文章编号:1001 - 9081(2007)03 - 0650 - 03

协同设计中的并行冲突检测算法

汪大勇,金炜东
(西南交通大学 电气工程学院,四川 成都 610031)
(dywang11@163.com)

摘 要:冲突是协同设计的本质,产生的原因是协同设计环境下不同设计群体考虑问题的角度、评价标准和专业知识有所不同。提出了对等式协同设计中操作序列的一致性模型,在操作序列一致性的基础上提出了基于几何级的冲突检测算法,并在此基础上实现了系统设计冲突检测原型系统。

关键词:协同设计;冲突检测;规则;一致性
中图分类号: TP311.15 **文献标识码**:A

Conflict detection algorithm in collaborative design

WANG Da-yong, JIN Wei-dong
(*School of Electrical Engineering, Southwest Jiaotong University, Chengdu Sichuan* 610031, *China*)

Abstract: Conflict is the essence of collaborative design. The causes of conflicts include different thinking angles, judging criteria and specialty knowledge of different designers in collaborative design environment. A consistency model of operation sequence and conflict detection algorithm based on geometry level were proposed. Accordingly, a prototype conflict detection system in collaborative design was implemented.

Key words: collaborative design; conflict detection; rule; consistency

0 引言

协同设计是指在计算机的支持下,各成员围绕一个设计项目,承担相应的部分设计任务,并行交互地进行设计工作,最终得到符合要求的设计结果的设计方法。由于协同设计的复杂性,冲突是协同设计中不可避免的问题。在分布式协同设计环境中,冲突可能来自概念设计、方案设计、详细设计和生产过程设计等各个阶段;事实上,设计的过程也就是冲突的不断发生和解决的过程[1]。协同设计过程中的冲突具有必发性、多样性、关系性等特点[2]。根据冲突的起因和现象,协同设计过程中的冲突可分为设计冲突、资源冲突和过程冲突三类。冲突检测的方法大致可以分为领域无关法和领域相关法,两者的区别在于是否利用了领域的知识来判断冲突是否发生,常用的方法有基于 Petri 网的冲突检测、基于真值的冲突检测方法和不可满足约束的检测方法等[4~7]。

本文主要研究对等式协同设计系统冲突管理中的并发操作冲突检测机制。首先对协同设计中关于冲突检测的相关概念进行定义,接着引入较弱的因果一致性模型解决了协同设计操作历史序列一致性的问题,并针对现有基于特征的冲突检测技术的不足,提出了协同设计中并发操作的冲突检测算法。它不仅有效地提高了冲突检测的准确率,而且可作为以冲突为线索对协同设计过程进行全面协调与管理的基础。

1 并发操作的冲突检测模型

1.1 相关定义

定义 1 实体。实体 E 可定义为 $E = (E_{id}, Parent, Part, T_C, T_A, U_{id}, V, A_E)$,其中 E_{id} 是实体在系统中惟一的标识符;当实体 E_{id} 为一复杂实体的一部分时,$Parent$ 是指向其父实体的指针;当实体 E_{id} 本身为一复杂实体时,$Part$ 包含了指向其所有下一级实体的指针; T_C 是实体 E_{id} 在该协同工作空间中的创建时间; T_A 是实体 E_{id} 在该协同工作空间中最近被修改的时间; U_{id} 是对实体 E_{id} 实施最后一次修改操作的用户标识符列表;V 是实体 E_{id} 最近的版本序号; A_E 是实体 E_{id} 属性列表。

由定义 1 可知,协同设计系统中的共享实体是层次定义的,最底层包含了一组简单实体,是所有操作作用的最小单位实体。该层次化定义是我们以实体为中心的并发操作控制、冲突检测和冲突消解机制的基础。

