第41卷 第2期 西 安 交 通 大 学 学 报 Vol.41 №2
2007年2月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Feb. 2007

基于命题逻辑的组件约束检测

陈宁，冯博琴
（西安交通大学计算机科学与技术系，710049，西安）

摘要：针对组件约束数量大、复杂度高的问题，提出了一种基于命题逻辑的组件约束检测算法.通过命题符号化的过程，即通过5个联结词将日常语言中的命题转化成数理逻辑中的形式命题.该算法首次使用受限真值表的概念来描述组件约束，清楚、准确地揭示了软件组织结构、动态行为以及组件之间的转换，通过包含、否定和插入等规则来合并真值表，以解决检测组件约束时存在的冗余与冲突问题，并获得期望行为的最小组件集合，从而保证了组件约束之间相互一致.与基于规则模式表方法相比，所提算法克服了人工检测的不确定性，改善了检测的时间性能，并可将冲突错判率降低大约10%，平均处理时间降低大约33%.

关键词：组件约束；命题逻辑；真值表；检测算法
中图分类号：TP311 **文献标识码**：A **文章编号**：0253-987X(2007)02-0172-04

Component Constraint Detection Based on Proposition Logic

Chen Ning, Feng Boqin
(Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: Aiming at the problem that there exists very complicated and a large amount of component constraints, an algorithm of component constraint detection based on proposition logic was proposed, in which the proposition in daily diction was transformed into the formal proposition of mathematical logic via the process of proposition symbolization, i.e. five connective words. The proposed method utilizes the concept of constrained truth table to describe the component constraints, and the software architecture, dynamic behavior and transformation among components are revealed transparently and exactly. The truth table is incorporated by inclusion, negation and insertion rules to solve the problem of redundancy and conflict existing in detecting the component constraints. Moreover, a minimal set of rules of the expected behavior is obtained to ensure the consistency of component constraints. Compared to the existing rule pattern table approach, the uncertainty in manual detection is overcome and the temporal performance of detection is improved. The misjudgment of conflict is decreased by 10% and the average processing time is lowered by about 33%.

Keywords: component constraint; proposition logic; truth table; detection algorithm

在基于组件的软件开发过程中，组件之间的复杂关系形成了组件约束.不同的组件分析人员在发现、描述以及修改组件约束时会引入冗余与冲突等异常问题，这些问题将增加软件系统开发和维护的成本，甚至使得软件系统产生不期望的行为[1-2].少量的简单约束的手工检测易于实现，但随着约束数量以及复杂度的增加，它已显得无能为力[3-6].

本文提出了基于命题逻辑的自动检测算法，以检验组件约束的冗余与冲突.其中，用冗余检测来保证约束的独特性，实现经济约束集，即软件可以得到

收稿日期：2006-07-05. 作者简介：陈宁(1970～)，男，博士生；冯博琴(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2003AA1Z2610)；西安交通大学、IBM联合研究项目.

期望行为的最小集合的组件,而用冲突检测来保证组件约束在使用时的一致性.

1　命题演算

定义 1　符号化的命题称为命题公式,其形式化定义如下.

(1)命题变元以及 T、F 是命题公式.

(2)如果 a 是命题公式,则 $\neg a$ 也是命题公式.

(3)如果 a、b 是命题公式,则$(a \wedge b)$、$(a \vee b)$、$(a \rightarrow b)$、$(a \leftrightarrow b)$都是命题公式.

(4)有限次使用形式化定义(1)～定义(3)构成的符号串为命题公式.

在定义 1 中,以真、假为变域的变元称为命题变元,如果一个命题公式中含有 n 个不同的命题变元,则称其为 n 元命题公式.

定义 2　将真值表定义为 n 元命题公式 a,对于所有不同指派 π 下的真、假值排列,其中的指派 π 是指在 n 元命题变元的真值确定后的取值序列.

