全国第一届“网络化制造与大规模定制”学术会议，杭州,2003 年 10 月，pp.522-526

基于集合约束的工作流多实例数据访问冲突分析

王 远， 范玉顺

(清华大学自动化系,北京 100084)

摘 要 在工作流管理系统运行期间，可能会有同一工作流过程模型的多个实例在同时运行。这些实例可能会需要同时访问一个工作流数据，这样，这些实例之间会发生数据访问冲突，这会在工作流执行期间造成严重后果。本文提出了一种基于集合约束的工作流多实例数据访问冲突的分析方法，可以在模型定义阶段分析出所有可能发生的多实例数据访问冲突。本方法首先在一种结构化的工作流模型基础上导出集合约束，然后通过对导出的集合约束求解得到所有可能发生的多实例数据访问冲突。分析的结果不仅对工作流模型的合理定义，而且对工作流执行阶段数据访问冲突的解决方案设计都有重要的意义。

关键词 集合约束；工作流实例；数据访问冲突；工作流管理系统

0 引言

工作流是一种反映业务流程的计算机化的模型，是为了在计算机环境支持下实现经营过程集成与经营过程自动化而建立的可由工作流管理系统执行的业务模型[1]。工作流管理系统是一个软件系统，它完成工作流的定义和管理，并按照在计算机中预先定义好的工作流逻辑推进工作流实例的执行。在工作流实例的执行期间，如果多个在逻辑上可以同时执行的活动共享同一数据，则可能造成数据访问冲突。这种数据访问冲突可能来自一个工作流实例内部的并行活动之间，也可能来自不同工作流实例的活动之间。如果不对这些数据访问冲突进行分析和解决，会对工作流的执行造成严重的后果。而在模型定义阶段对数据访问冲突进行分析不仅对工作流模型的合理定义，而且对工作流执行阶段数据访问冲突的解决方案设计都有重要的意义。

在编程语言研究领域，对程序执行过程中数据的竞态条件分析已有大量的成果，如War-lock [2] 是ANSI C 的一个静态数据访问冲突分析系统，Eraser[3]是一个动态数据访问冲突和死锁的分析工具，Flanagan和Freund提出了一种Java语言中多线程数据访问冲突分析技术[4]。对工作流执行期间的数据访问冲突分析研究多是基于工作流的事务模型展开的；而Minkyu Lee在一种结构化的工作流模型上提出了一种基于集合约束的分析同一个工作流实例内部的并行活动之间数据访问冲突的方法[5]。

在上述研究成果的基础上，本文提出了一种基于集合约束的工作流多个实例间数据访问冲突的分析方法，该方法可以在模型定义阶段对工作流执行时可能发生的多实例间数据访问冲突进行分析。

1 实例间数据访问冲突描述

1.1 结构化工作流定义语言

本文在一种结构化的工作流定义语言(SWDL)[5]的基础上对数据访问冲突进行分析。SWDL的语

法如表1所示。显然SWDL省略了许多工作流模型信息，不是一个可用于实际执行的工作流模型定义语言，但是它可以描述工作流中的所有控制逻辑和数据，这对于数据访问冲突分析来说已经足够。SWDL的语义在表1中已有解释，如表达式**A ; (B (in a_0 , out b_0) || C (in a_1 , out b_1))** 表示这样一种执行逻辑： 活动**A**首先执行；然后活动**B**和活动**C**并发执行；活动**A**中没有输入输出数据，而活动**B**和活动**C**分别有输入数据**a_0**、**a_1**和输出数据**b_0**、**b_1**。需要说明的是在 **if then** 和 **while-do**控制结构中没有定义选择和循环的条件，这是因为对数据访问冲突分析来说，选择和循环的条件是无用的信息。

表1 SWDL的语法

w ::= **0**	(空活动)
| (w)	(优先级)
| **task** t (p_1,p_2,…,p_n)	(执行活动t)
| w_0 **;** w_1	(w_0 ,w_1顺序执行)
| w_0 || w_1	(w_0 ,w_1并行执行)
| **if then** w_0 **else** w_1	(选择分支)
| **while-do** w	(循环)
p ::= **in** x	(活动的输入参数)
| **out** x	(活动的输出参数)

1.2 实例间数据访问冲突

在工作流执行期间，可能会有多个实例在工作流管理系统中运行。这些实例可能来自同一个工作流模型，也可能来自不同的工作流模型。无论是否来自同一个工作流模型，分别在两个实例中执行的活动都可能共享同一个工作流数据。我们将因为两个实例中的活动共享同一个工作流数据而造成的数据访问冲突分为以下两类：