定义 2 操作。操作 M 定义为 $M = (M_{id}, U_{id}, E_{id}, Cat, Type, Item, PRI, Part, Attribute, T_s, T_e)$,其中 M_{id} 是用户 U_{id} 在系统中作用于实体 E_{id} 之上任一操作的惟一标识符;Cat 是该操作所属类别;$Type$ 是该操作所属子类别;$Item$ 是该操作的项目名称;PRI 是子类别 $Type$ 的优先级别;$Part$ 是作用于实体 E_{id} 之上的操作 M_{id} 所涉及的部件列表; $Attribute$ 是操作 M_{id} 所涉及的所有属性列表; T_s 是操作 M_{id} 的开始时刻; T_e 是操作 M_{id} 的结束时刻(由于系统全局时间的不确定性,我们采用相对时间来加以描述[8])。

定义 3 并发操作。并发操作集合 $CONCMS$ 包含了一组操作 $M_1, M_2, \cdots, M_L (L > 1)$ 满足:

1) $M_1 \cdot E_{id} = \cdots = M_L \cdot E_{id}$;

2) $M_1 \cdot U_{id} \neq \cdots \neq M_L \cdot U_{id}$;

3) $[M_1 \cdot T_s, M_1 \cdot T_e] \cap \cdots \cap [M_L \cdot T_s, M_L \cdot T_e] \neq \varnothing$。

定义 4 因果关系"→"。给定两个操作 M_1 和 M_2,它们分别产生于站点 i 和站点 j,那么 $M_1 \rightarrow M_2$ 当且仅当下面三个条件之一成立:

1) $i = j$,并且 M_1 在 M_2 之前产生;

2) $i \neq j$,并且在站点 j 上先执行 M_1 然后产生 M_2;

收稿日期:2006 - 0913 - ;**修订日期**:2006 - 11 - 15

作者简介:汪大勇(1977 -),男,四川仁寿人,博士研究生,主要研究方向:协同设计、系统仿真、网络安全; 金炜东(1959 -),男,安徽人,教授,博士生导师,主要研究方向:智能信息处理、控制技术与控制理论、优化理论.

3) 存在一个操作 Mx,满足 $M_1 \rightarrow Mx$ 且 $Mx \rightarrow M_2$。

定义 5　历史操作队列。操作历史队列(History Buffer, HB) 用来保存站点已执行的操作序列。冲突检测算法的关键是在接收到操作并检测冲突操作的过程中形成一个能区分操作间并发与因果关系的序列 HB,以便加快冲突检测。

定义 6　操作粒度。操作粒度 $G(M)$ 包含了一组为顺利完成操作 M 而需要锁定的实体的最小部件或者简单实体的列表。

定义 7　设计约束。协同产品设计中的约束是一个包括设计规范、设计对象的基本规律、各种一致性要求、制造技术水平、资源环境限制、市场客户需求的所有信息的集合。约束表达式为:$C = \{X, R(X)\}$,其中,$X = [X_1, X_2, \cdots, X_n]$ 为变量集,X 的对应值域为 $D = [D_1, D_2, \cdots, D_n]$。各变量之间的关系一般满足表达式 $R(X) = R(X_1, X_2, \cdots, X_n)$。从定义可以看出,约束的实质是一种剔除规则,它将变量的取值范围限制在满足对应约束的区间之间,剔除违约变量值。

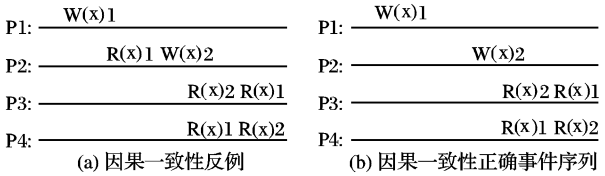
定义 8　操作冲突。操作 M_1 和 M_2 存在操作冲突意味着 M_1 和 M_2 之间满足:1) M_1 和 M_2 是并发操作;2) M_1 和 M_2 均为修改操作;3) $G_1(M_1) \cap G_2(M_2) \neq \varnothing$;4) $(M_1 \cdot Type \neq M_2 \cdot Type)$ 且 $(M_1 \cdot PRI = M_2 \cdot PRI = First)$;则记为 $Confliction(M_1, M_2)$。