定义 3　将组件约束形式化定义为一个命题公式 f,则有 $\mathrm{lhs}(f) \rightarrow \mathrm{rhs}(f)$,其中 $\mathrm{lhs}(f)$ 为公式 f 的左端,称为前提,表示为 $p_1 \wedge p_2 \wedge \cdots \wedge p_n$,$p_i$ 为一个原子命题,又称为条件命题,$|\mathrm{lhs}(f)|$ 为所有条件命题的个数,任意 p_i 与 p_j 之间的操作符号为 AND,表示为 $\wedge$;$\mathrm{rhs}(f)=q$ 为公式 f 的结论,其中 q 为一个原子命题,又称为结论命题.

命题演算[7]是一种形式语言,其目的在于将数学中的逻辑论证符号化. 命题符号化的过程是通过 5 个联结词将日常语言中的命题转化成数理逻辑中的形式命题.

每个约束的任意条件命题以及结论命题都赋给一个真值. 如果执行命题,真值为 1(真),否则真值为 0(假). 任意组件约束 f 如果满足前提触发,则一定会触发结论的执行. 也就是说,在组件约束中,条件命题都为 1,而结果命题为 0 的情况没有意义. 可见,组件约束并不要求所有真值是枚举的,因此可用受限的真值表表示组件约束前提与结论之间的逻辑关系. 受限的真值表排斥组件约束中不可能情况的真值,例如构建的组件约束 $p_1 \wedge p_2 \rightarrow q$ 的真值表如表 1 所示,显然只有第 1、第 8 行有实际意义,表示这条约束能否执行.

在真值表中,每个真值指派都包括了所有的原子命题公式以及复合命题公式,而在组件约束的实际应用中,只需要前提与结论的原子命题的真值.

表 1　$p_1 \wedge p_2 \rightarrow q$ 的真值表

p_1	p_2	q	$p_1 \wedge p_2$	$p_1 \wedge p_2 \rightarrow q$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

因此,根据组件约束的应用需要,受限真值表只需包含条件命题与结论命题的真值指派为 1 的情况.

定义 4　组件约束 x 的受限真值表的形式化表达式为 $\Gamma(x)=(p_1,\cdots,p_n,q)$,其中 $p_1,\cdots,p_n$ 为条件命题,q 为结论命题,而对应命题的真值指派 $\Gamma_\pi(x)=(p_1^0,\cdots,p_n^0,q^0)$,其中 $p_i^0=1, 1 \leqslant i \leqslant n, q^0=1$,指派个数 $|\Gamma_\pi(x)|=1$.

2　组件系统异常检测

检测组件约束冗余和冲突的目标是帮助软件系统产生一个最小无冲突的组件约束集合. 下面给出冗余和冲突的定义,并提出一系列检验步骤.

定义 5　冗余是指一条组件约束存在多个版本. 例如,组件约束 $p_i \wedge p_j \rightarrow q$ 与 $p_j \wedge p_i \rightarrow q$ 除了条件命题顺序有所不同,表达的内容是一致的.

定义 6　冲突是指不一致性,包括前提不一致性与结论不一致性,前者指组件约束的条件命题不一致,而结论命题一致,例如 $p_i \wedge p_j \rightarrow q$ 与 $p_i \wedge \neg p_j \rightarrow q$;后者指组件约束的条件命题一致,而结论命题不一致,例如 $p_i \wedge p_j \rightarrow q$ 与 $p_i \wedge p_j \rightarrow \neg q$.

基于命题逻辑的检验方法的核心思想是:先为组件约束生成受限真值表,再利用一套规则来合并不同组件约束的真值表,最后通过定理来识别存在冗余与冲突的组件约束.

2.1　构建组件约束的有限真值表

基于定义 1,为每条组件约束生成受限真值表,其中组件约束的前提划分为多个条件命题,结论划分为一个结论命题,而构建的受限真值表只包含真值都为 1 的一行.

2.2　合并真值表

$\forall x$, y 为 2 条约束,$\Gamma(x)=(p_1, \cdots, p_m, q)$,

$\Gamma(y)=(p'_1,\cdots,p'_n,q')$，其中 $p_1,\cdots,p_m$ 为 x 约束的条件命题，q 为结论命题，$p'_1,\cdots,p'_n$ 和 q' 分别为 y 约束的条件、结论命题，合并的真值表为 $\Gamma(x\cup y)$（初为空），则本文提出合并真值表的3条规则描述如下.