1、 读写冲突（**read-write conflict**）,这是由一个活动需要读出一个工作流数据而同时另一个活动需要对同一个工作流数据进行写入操作而产生的。

2、 写入冲突（**write-write conflict**）,这是由两个活动需要同时对同一个工作流数据进行写入操作而产生的。

2 实例间数据访问冲突分析

2.1 SWDL上的集合约束

我们对实例间数据访问冲突采用集合约束的分析方法。一个集合约束的形式化表达为：$X \supseteq Y$，其中 X 和 Y 均为集合表达式。一个集合约束系统是有限个集合约束的逻辑与连接：$\wedge_i X_i \supseteq Y_i$；而集合约束系统的解是一个从约束系统中的集合表达式到所研究的问题空间中集合的映射Ω，Ω满足：$\wedge_i \Omega(X)_i \supseteq \Omega(Y_i)$[3]。在实例间数据访问冲突分析中，设任意两个由SWDL定义的工作流表达式w_0与w_1，用集合变量 $X_{w0\text{-}w1}$ 来表示w_0与w_1在运行过程中的两个实例之间可能发生的数据访问冲突。而对两个工作流模型之间的任意一对表达式都定义这样的集合变量。举例说明，对于两个工作流模型 **A ; (B (in a_0 , out b_0) || C (in a_1 , out b_1))** 和 **E(in b_0 , out a_1) ; F (in b_1 , out (a_0 , b_1))** 对应的

集合变量有，$X_{A\text{-}E}$，$X_{A\text{-}F}$，$X_{A\text{-}EF}$，$X_{B\text{-}E}$，$X_{B\text{-}F}$，$X_{B\text{-}EF}$，$X_{C\text{-}E}$，$X_{C\text{-}F}$，$X_{C\text{-}EF}$，$X_{BC\text{-}E}$，$X_{BC\text{-}F}$，$X_{BC\text{-}EF}$，$X_{ABC\text{-}E}$，$X_{ABC\text{-}F}$，$X_{ABC\text{-}EF}$。任意的集合变量 $X_{w0\text{-}w1}$ 都对应一个集合约束 $X_{w0\text{-}w1} \supseteq E$，其中$E$是一个集合表达式，此集合约束的含义是：$X_{w0\text{-}w1}$ 所代表的数据访问冲突包含由集合表达式E所代表的数据访问冲突。

运用集合约束的方法对实例间数据访问冲突进行分析分为三个步骤：

1、 从给定的对应多实例的工作流模型中生成一个集合约束系统。

2、 对第一步生成的集合约束系统进行求解，求出其最小解。

3、 从集合约束系统的解找出显式表达部分即为所求的实例间数据访问冲突。

2.2 集合约束系统的生成

表2给出了基于SWDL定义的两个工作流模型之间任意两个表达式对应的集合约束的生成规则。表2中一行表示一个集合约束的生成规则，左边一列代表生成规则的应用条件，右边一列代表具体的生成规则。其中，w和v分别代表不同的实例的定义表达式，X表示两个不同实例的表达式对应的集合变量；C表示两个不同实例的表达式生成的集合约束。

表2 集合约束生成规则

规则的应用条件	集合约束生成规则
	w – 0 → Φ
w_0-w_1 → C	**(w_0) – (w_1) → C**
w_0-v_0 →C_{00}, w_0-v_1 →C_{01} **w_1-v_0 →C_{10}, w_1-v_1 →C_{11}**	**w_0; w_1 – v_0;v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}, X \supseteq X_{w1-v0}, X \supseteq X_{w1-v1}\}$ $\bigcup C_{00} \bigcup C_{01} \bigcup C_{10} \bigcup C_{11}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01} **w_1-v_0 →C_{10}, w_1-v_1 →C_{11}**	**w_0|| w_1 – v_0||v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}, X \supseteq X_{w1-v0}, X \supseteq X_{w1-v1}\}$ $\bigcup C_{00} \bigcup C_{01} \bigcup C_{10} \bigcup C_{11}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01} **w_1-v_0 →C_{10}, w_1-v_1 →C_{11}**	**w_0; w_1 – v_0||v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}, X \supseteq X_{w1-v0}, X \supseteq X_{w1-v1}\}$ $\bigcup C_{00} \bigcup C_{01} \bigcup C_{10} \bigcup C_{11}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01} **w_1-v_0 →C_{10}, w_1-v_1 →C_{11}**	**If then w_0 else w_1 – v_0;v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}, X \supseteq X_{w1-v0}, X \supseteq X_{w1-v1}\}$ $\bigcup C_{00} \bigcup C_{01} \bigcup C_{10} \bigcup C_{11}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01} **w_1-v_0 →C_{10}, w_1-v_1 →C_{11}**	**If then w_0 else w_1 – v_0||v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}, X \supseteq X_{w1-v0}, X \supseteq X_{w1-v1}\}$ $\bigcup C_{00} \bigcup C_{01} \bigcup C_{10} \bigcup C_{11}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01}	**While do w_0 – v_0;v_1 →** $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}\} \bigcup C_{00} \bigcup C_{01}$

