentSeq (S_abc_d),
hasLeftComponent(S_abc_d, S_abc), hasRightComponent(S_abc_d, d)
ComponentSeq (S_abd_c),
hasLeftComponent(S_abd_c, S_abd), hasRightComponent(S_abd_c, c)

Fig. 2. The ABox of the example shown in Fig. 1

According to the representation of assembly knowledge and reasoning rules given in Section 3 and 4, the ABox of the example shown in Fig. 1 is derived as Fig. 2. After using JESS engine to make inference about OWL individuals of Fig. 2 over the total reasoning rules, **Subassembly** class stores SWRL inference results, that is **Subassembly** instances are generated accordingly. These **Subassembly** instances include:

(1) *S_ab*, *S_ac*, *S_ad*, *S_bd* and *S_cd*, which denote the feasible assembly tasks are {(*a*), (*b*)}, {(*a*), (*c*)}, {(*a*), (*d*)}, {(*b*), (*d*)}, {(*c*), (*d*)}, and the corresponding feasible two part subassemblies are (*a, b*), (*a, c*), (*a, d*), (*b, d*), and (*c, d*);

(2) *S_ab_c*, *S_ac_b*, *S_ab_d*, *S_ac_d*, *S_cd_a* and *S_cd_b*, which denote the feasible assembly tasks are {(*a, b*), (*c*)}, {(*a, c*), (*b*)}, {(*a, b*), (*d*)}, {(*a, c*), (*d*)}, {(*c, d*), (*a*)} and{(*c, d*), (*b*)}, and the corresponding feasible three part subassemblies are (*a, b, c*), (*a, b, d*), (*a, c, d*) and (*b, c, d*).

(3) *S_abc_d* and *S_acb_d*, which denote the feasible assembly task is {(*a, b, c*), (*d*)}, and the corresponding feasible four part subassembly is (*a, b, c, d*).

Through analyzing the above results, the two part subassemblies (*a, d*) and (*b, d*) are not used in any tasks in the subsequent assembly, that is, no further higher order subassemblies can be generated from these subassemblies, so (*a, d*) and (*b, d*) are invalid subassemblies and should be deleted. Similarly, the (*a, b, d*), (*a, c, d*) and (*b, c, d*) should be deleted. After deleting all the invalid subassemblies, the resulting feasible subassemblies are (*a, b*), (*a, c*), (*c, d*), (*a, b, c*), (*a, b, c, d*), and the corresponding feasible assembly tasks are {(*a*), (*b*)}, {(*a*), (*c*)}, {(*c*), (*d*)}, {(*a, b*), (*c*)}, {(*a, c*), (*b*)} and {(*a, b, c*), (*d*)}. Then all the feasible assembly sequences are obtained as follows:

(1) ({{*a*} {*b*} {*c*} {*d* }} {{*a, b*} {*c*} {*d* }} {{*a, b, c*} {*d* }} {{*a, b, c, d* }});
(2) ({{*a*} {*b*} {*c*} {*d* }} {{*a, c*} {*b*} {*d* }} {{*a, b, c*} {*d* }} {{*a, b, c, d* }});
(3) ({{*a*} {*b*} {*c*} {*d* }} {{*a, b*} {*c*} {*d* }} {{*a, b*} {*c, d* }} {{*a, b, c, d* }});
(4) ({{*a*} {*b*} {*c*} {*d* }} {{*a*} {*b*} {*c, d* }} {{*a, b*} {*c, d* }} {{*a, b, c, d* }}).

6 Conclusion

Description Logic (DL) is well-known for representing and reasoning about knowledge of static application domains. But it is hard for DL to define multiple, concurrent conditions rules. In order to improve the level of reasoning automation about assembly sequences, we firstly describe the knowledge on assembly model by DL and establish a rule set of the assembly reasoning rules. Then, the feasibility of assembly operations is decided by utilizing the reasoning services provided by description logics and rules. Finally, an example has been provided to demonstrate the usefulness and executability of the proposed approach.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 60903079, 60963010) and the Guangxi Key Laboratory of Trusted Software.

References

1. Wang,J.F., Li,S.Q., Liu,J.H., et al.: Computer aided assembly planning: a survey. Journal of Engineering Graphics.26(2),1-6(2005)
2. Bourjault,A. : Contribution à une approche méthodologique de l'assemblage automatisé élaboration automatique des séquences opératoires. Thèse d'État, Université de Franche-Comté, Besancon, France(1984)
3. De Fazio,TL., Whitney,DE. : Simplified generation of all mechanical assembly sequences. IEEE Journal of Robotics and Automation. RA-3(6),640–658(1987)
4. Homem de Mello,LS., Sanderson,AC.: A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Transactions on Robotics and Automation. 7(2),228–240(1991)
5. Gottipolu,RB., Ghosh,K.: A simplified and efficient representation for evaluation and selection of assembly sequences. Computers in Industry.50(3),251-64(2003)
6. Baader,F., McGuinness,D., et al.: The description logic handbook: theory, implementation and applications. Cambridge University Press(2003)
7. Fiorentini,X., Rachuri,S., et al. : An Evaluation of Description Logic for the Development of Product Models (2008)
8. Zhu,L., Jayaram,U., Kim,O.: Semantic Applications Enabling Reasoning in Product Assembly Ontologies—Moving Past Modeling. Journal of Computing and Information Science in Engineering. 12(1)(2012)
9. Dizza,B.: Using OWL and SWRL to represent and reason with situation-based access control policies. Data and Knowledge Engineering.70(6), 596-615(2011)