(1)包含规则. 如果 $p_i\rightarrow p'_i$，可将 p_i 添加到 $\Gamma(x\cup y)$作为条件命题，且约束 x、y 在此命题列的真值取值分别为1.

(2)否定规则. 如果 $p_i\leftrightarrow\neg p'_i$，可将 p_i 添加到 $\Gamma(x\cup y)$作为条件命题，且约束 x、y 在此命题列的真值取值分别为1、0.

(3)插入规则. 如果 $\neg p_i\rightarrow p'_i$，可将 p_i、p'_i 插入到 $\Gamma(x\cup y)$作为条件命题，且约束 x、y 在 p_i 命题列的真值取值分别为1、0，而在 p'_i 命题列的真值取值分别为0、1.

类似地，结论命题 q、q' 也遵循上述3条规则添加到 $\Gamma(x\cup y)$之中，由此生成的合并真值表最多包含组件约束 x、y 的所有条件、结论命题，命题公式个数不超过 $m+n$，并含有2行真值取值序列，每行表示1条组件约束，即真值指派序列为组件约束的个数.

同样，遵循上述规则，合并 n 条组件约束 $x_1,\cdots,x_n$，可以得到合并真值表的形式化表示，描述如下.

$\Gamma(x_1\cup\cdots\cup x_n)=(p_1^\Gamma,\cdots,p_k^\Gamma,q_1^\Gamma,\cdots,q_s^\Gamma)$，满足 $1\leqslant k\leqslant|\mathrm{lhs}(x_1)|+\cdots+|\mathrm{lhs}(x_n)|$，$1\leqslant s\leqslant n$，指派序列 $\Gamma_\pi(x_1\cup\cdots\cup x_n)=(p^0_{x1},\cdots,p^0_{xk},q^0_{x1},\cdots,q^0_{xs})$，$1\leqslant i\leqslant n$. 指派个数 $|\Gamma_\pi(x_1\cup\cdots\cup x_n)|=n$.

2.3 冗余与冲突的检测

$\forall x$，y 为2条约束，$\Gamma(x)=(p_1,\cdots,p_m,q)$，$\Gamma(y)=(p'_1,\cdots,p'_n,q')$，其中 $p_1,\cdots,p_m$ 为 x 约束的条件命题，q 为结论命题，$p'_1,\cdots,p'_n$ 和 q' 分别为 y 约束的条件与结论命题，合并的真值表 $\Gamma(x\cup y)=(p_1^\Gamma,\cdots,p_k^\Gamma,q_1^\Gamma,\cdots,q_s^\Gamma)$ 的指派序列 $\Gamma_\pi(x\cup y)$ 满足如下定理.

定理 如果 $p^0_{x1}\oplus p^0_{y1}+\cdots+p^0_{xk}\oplus p^0_{yk}=0$，条件异或之和为假，而 $q^0_{x1}\oplus q^0_{y1}+\cdots+q^0_{xs}\oplus q^0_{ys}=1$，结论异或为真，属于结论不一致性的冲突情况；如果 $p^0_{x1}\oplus p^0_{y1}+\cdots+p^0_{xk}\oplus p^0_{yk}=1$，条件异或有一个为真，而 $q^0_{x1}\oplus q^0_{y1}+\cdots+q^0_{xs}\oplus q^0_{ys}=0$，结论异或为假，则属于前提不一致性的冲突情况；如果 $p^0_{x1}\oplus p^0_{y1}+\cdots+p^0_{xk}\oplus p^0_{yk}=0$ 且 $q^0_{x1}\oplus q^0_{y1}+\cdots+q^0_{xs}\oplus q^0_{ys}=0$，标志每2行异或为假的情况，则属于组件约束冗余的情况.