w_0-v_0 →C_{00}, w_0-v_1 →C_{01}	**While do** w_0 –v_0‖v_1 → $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}\} \cup C_{00} \cup C_{01}$
w_0-v_0 →C_{00}, w_0-v_1 →C_{01}	**While do** w_0 – **If then** v_0 **else** v_1 → $\{X \supseteq X_{w0-v0}, X \supseteq X_{w0-v1}\} \cup C_{00} \cup C_{01}$
	task w(**in** i_0,…, **in** i_n,**out** p_0,…,p_m) – **task** v(**in** j_0,…, **in** j_k,**out** q_0,…,q_h) → { $X \supseteq$ R(w,i_0),…, $X \supseteq$ R(w,i_n), $X \supseteq$ R(v,j_0),…, $X \supseteq$ R(v,j_k), $X \supseteq$ W(w,p_0),…, $X \supseteq$ W(w,p_m), $X \supseteq$ W(v,q_0),…, $X \supseteq$ W(v,q_h)}
	task w(**in** i_0,…, **in** i_n,**out** p_0,…,p_m) – **task** v →Φ

以在2.1中给出的两个工作流模型 **A ; (B (in a_0 , out b_0) || C (in a_1 , out b_1))** 和 **E(in b_0 , out a_1) ; F (in b_1 , out (a_0 , b_1))**为例，表3给出了对这两个工作流模型的执行实例运用生成规则所生成的集合约束系统。

表3 两个工作流模型的执行实例运用生成规则所生成的集合约束系统

集合变量	生成的集合约束
X_{A-E}, , X_{A-F} , X_{A-EF}, , X_{B-E}, , X_{B-F} , X_{B-EF}, , X_{C-E}, , X_{C-F} , X_{C-EF}, , X_{BC-E}, , X_{BC-F} , X_{BC-EF}, , X_{ABC-E}, , X_{ABC-F} , X_{ABC-EF},	$X_{ABC-EF} \supseteq X_{BC-EF}$, , $X_{ABC-F} \supseteq X_{BC-F}$, $X_{ABC-E} \supseteq X_{BC-E}$, $X_{BC-EF} \supseteq X_{BC-F}$, $X_{BC-EF} \supseteq X_{BC-E}$, $X_{BC-F} \supseteq X_{C-F}$, $X_{BC-F} \supseteq X_{B-F}$, $X_{BC-E} \supseteq X_{B-E}$, $X_{BC-E} \supseteq X_{C-E}$, $X_{C-EF} \supseteq X_{C-E}$, $X_{C-EF} \supseteq X_{C-F}$, $X_{B-EF} \supseteq X_{B-E}$, $X_{B-EF} \supseteq X_{B-F}$, { $X_{B-E} \supseteq$R(B,a_0) , $X_{B-E} \supseteq$R(E,b_0) , $X_{B-E} \supseteq$W(B,b_0) , $X_{B-E} \supseteq$W(E,a_1)} { $X_{C-E} \supseteq$R(C,a_1) , $X_{C-E} \supseteq$R(E,b_0) , $X_{C-E} \supseteq$W(C,b_1) , $X_{C-E} \supseteq$W(E,a_1)} { $X_{B-F} \supseteq$R(B,a_0) , $X_{B-F} \supseteq$R(F,b_1) , $X_{B-F} \supseteq$W(B,b_0) , $X_{B-F} \supseteq$W(F,a_0) , $X_{B-F} \supseteq$W(F,b_1)} { $X_{C-F} \supseteq$R(C,a_1) , $X_{C-F} \supseteq$R(F,b_1) , $X_{C-F} \supseteq$W(C,b_1) , $X_{C-F} \supseteq$W(F,a_0) , $X_{C-F} \supseteq$W(F,b_1)}