1.2　操作历史队列一致性分析

在对等式分布式协同设计系统中,各个节点对某个对象的操作序列可能有多个版本,如何保证所有拷贝操作序列的一致性,这是检测各节点操作冲突的基础。由于操作对象分布于不同的节点,而操作节点间的通讯只能通过慢速(和本地写操作相比) 的网络发送信包来完成,维护绝对的操作一致性就显得尤其困难,甚至是不可能的[8]。在本文的模型中,降低了对操作一致性的要求,采用较弱的因果一致性模型来保证各协同节点操作对象的一致性。

因果一致性的协同节点应遵守的条件是:定义 4 所定义因果相关的写操作应对所有的协同节点可见,且顺序一致。并发操作在不同的节点看来顺序可能是不同的。

图 1 是一个因果一致性的示意图(其中:P1 ~ P4 代表不同机器的操作,每一节点完成的操作水平的显示时间轴向右增加,直线分割不同的节点操作)。图中,W(x)2 可能决定于 W(x)1,因为 2 可能是 R(x)1 所读的值计算的结果。两个写操作是因果联系的,所有进程必须视它们为同一顺序,因此图 1(a)不正确。图 1(b)中,W(x)1 和 W(x)2 变为并发事件。因果一致性模型不要求并发写有全局一致的次序,因此图 1(b)是正确的。



(a) 因果一致性反例　　(b) 因果一致性正确事件序列

图 1　因果一致性示意图

实现因果一致性需要由记录来跟踪哪个机器看到哪个写操作。这要建立和维护一个全局依赖图:即一个操作依赖于其他什么操作,这样做需要一些额外的开销。

1.3　冲突检测算法

通过上面的方法可以解决各站点操作历史队列的一致性问题,下面就在此基础上给出协同设计系统的冲突检测算法。从定义 6 中可以看出,不同站点发出的操作产生冲突的必要条件有以下三个:1)操作的并发性;2)操作对象的同一性;3)操作修改对象的同一属性,但其修改结果的不一致性。在介绍算法前,先介绍几个相关的概念。

$Attr(M)$:代表 M 作用的对象 E 的某个属性。

$Attr(M).type$:代表对象的属性类型,如位置、大小、线型等。

$Attr(M).Value$:代表对象的属性值。

GS:代表几何属性集合(如位置、大小等)。

$nonGS$:代表非几何属性集合(如颜色、线型等)。

$E.Mid$:表示对象的标识。

$M.tMid$:为操作的目标对象,则有 $E.Mid = M.tMid$。

冲突检测算法 $Conflict(M, HB)$ 如下:

1) 若操作 M 是创建操作类,则没有冲突;否则转 2)。

2) 令 $L = |HB|$,i 从 1 到 L 扫描 HB 中每一个操作是否与操作 M 具有并发关系。若存在并发关系,则转 3);否则操作 M 不会发生冲突。

3) 若存在一个操作 M_k 与 M 具有并发关系,得到候选对象集合 COS。若操作 M_k 的目标对象存在于 COS 中,即有 $M_k.tMid \in COS$,则转 4),否则没有冲突。

4) 若 M 是删除操作,则发生冲突,且为删除冲突;否则转 5)。

5) 采用约束网检测算法[10] 检查 M 是否违反操作对象 E 的相关约束条件,如果违反则发生违反约束冲突;否则转 6)。

6) 若 M 为修改操作:

a) 若有 $Attr(M) \neq Attr(M_k)$,则不存在冲突;

b) 如果 $Attr(M) = Attr(M_k)$,$Attr(M).type \neq Attr(M_k).type$,不存在冲突;

c) 如果 $Attr(M) = Attr(M_k)$,$Attr(M).type = Attr(M_k).type$,且 $Attr(M).value = Attr(M_k).value$,则为同一操作,可撤销 M;

d) 如果 $Attr(M) = Attr(M_k)$,$Attr(M).type = Attr(M_k).type$,且 $Attr(M).value \neq Attr(M_k).value$,则发生冲突;若 $Attr(M).type \in GS$ 则为几何冲突;若 $Attr(M).type \in nonGS$ 则为非几何冲突。