证明 合并的真值表提供了一系列的组合，它可扩展为一系列组件约束，同时提供一种冲突与冗余的检测方式. 在合并的真值表中，真值指派行表示组件约束，列表示原子命题. 显然，在条件命题逻辑原子公式集合中，所有的列都有相同的真值，但却收集了不同行，这意味着结论冲突. 如果在结论命题逻辑原子公式集合中所有的列的真值均相同，但收集了不同行，则为前提冲突；如果不同行的所有列项一致，则为冗余的情况. 总之，异常的情况是指：在合并的真值表中真值有共同的列，以及在同一个列中占据不同的行.

冗余与冲突的检测算法如图1所示.

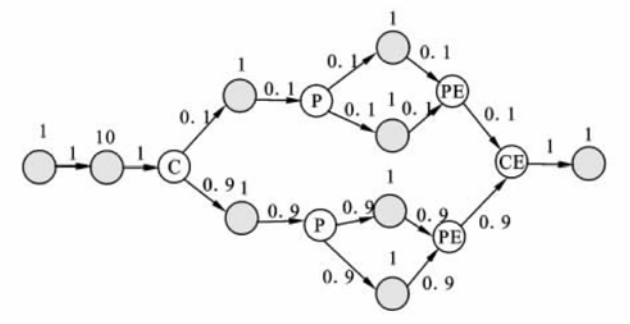
步骤1：构造组建约束$\forall x,\Gamma(x)=(p_1,\cdots,p_m,q)$的受限真值表；
步骤2：合并$\forall x,\Gamma(x)=(p_1,\cdots,p_m,q)$的真值表得到合并真值表
$\Gamma(x_1\cup\cdots\cup x_n)=(p_1^\Gamma,\cdots,p_k^\Gamma,q_1^\Gamma,\cdots,q_s^\Gamma)$，即
$\Gamma=\Gamma(x_1)$；//合并真值表的初始值
For ii=2 to n
$\Gamma=\Gamma$ merge$\Gamma(x_{ii})$；//按照包含、否定和插入规则合并真值表
End ii
步骤3：假设步骤2得到的合并真值表共w行，检测$\Gamma(x_1\cup\cdots\cup x_n)$的冗余与冲突，即
For ii=1 to $w-1$
For $jj=ii$ to $w-1$
If $p^0_{jj,1}\oplus p^0_{jj+1,1}+\cdots+p^0_{jj,k}\oplus p^0_{jj+1,k}=1$ and
$q^0_{jj,1}\oplus q^0_{jj+1,1}+\cdots+q^0_{jj,s}\oplus q^0_{jj+1,s}=1$
$q^0_{jj,1}\oplus q^0_{jj+1,1}+\cdots+q^0_{jj,s}\oplus q^0_{jj+1,s}=1$
无异常；
Elseif $p^0_{jj,1}\oplus p^0_{jj+1,1}+\cdots+p^0_{jj,k}\oplus p^0_{jj+1,k}=1$ and
$q^0_{jj,1}\oplus q^0_{jj+1,1}+\cdots+q^0_{jj,s}\oplus q^0_{jj+1,s}=0$
前提冲突；
Elseif $p^0_{jj,1}\oplus p^0_{jj+1,1}+\cdots+p^0_{jj,k}\oplus p^0_{jj+1,k}=0$ and
$q^0_{jj,1}\oplus q^0_{jj+1,1}+\cdots+q^0_{jj,s}\oplus q^0_{jj+1,s}=1$
结论冲突；
Elseif $p^0_{jj,1}\oplus p^0_{jj+1,1}+\cdots+p^0_{jj,k}\oplus p^0_{jj+1,k}=0$ and
$q^0_{jj,1}\oplus q^0_{jj+1,1}+\cdots+q^0_{jj,s}\oplus q^0_{jj+1,s}=0$
冗余；
Elseif
End jj
End ii

图1 冗余与冲突的检测算法

3 实验结果及分析

通过分析业务流程中的控制流信息[8]，将整个业务流程中的所有数据访问操作组成的数据访问序列称为数据访问流，用带权有向图表示. 数据访问节点的权值规定如下：并行、选择、序列活动的权值为1，重复活动的权值为重复的次数. 有向边的权值规定如下：序列、并行、重复控制结构中的边，权值为1，选择控制结构中的边，权值为该边可能被执行的

概率. 图 2 为一个数据访问流的例子.



