Symbolic ZBDD Representations for Mechanical Assembly Sequences

Fengying Li, Tianlong Gu, Guoyong Cai, Liang Chang

Symbolic ZBDD Representations for Mechanical Assembly Sequences

Fengying Li, Tianlong Gu, Guoyong Cai, Liang Chang

Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
{ lfy, cctlgu, ccgycai, changl }@guet.edu.cn

Abstract. The representations of assembly knowledge and assembly sequences are crucial in assembly planning, where the size of parts involved is a significant and often prohibitive difficulty. Zero-suppressed binary decision diagram (ZBDD) is an efficient form to represent and manipulate the sets of combination, and appears to give improved results for large-scale combinatorial optimization problems. In this paper, liaison graphs, translation functions, assembly states and assembly tasks are represented as sets of combinations, and the symbolic ZBDD representation of assembly sequences is proposed. An example is given to show the feasibility of the ZBDD-based representation scheme.

Keywords: assembly sequence; assembly knowledge; Zero-suppressed binary decision diagram

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, 2]. In order to shorten the time and reduce the costs required for the development of the product and its manufacturing process, it is desirable to automate and computerize the assembly sequence planning activity. Typically, a product can have a very large number of feasible assembly sequences even at a small parts count, and this number rises exponentially with increasing parts count, which renders it staggeringly difficult and even impossible for one to represent all the sequences individually. Thus, the choices of representation for assembly sequences can be crucial in assembly sequence planning, and there has been a need to develop systematic and efficient methods to represent all the available alternatives.

In the literature, several representation schemes have been proposed to represent the assembly sequences. These representations can be classified into two groups: ordered lists and graphical representations. The ordered list could be a list of tasks, list of assembly states, or list of subsets of connections. In the ordered lists each assembly sequence is represented by a set of lists. Although this set of lists might represent a complete and correct description of all feasible assembly sequences, it is not necessarily the most compact or most useful representation of sequences. The graphi-

cal schemes map the assembly operations and assembly states into specified diagrammatic elements, and share common subsequences and common states graphically in many assembly sequences, which create more compact and useful representations that can encompass all feasible assembly sequences. The most common diagrammatic representation schemes are: precedence diagrams[3], state transition diagrams[4], inverted trees[5], liaison sequence graphs[6], assembly sequence graphs[7] directed graphs[2] and AND/OR graphs[8] etc.

In recent years, implicitly symbolic representation and manipulation technique, called as symbolic graph algorithm or symbolic algorithm[9], has emerged in order to combat or ease combinatorial state explosion. Typically, zero-suppressed binary decision diagram (ZBDD) is used to represent and manage the sets of combinations[10, 11]. Efficient symbolic algorithms have been devised for hardware verification, model checking, testing and optimization of circuits. Symbolic representations appear to be a promising way to improve the computation of large-scale combinatorial computing problems through encoding and searching nodes and edges implicitly.

In this regard, we present the symbolic ZBDD formulation of all the assembly sequences. The subassemblies, assembly states, assembly tasks and assembly sequences are represented by sets of combinations, and the ZBDD representation for them are given. The example shows that the ZBDD formulation is feasible and compact.

2 Zero-suppressed binary decision diagram

Zero-suppressed binary decision diagram (ZBDD)[10, 11], a variant of ordered binary decision diagram (OBDD), is introduced by Minato for representing and manipulating sets of combinations efficiently. With ZBDDs, the space requirement of the representation of combination sets is reduced and combinatorial problems are solved efficiently.

A combination on n objects can be represented by an n bit binary vector, ($x_n x_{n-1} \ldots x_2 x_1$), where each bit, $x_k \in \{0, 1\}$, expresses whether the corresponding object is included in the combination or not. A set of combinations can be represented by a set of the n bit binary vectors. We call such sets combination sets. Combination sets can be regarded as subsets of the power set on n objects.

A set of combination can be mapped into Boolean space by using n-input variables for each bit of the combination vector. If we choose any one combination vector, a Boolean function determines whether the combination is included in the set of combinations. Such Boolean functions are called characteristic functions.

By using OBDDs for characteristic functions, we can manipulate sets of combinations efficiently. Because of the effect of node sharing, OBDDs represent combination sets with a huge number of elements compactly. However, there is one inconvenience in that the form of OBDDs depends on the number of input variables. For example, S (*abc*) and S (*abcd*), shown in Fig. 1(a), represent the same set of combinations {*a*, *b*}, if we ignore the irrelevant input variables *c* and *d*. In this case, the OBDDs for S (*abc*) and S (*abcd*) are not identical. This inconvenience comes from the difference in the model of default variables. In combination sets, default variables are regarded as zero

when the characteristic function is true, since the irrelevant object never can be suppressed in the OBDD representation.