2　系统实现

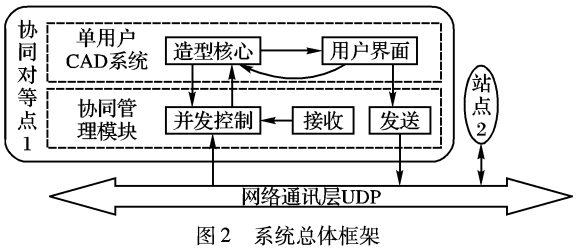


图 2　系统总体框架

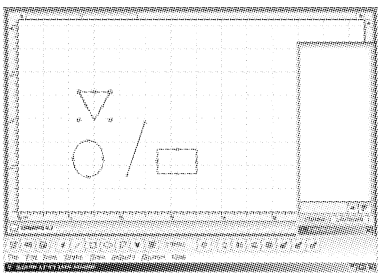


图 3　系统主界面

依据上述的分析,建立了对等式协同设计的冲突检测体系结构并建立了原型系统。其中对等式协同设计系统中每个协

作点都有相同的功能,主要模块包括多个协作对等点和网络通信层,各个站点上有独立的 CAD 系统,并通过协同管理模块进行调度,协同管理模块通过网络通信层将站点发出的操作信息和用户信息发送到其他站点,并同时接收来自这些站点的操作信息和用户信息。系统中没有规定的集中服务器,每个对等点都有自己的协同管理模块,系统将协作发起者作为主要的协同管理者,主要负责任务分配、冲突检测、冲突消解等任务。系统的总体框架和系统主界面如图 2,图 3 所示。

某个用户执行操作后,操作被发送到本地站点的造型模块立刻执行,同时该操作及与其相关的协同信息通过站点的网络发送模块发送到其他站点。站点通过网络接收模块接收其他站点发送的操作,并将这些操作送入并发控制模块进行判别,对于不存在并发冲突的操作则送入造型核心执行,对于发生冲突的操作则用消息通知用户界面,并作相应的调整。

3 应用对比分析

在局域网情况下,系统发生冲突的几率比较小,为了模拟 Internet 的操作环境,系统在每个操作后都随机加入了延迟,使得发生因果分离和并发操作的几率增加,以便于实验。

图 4(a)为某部件布局的简化模型,包括矩形模块 R1、R2 和圆形模块 C1 三个模块。R1 的中心距离坐标原点的距离为 r1,坐标为(x1,y1),质量为 m1,长宽为(L1,B1);R2 的中心距离坐标原点的距离为 r2,坐标为(x2,y2),质量为 m2,长宽为(L2,B2)。C1 的中心距离坐标原点的距离为 r3,坐标为(x3,y3),质量为 m3,直径为 L3。其中约束关系:r1 + r2≤50;质心约束,m1r1 = m3r3;L3 < min(L1,B1);x1 = x3,y1 = y3。

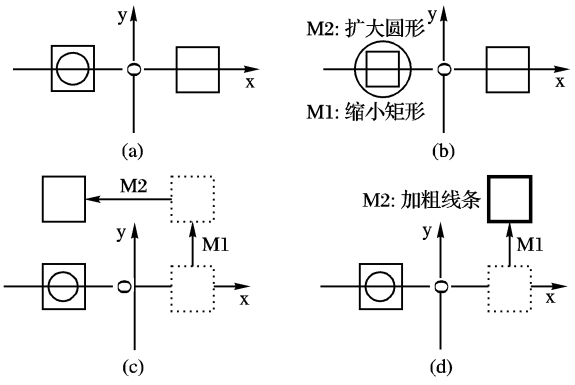


图 4 冲突检测应用示例

图 4(b)所示的两个并发操作 M1 为缩小 R1,M2 为扩大 C1,结果违反约束 L3 < min(L1,B1),发生冲突。图 4(c)所示的两个并发操作 M1 和 M2 同时改变对象 R2 的位置,它们的目标对象相同,且修改同一几何属性值,发生几何冲突。图 4(d)所示的两个并发操作 M1 为改变对象 R2 的位置,M2 为加粗 R2 边,尽管它们的目标对象相同,但修改的属性不同,不会发生冲突。