C：选择活动开始节点；CE：选择活动结束节点；P：并行活动开始节点；PE：并行活动结束节点

图 2　数据访问流

数据访问流可以通过对业务流程进行如下处理得到：删除流程中所有的非数据访问节点；根据活动的控制结构，确定每个活动节点的执行次数，并作为该活动在数据访问流中的权值；指定数据访问节点的操作类型、操作要求.

因此，通过业务分析[9]，可以找到数据访问节点之间的约束关系，再将数据访问节点映射成数据访问组件.

选取 U-Rent 的用例[10]，可以找到 500 条组件约束，分别采用基于命题逻辑和基于规则模式表[11]的方法进行实验，通过运行测试比较了这 2 种检测方法的性能，结果如表 2 所示. 其中，错判率是检测出错的约束与所有检测到的约束之比，漏判率是未能检测出问题的约束与所有的约束之比.

表 2　2 种检测方法的性能比较

方法	冗余错判率/%	冗余漏判率/%	冲突错判率/%	冲突漏判率/%	平均处理时间/s
命题逻辑	0.11	0.41	0.83	1.23	0.89
规则模式表	0.13	0.48	11.33	1.67	1.32

通过实验结果可以得出，对于组件约束的冗余与冲突的检测情况，命题逻辑的方法能降低冲突错判率，而且极大地改善了检测处理的时间性能，平均处理时间要比规则模式表方法降低 33%.

4　结　论

最少无冲突的组件约束集可以客观、准确地描述软件系统，包括软件组织结构、动态行为以及组件之间的转换. 本文提出了一种基于命题逻辑的检验方法，并利用它进行组件约束的冲突与冗余检验. 该方法可以全面描述组件约束以及异常类型，改善检测的时间性能，降低异常检测的错判率与漏判率，这对于提高软件的可靠性和稳定性意义重大.

致谢　感谢 IBM 中国研究中心提供试验软件和大量相关技术文献和资料，感谢 IBM 中国研究中心组件部门张冠群先生提供的有益建议.

参考文献：

[1] Crnkovic I, Larsson M. Challenges of component-based development [J]. The Journal of Systems and Software, 2002, 61: 201-212.

[2] Mei Hong, Cheng Feng, Feng Yaodong, et al. ABC: an architecture based, component oriented approach to software development [J]. Journal of Software, 2003, 14(4): 721-732.

[3] Zaid A K, Levis A H. Validation and verification of decision making rules [J]. Automatica, 1997, 33(2): 155-169.

[4] Le M, Frydman G C, Torres L. Verification and validation of the SACHEM conceptual model [J]. International Journal of Human-Computer Studies, 2002, 56: 199-223.

[5] Murrell S, Plant R. On the validation and verification of production systems: a graph reduction approach [J]. International Journal of Human-Computer Studies, 1996, 44: 127-144.

[6] Sadiq W, Orlowska M E. Analyzing process models using graph reduction techniques [J]. Information systems, 2000, 25(2): 117-134.

[7] 王永庆. 人工智能原理与方法. [M]. 西安：西安交通大学出版社，2001.

[8] Mayer R J, Painter M K, Lingineni M. Toward a method for business constraint discovery (IDEF9) [M]. Washington DC: Knowledge Based Systems Inc., 1996.

[9] Object Connections Inc. Business rules management [EB/OL]. [2005-12-05]. http://www.objectconnections.com/documents/Whitepaper%20-%20Business%20Rules%20Management.pdf.

[10] Frias L, Queralt1 A, Olivé A. EU-rent car rentals specification [EB/OL]. [2006-01-08]. http://www.lsi.upc.es/dept/techreps/techreps.html.

[11] von Halle B. Business rules applied [M]. Hoboken, USA: Wiley Computer Publishing, 2002.

（编辑　苗凌）