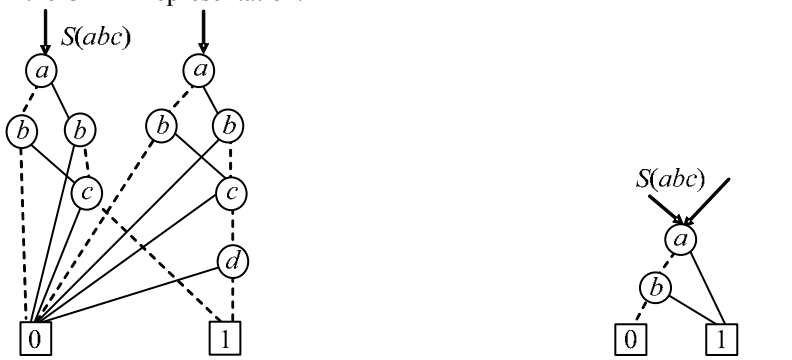


Fig. 1. (a) OBDDs for sets of combinations (b) ZBDDs for sets of combinations

For representing sets of combinations efficiently, ZBDD introduced the following special deletion rules:

·Delete all nodes whose 1-edge points to the 0-terminal node, and then connect the edges to the other sub-graph directly as shown in Fig. 2.

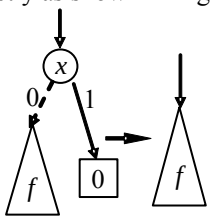


Fig. 2. New reduction rule for ZBDDs

This is also called the *pD-deletion* rule. ZBDD does not delete the nodes whose two edges point to the same node, which used to be deleted by OBDD. The zero-suppressed deletion rule is asymmetric for the two edges, as we do not delete the nodes whose 0-edge points to a 0-terminal node.

Fig. 1(b) shows the ZBDD representation for the same sets of combinations shown in Fig. 1(a). The form of ZBDDs is independent of the input domain. The ZBDD node deletion rule automatically suppresses the variables which never appear in any combination. This feature is important when we manipulate sparse combinations.

Another advantage of ZBDDs is that the number of l-paths in the graph is exactly equal to the number of elements in the combination set. In OBDDs, the node elimination rule breaks this property. Therefore, ZBDDs are more suitable than OBDDs to represent combination sets.

3 Symbolic representation of assembly knowledge

A mechanical assembly is a composition of interconnected parts forming a stable unit. Each part is a solid rigid object, that is, its shape remains unchanged. Parts are inter-

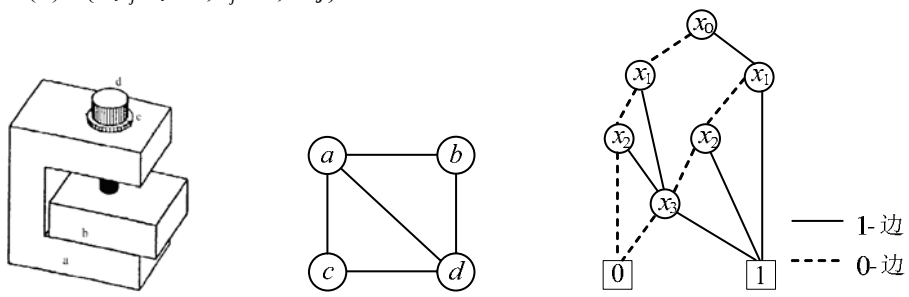
connected whenever they have one or more compatible surfaces in contact. Surface contacts between parts reduce the degree of freedom for relative motion. These contacts and relative motions are embedded in various logical and physical relations among the parts of the assembly, called as assembly knowledge, and can be extracted directly from the CAD model of assembly.

3.1 Symbolic representation of liaison graph

Liaison graph can explicitly describe various logical and physical contact relations among the parts of the assembly. Liaison graph is a two-tuples $G = (P, L)$, in which P is a set of nodes that represent parts, and L a set of edges that represent any of certain user-defined relations between parts called liaisons. User-accepted definitions of liaisons in a general sense follow the principal literal definition "a close bond or connection" and generally include physical contacts.