2.3 集合约束系统的求解

对任何一个集合约束系统的求解都是反复使用一些求解规则对初始的集合约束系统进行变换。我们在表4中给出了上述集合约束系统的求解规则。在表4中，T和T’表示不属于同一实例的两个活动，表的每一行代表一个求解规则，这些求解规则的使用方法为：如果现有的集合约束系统中含有表4左边一列的某一项，则将表4右边一列对应的一项添加到现有的集合约束系统中去，形成新的集合约束系统。对初始的集合约束系统反复使用这些求解规则，直到集合约束系统不再有新集合约束生成为止。然后，在最终得到的集合约束系统中去除冗余的约束，得到的最终解具有如下形式(其中**C**表示最终得到的集合约束系统)：

{ $X \supseteq$conflict WW(T,T’,a) , $X \supseteq$conflict RW(T,T’,a) | $X \supseteq$conflict WW(T,T’,a) , $X \supseteq$conflict RW(T,T’,a) $\in$ **C**}

表4 集合约束系统的求解规则

旧的集合约束	生成的新集合约束
$X \supseteq Y \vee Y \supseteq$R(T,a)	$X \supseteq$R(T,a)
$X \supseteq Y \vee Y \supseteq$W(T,a)	$X \supseteq$W(T,a)
$X \supseteq Y \vee Y \supseteq$W(T,a) $\vee Y \supseteq$W(T’,a)	$X \supseteq$conflict WW(T,T’,a)
$X \supseteq Y \vee Y \supseteq$R(T,a) $\vee Y \supseteq$W(T’,a)	$X \supseteq$conflict RW(T,T’,a)
$X \supseteq Y \vee Y \supseteq$conflict WW(T,T’,a)	$X \supseteq$conflict WW(T,T’,a)
$X \supseteq Y \vee Y \supseteq$conflict RW(T,T’,a)	$X \supseteq$conflict RW(T,T’,a)

对表3中生成的集合约束运用上述规则进行求解可得：

{ $X_{ABC\text{-}EF} \supseteq$conflict WW(C,F,$b_1$) , $X_{ABC\text{-}EF} \supseteq$conflict RW(B,E,$b_0$) , $X_{ABC\text{-}EF} \supseteq$conflict RW(B,F,$a_1$) , $X_{ABC\text{-}EF} \supseteq$conflict RW(C,E,$a_1$) , $X_{ABC\text{-}EF} \supseteq$conflict RW(C,F,$b_1$) ,}

2.4 算法复杂性分析

本算法的时间复杂性为$O(mn)^3$，m和n分别表示两个实例的模型定义中所包含的表达式的个数。在集合约束生成阶段，所能生成的最多的集合约束为mn，所以此阶段的时间复杂性为O(mn)；在利用求解规则求解的阶段，所能生成的最多新的集合约束为$(mn)^2$，而在每次生成一个新的集合约束时，需要对生成的约束是否为真正的新约束进行判断需要的计算时间复杂性为O(mn)。所以总的时间复杂性为：$O(mn)+ O(mn)^2\ O(mn)= O(mn)^3$。

3 结束语

本文提出了一种基于集合约束的工作流多实例数据访问冲突分析方法，可以在工作流模型定义阶段对将来执行期间可能发生的多实例间数据访问冲突进行分析。该方法首先从对应多实例的结构化工作流模型中生成一个集合约束系统，然后通过对此集合约束系统求解得到所有可能发生的实例间数据访问冲突。

参考文献

[1] 范玉顺. 工作流管理技术基础. 北京: 清华大学出版社,2001

[2] N. Sterling, A static data race analysis tool, in: USENIX Winter Technical Conference, 1993： 97~106.

[3] S. Savage, M. Burrows, G. Nelson, P. Sobalvarro, T. Anderson. Eraser: A dynamic data race detector for multi-threaded programs, ACM Trans. Comput. Systems 15 (4) (1997) :391~411.

[4] C. Flanagan, S. Freund, Type-based race detection for Java, in:Proceedings of ACM Conference on Programming Language Design and Implementation, June 2000.

[5] Minkyu Lee , Dongsoo Han. Set-based access conflict analysis of concurrent workflow definition. Information Processing Letters ,80 (2001): 189~194

[6] Alexander Aiken. Introduction to set constraint-based program analysis. Science of Computer Programming, 35 (1999) :79~111

[7] N. Heintze, Set based program analysis, Ph.D. Thesis, Carnegie Mellon University, 1992.

全国第一届“网络化制造与大规模定制”学术会议，杭州,2003 年 10 月，pp.522-526