采用上述算法和基于特征级的冲突检测算法的仿真对比结果如表 1,其中模拟操作通过定义 $M = (M_{id}, U_{id}, E_{id}, Cat, Type, Item, PRI, Part, Attribute, T_s, T_e)$ 产生。仿真程序根据预先定义好的数目预先产生(冲突)操作。

表 1 冲突算法仿真对比分析

录入冲突次数	检测出冲突次数		查全率(%)		准确率(%)		反应时间/s	
	特征	几何	特征	几何	特征	几何	特征	几何
10	14	11	100	100	71.4	90.9	0.58	0.59
50	63	53	99.3	98.9	76.2	88.7	3.36	4.02
500	643	529	99.5	99.1	76.2	91.3	36.25	41.23

从表 1 可以看出,基于特征级的冲突检测和本文所提出的基于几何级的冲突检测在查全率、反应时间上都基本差不多,但从准确率上来说后者的方法更好。这样可以减少大量的冲突解决时间,提高工作效率,如果冲突解决需要人为介入时效果更明显。

当前,大多数协同设计系统的冲突检测都是基于特征级的冲突检测,这扩大了冲突检测的范围,本文提出的冲突检测方法在协同对等点操作序列一致性的基础上,引入约束检测机制,实现了基于几何级的冲突检测方法,它能够较好地解决绝大部分二,三维造型平台上进行协同设计的冲突检测问题,提高了冲突检测的效率。

参考文献:

[1] 杨友东,周勋,方志民. 复制式协同设计系统的并发控制研究[J]. 中国制造业信息化,2005,34(6):95 - 97.

[2] 陈明. 协同设计环境下的冲突检测模型设计[J]. 计算机时代,2005,(7):3 - 5.

[3] 余春燕,侯宏仑,吴明晖,等. 协同设计中以实体为中心的并发操作控制机制[J]. 计算机辅助设计与图形学学报,2004,16(9):1289 - 1295.

[4] TARATOUKHINE V. Conflict Management in Computer Supported Collaborative Design. The Methodology and Applications[A]. 2002 IEEE International Conference on Artificial Intelligence Systems[C]. 2002. 152 - 157.

[5] SCHWARTZ DR. Database Support for Conflict Detection in a Computer-Supported Cooperative Work Environment[A]. Proceedings of Database Engineering and Applications Symposium[C]. 1997. 240 - 249.

[6] EGYED A, GRüNBACHER P. Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability Can Help[J], IEEE Software, 2004,21(6):50 - 58.

[7] BOULILA N. Group Support for Distributed Collaborative Concurrent Software Modeling[A]. Proceedings of 19th International Conference on Automated Software Engineering[C]. 2004. 422 - 425.

[8] TANENBAUM AS. Distributed Operating Systems[M]. Publishing House of Electronics Industry, 1999. 50 - 250.

[9] 尚建忠,唐力,孟凡磊,等. 基于并发式检测时序的几何级冲突检测策略研究[J]. 国防科技大学学报,2004,26(5):90 - 108.

[10] 陈梦,王新平,宿英新. 并行设计中冲突的一致性检测算法[J]. 北京科技大学学报,2002,24(4):479 - 483.

(上接第 649 页)

[8] W3C. OWL Web Ontology Language Guide[EB/OL]. http://www.w3.org/TR/2004/REC-owl-guide-20040210/, 2004.

[9] W3C. Resource Description Framework (RDF): Concepts and Abstract Syntax [EB/OL]. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.

[10] BONINO D, CORNO F, FARINETTI L, *et al*. Multilingual Semantic Elaboration in the DOSE Platform[A]. 2004 ACM Symposium on Applied Computing[C]. 200. 1642 - 1646.

[11] ROUSSEY C, CALABRETTO S, PINON J - M. A Multilingual Information System based on Knowledge Representation[A]. Proceeding of the 5th East European Conference on Advances in Databases and Information Systems[C]. 2001. 98 - 111.