Given an assembly and its liaison graph $G = (P, L)$, we can convert the liaison graph to a ZBDD by encoding the parts of the assembly or the elements in P with n binary variables $X=(x_0, x_1, \ldots, x_{n-1})$, where $n = |P|$. Essentially, the liaison set or edge set is a relation on nodes, and each element or a liaison $(a, b) \in L$ is a pair. A liaison $(a, b) \in L$ can be encoded as a combination of binary variables (x_i, x_j), where x_i and x_j are the encoded binary variables corresponding to part a and b respectively. Thus, the liaison graph can be uniquely determined by the following set of combinations:

$C(X) = \{ x_i x_j \mid x_i \in X, x_j \in X, i \neq j \}$



(a) Assembly view (b) Liaison graph (c) ZBDDs for liaison graph

Fig. 3. An example of assembly

For example, an assembly shown in Fig. 3(a) includes 4 parts, and its liaison graph is presented in Fig. 3(b), where $L = \{(a, b), (a, c), (a, d), (b, d), (c, d)\}$. We formulate the parts with 4 binary variables by encoding part a, b, c and d as x_0, x_1, x_2 and x_3 respectively. The combination set of relation E is derived as following:

$C(x_0, x_1, x_2, x_3) = \{x_0x_1, x_0x_2, x_0x_3, x_1x_3, x_2x_3\}$

Therefore, the liaison graph of the assembly is formulated by a ZBDD corresponding to the $C(x_0, x_1, x_2, x_3)$ as shown in Fig. 3(c).

3.2 Symbolic representation of translation function

The liaison graph provides only the necessary conditions but not sufficient to assembly two components. To be a feasible assembly operation, it is necessary that there is a collision-free path to assembly parts. Gottipolu and Ghosh [6] represented the relative motion between parts of the assembly as a translation function, from which the existence or absence of a collision-free path can be conveniently verified.

The freedom of translation motion between two parts *a* and *b* can be represented by $T_{ab} = (T_0, T_1, T_2, T_3, T_4, T_5)$, which is a 1×6 binary function. Hence, it is called the translation function or *T*-function. It can be defined as:

$$T_{ab} = T_i \rightarrow \{0, 1\},\ i = 0, 1, 2, 3, 4, 5$$

Where T_i=1 if the part *b* has the freedom of translation motion with respect to the part *a* in the direction *i*, T_i=0 if the part *b* has no freedom of translation motion with respect to the part *a* in the direction *i*. 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. If the part *b* has the freedom of translation motion with respect to the part *a* in the direction *i*, then the part *a* has the freedom of translation motion with respect to the part *b* in the direction ((*i*+3) mod 6). Hence, it is enough to give the front half part of the translation function.

Table 1. *T*-function for the assembly shown in Fig. 3(a)

Pair	*T*-function					
	T_0	T_1	T_2	T_3	T_4	T_5
(*a*, *b*)	1	0	1	0	0	1
(*a*, *c*)	1	1	1	1	0	1
(*a*, *d*)	0	1	0	0	0	0
(*b*, *c*)	1	1	1	1	0	1
(*b*, *d*)	0	1	0	0	0	0
(*c*, *d*)	0	1	0	0	0	0

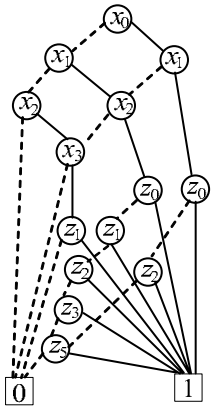


Fig. 4. ZBDDs for translation function

For example, the translation function *T* of the assembly shown in Fig. 3(a) is shown in table 1.

We can convert the translation function to a ZBDD by encoding the parts of the assembly in *P* with *n* binary variables $X=(x_0, x_1, \ldots, x_{n-1})$ and the direction with 6 binary variables $Z=(z_0, z_1, z_2, z_3, z_4, z_5)$, where $n = |P|$. We represent the translation function as set of combinations:

$$T(XZ) = \{ x_i x_j z_k \mid x_i \in X, x_j \in X, z_k \in Z, i \neq j \}$$

The combination set of translation function includes all the pairs (*a*, *b*), between which there exists the freedom of translation motion of part *b* with respect to the part

a in the direction i. We can construct the combination set of these translation relations as $T(XZ)$, and thus implicitly formulate the translation functions using ZBDDs.

For example, the translation relations of the assembly in Fig. 3(a) are derived as following:

$T(x_0, x_1, x_2, x_3, z_0, z_1, z_2, z_3, z_4, z_5) = \{x_0x_1z_0, x_0x_1z_2, x_0x_1z_5, x_0x_2z_0, x_0x_2z_1, x_0x_2z_2, x_0x_2z_3, x_0x_2z_5, x_0x_3z_1, x_1x_2z_0, x_1x_2z_1, x_1x_2z_2, x_1x_2z_3, x_1x_2z_5, x_1x_3z_1, x_2x_3z_1\}$

Fig. 4 gives the ZBDDs corresponding to the translation relations.

4 Symbolic representation of assembly sequences

A mechanical assembly is a composition of interconnected parts forming a stable unit. Each part is a solid rigid object, that is, its shape remains unchanged. Parts are interconnected whenever they have one or more compatible surfaces in contact. Surface contacts between parts reduce the degree of freedom for relative motion. It is assumed that whenever two parts are put together all contacts between them are established.

A subassembly consists of a unique part or some parts in which every part has at least one surface contact with another part. Although there are cases in which it is possible to join the parts in more than one way, an unique assembly geometry will be assumed for each subassembly. This geometry corresponds to their relative location in the whole assembly. A subassembly is said to be stable if its parts maintain their relative position and do not break contact spontaneously. All one-part subassemblies are stable. To formulate an assembly consisting n parts, n binary variables ($x_1, x_2, \ldots, x_n$) are demanded, if a part is characterized by a binary variable. Therefore, any subassembly can be characterized by the subset of parts set, and the ith component is presence or absence, respectively, if the nth part is involved in the subassembly or not. Hence, the assembly states can be represented by sets of combinations. For example, the assembly shown in Fig. 3 has four parts, represented by four binary variables x_1, x_2, x_3 and x_4, $\{abc\}$ and $\{cd\}$ are two subassemblies of the assembly, and can be represented by binary variable sets $\{x_1x_2x_3\}$ and $\{x_3x_4\}$ respectively. The initial state and final state are represented as sets of combinations $\{x_1, x_2, x_3, x_4\}$ and $\{x_1x_2x_3x_4\}$ respectively.

The assembly process consists of a succession of tasks, through each of which the subassemblies are joined into a larger subassembly. The process starts with all parts separated, and ends with all parts properly joined together to obtain the whole assembly. It is assumed that exactly two subassemblies are joined by each assembly task, and that after parts have been put together, they remain together until the end of the assembly process. An assembly task is said to be geometrically feasible if there is a collision-free path to bring the two subassemblies into contact from a situation in which they are far apart. And an assembly task is said to be mechanically feasible is it is feasible to establish the attachments that act on the contacts between the two subassemblies. Given two subassemblies characterized by their sets of parts S_1 and S_2, we say that joining S_1 and S_2 is an assembly task if the set $S_3 = S_1 \cup S_2$ characterizes a subassembly. Alternatively, a task can be seen as a decomposition of the output subassembly into the two input subassembly. Therefore, an assembly task τ_i can be char-

acterized by an ordered pair ($\{S_{i1}, S_{i2}\}$, S_{i3}) of its output subassembly and input subassemblies. Since the equation $S_{i3} = S_{i1} * S_{i2}$ holds, an assembly task τ_i can be represented by a set of combinations on $2n$ binary variables (X, Y) in which $X=(x_0, x_1,\ldots, x_{n-1})$ and $Y=(y_0, y_1,\ldots, y_{n-1})$ are the binary variables for subassembly S_{i1} and S_{i3} respectively. For example, for the assembly shown in Fig. 3, if $S_1 = \{ab\}$ and $S_2 = \{c\}$, then joining S_1 and S_2 is an assembly task τ. The assembly task τ is characterized by an ordered pair $\tau = (\{ab, c\}, \{abc\})$.

Given an assembly with n parts, an assembly sequence is an ordered list of n-1 assembly tasks $\sigma=\tau_1\tau_2\ldots\tau_{n-1}$, in which the input subassemblies of the first task τ_1 is the separated parts, the output subassembly of the last task τ_{n-1} is the whole assembly, and the input subassemblies to any task τ_i is either a one-part subassembly or the output subassemblies of a task that precedes τ_i. An assembly sequence is said to be feasible if all its assembly tasks are geometrically and mechanically feasible, and the input subassemblies of all tasks are stable. The second input subassembly of task τ_i can be deduced by $S_{i2}= S_{i3}/S_{i1}$, where S_{i1} is the first input subassembly of task τ_i, and S_{i3} is the output subassembly of task τ_i. In this regard, an assembly sequence can be represented by the following set of combinations:

$\varphi_\sigma(X, Y)= \{S_{i1}S_{i3}|\ S_{i1}$ is the first input subassembly of task τ_i, and S_{i3} is the output subassembly of task τ_i, $i = 1, 2,\ldots, n$-1, $\sigma=\tau_1\tau_2\ldots\tau_{n-1}\}$

For example, for the assembly shown in Fig. 3, if $\tau_1=(\{a, b\},\{ab\})$, $\tau_2=(\{ab, c\},\{abc\})$, $\tau_3=(\{abc,d\},\{abcd\})$, $\sigma=\tau_1\tau_2\tau_3$ is a feasible assembly sequence. The assembly sequence can be represented by the set of combinations:

$\varphi_\sigma(x_1, x_2,\ldots, x_n, y_1, y_2,\ldots, y_n)=\{ x_0y_0y_1, x_0x_1y_0y_1y_2, x_0x_1x_2y_0y_1y_2y_3\}$

The ZBDD of the assembly sequence is shown in Fig. 5.

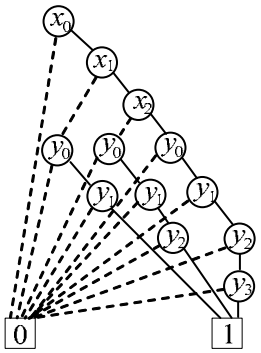


Fig. 5. ZBDD for assembly sequence

An assembly might have a number of different feasible assembly sequences, and many assembly sequences share common subsequences. All the feasible assembly sequences can be represented compactly by the following set of combinations:

$$\psi(X,Y) = \bigcup_{\sigma} \varphi(X,Y)$$

5 Conclusion

The choice of representation for assembly sequences has been crucial in assembly sequence planning. However, traditional representations, such as AND/OR graph and Petri nets, face the same challenge that increasing parts count renders it staggeringly difficult and even impossible to represent all the sequences individually. In order to alleviate the state-space explosion problem, a symbolic ZBDD scheme for representing all the feasible assembly sequences is presented in this paper. Validity and efficiency of this symbolic ZBDD-based assembly sequence representation are also demonstrated by the experiment.

Acknowledgments

The authors are very grateful to the anonymous reviewers for their helpful comments. This work has been supported by National Natural Science Foundation of China (No. 60903079, 60963010, 61063002), the Natural Science Foundation of Guangxi Province (No. 2012GXNSFAA053220), and the Project of Guangxi Key Laboratory of Trusted Software (No. PF11044X).

References

1. Gottipolu RB, Ghosh K.: A simplified and efficient representation for evaluation and selection of assembly sequences. Computers in Industry, 50(3): 251-264(2003)
2. Homem de Mello LS, Sanderson AC.: Representation of mechanical assembly sequences. IEEE Transactions on Robotics and Automation, 7(2): 211-227(1991)
3. Prenting T, Battaglin R.: The precedence diagram: A tool for analysis in assembly line balancing. Journal of Industrial Engineering, 15(4): 208-213(1964)
4. Warrats JJ, Bonschancher N, Bronsvoort WF.: A semi-automatic sequence planner. In: Proceedings of IEEE international conference on robotics and automation, pp: 2431-2438. IEEE, Piscataway(1992)
5. Bourjault A.: Contribution to a methodological approach of automated assembly: automatic generation of assembly sequence. Ph. D. Thesis, University of Franch-Comte, Besancon, France(1984)
6. De Fazio TL, Whitney DE.: Simplified generation of all mechanical assembly sequences. IEEE Journal of Robotics and Automation, RA-3(6): 640-658(1987)
7. Gottipolu RB, Ghosh K.: An integrated approach to the generation of assembly sequence. International Journal of Computer Applications in Technology, 8(3–4): 125-138(1995)
8. Homem de Mello LS, Sanderson AC.: AND/OR graph representation of assembly plans. IEEE Transactions on Robotics and Automation, 6(2): 188-199(1990)
9. Bryant RE.: Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys, 24(3): 293-318(1992)
10. Minato S.: Zero-suppressed BDDs for set manipulation in combinatorial problems. In: Proceedings of the 30th DAC in Dallas, pp: 272-277. IEEE, Piscataway(1993)
11. Minato S.: Fast factorization for implicit cube set representation. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 15(4): 377-384(1996)