计算机科学2002Vol. 29№. 2

基于时态逻辑的虚拟企业活动规划

Temporal Logical Based Virtual Organization Activity Planning

袁成祥　高　济　林东豪
（浙江大学计算机系　杭州310027）

Abstract　As a method of collaboration between general organizations, it is of great importance for the virtual organization to plan it's activities. This paper first describes the virtual organization activities using temporal logical, then analyses the safety and liveness of the virtual organization activity planning. At last the planning arithmetic under safety and liveness constrains is come up.

Keywords　Temporal logical, Virtual organization, Activity planning

1　引言

常规的组织(或个人)为了提高竞争能力和快速响应市场变化的能力，需要合作组成虚拟企业。虚拟企业根据外界的情况决定自己的生产活动。虚拟企业活动规划对协调其活动很重要。

文[3]中，MTL(Metric Temporal Logic，一种时态逻辑)被用来对 agent 的活动进行规划。MTL 是以单位时间计数来指示时间的持续，如活动 a 在三个单位时间内完成可表示为(a,3)。在现实中，给出完成一个活动所需要的时间范围更合理。如活动 a 在二至三个单位时间内完成可表示为(a,2,3)。将活动所需要的时间从一个时间段扩展到一个范围是文[1]对 MTL 扩展的部分，记为 EMTL(Extern Metric Temporal Logic)。本文在 EMTL 的基础上对虚拟企业活动建模，并对其进行规划。

基于时态逻辑的虚拟企业活动模型和活动规划很好地表达了各个活动之间复杂的时序关系，能够规划复杂的活动。本文的规划方法还可以用在其它需要动态规划的地方，如机器人规划。另外，本文对安全性和活性的讨论为行为的可控制性提供了一个借鉴。

2　虚拟企业活动规划参考模型

虚拟企业活动在本文指企业的生产活动。虚拟企业的生产是根据市场的需求和自身的生产能力和资源来安排的，在市场不断变化的情况下，虚拟企业要能够灵活地改变生产计划，必须提前做好规划，以便在将来环境改变的时候采取最有利于它的行动。

3　虚拟企业活动过程模型

3.1　EMTL 时态逻辑

EMTL 公式由可数的命题集合：布尔算子 $\wedge$(与)、$\vee$(或)、$\neg$(非)、$\rightarrow$(蕴涵)；以及时态算子$\bigcirc_{\sim t}$(next)、$\Box_{\sim t}$(always)、$U_{\sim t}$(until)等构成。这里的时间连接符"$\sim$"$\in\{$">","<","$\geqslant$","$\leqslant$"$\}$，t 是非负整数，表示时间。

定义1　EMTL 公式(简称公式)递归定义如下：

·任何一个命题 p 是一 EMTL 公式；

·若 f_1与 f_2是 EMTL 公式，则$\neg f_1$，$f_1\rightarrow f_2$，$f_1\wedge f_2$，$f_1\vee f_2$，$\bigcirc_{\sim t} f_1$，$\Box_{\sim t} f_1$以及 $f_1 U_{\sim t} f_2$也是 EMTL 公式。当 $f_1=true$ 时，$f_1 U_{\sim t} f_2$可简写为$\Diamond_{\sim t} f_2$；

·EMTL 公式只能由上述两条规则产生。

EMTL 公式可由一个模型 M 来解释。

定义2　模型 M 定义为一个三元组(W,π,D)，其中：

·W 是一个有限的状态序列 $w_0 w_1\cdots$；

·π 是一个二元函数，若命题 p 在状态 w 中成立，则 $\pi(p,w)=true$；

·D 是一个时间函数，$D(w_i,w_{i+1})$表示从状态 w_i 到状态 w_{i+1}所需的时间，其返回值是个时间段$[d_l,d_k]$，d_l 表示所需的最短时间，d_k 表示所需的最长时间。

任何一 EMTL 公式都可以看作是一个安全性限制(safety constraint)公式或活性限制(liveness constraint)公式。安全性限制可以由形式为$\bigcirc_{\sim t} f_1$，$\Box_{\sim t} f_1$，$f_1 U_{\leqslant t} f_2$，以及 $f_1 U_{<t} f_2$的公式来表示。活性限制可以由形式为 $f_1 U_{\geqslant t} f_2$或 $f_1 U_{>t} f_2$的公式来表示。

3.2　虚拟企业的活动模型

在虚拟企业业务活动过程中，每个逻辑步骤或环节称之为活动。它包含以下信息：开始和结束条件，预处理和后处理，可完成此活动的成员企业，活动完成的时间限制条件，参考的信息模型等。我们在 EMTL 的基础上定义虚拟企业业务活动过程模型。

定义3　虚拟企业业务活动过程模型可以定义为一个六元组 EMTL_VOAP$=(w_0,W,VE,AE,succ,f)$，其中：

·w_0为系统的初始状态；

·$W=\{w_0,w_1,w_2,\cdots\}$为系统所有状态的集合。

·$VE=\{ve_0,ve_1,ve_2,\cdots,ve_n\}$是各个虚拟企业的集合。

·$AE=(A,E)$，其中 $A=\{a_1,a_2,\cdots,a_m\}$，是系统中所有活动的集合；E 是一个二元关系，$E\in VE\times A$，如$\langle ve_i,a_j\rangle$表示 ve_i 有承担活动 a_j 的能力，$i\in\{1,2,\cdots,n\}$，$j\in\{1,2,\cdots,m\}$。

·$succ$ 是一个状态转变函数，$succ(w)=((a_1,d_{1l},d_{1k},W_1),(a_2,d_{2l},d_{2k},W_2),\cdots,(a_k,d_{kl},d_{kk},W_k))$，其中 d_{il}与 d_{ik}分别表示执行活动 a_i 所需的最少时间和最长时间，W_i 表示在状态 w 下，执行活动 a_i 后，系统可能的状态集合。

·f 是一个 $EMTL$ 公式，表示系统所要满足的约束条件，即系统进行活动规划所要达到的目标。

定义4　进行活动规划时使用的每一条规划规则可以用

袁成祥　博士生，高　济　教授，博导，林东豪　博士生。

一个五元组(w,a,d_l,d_h,W)来表示，其中：

·w 代表当前状态；

·a 表示在状态 w 下系统能执行的一个活动；

·d_l、d_h 分别表示活动 a 执行所需的最短时间和最长时间；

·W 是一个状态集，代表执行活动 a 后，系统可能的状态集合。

定义5 如果存在状态序列 $w,w_1w_2\cdots w_nw,$，i 不属于{1，2，…，n}且在状态 w，时要满足的公式都是 f，则称此状态序列为一个环。

定义6 活动规划的结果是一满足公式 f 的规划，该规划可由一组状态控制规则(Situation Control Rule，简称SCR)来表示。每个SCR可用一个四元组(n,w,a,N)来表示，其中：

·n 是一个整数，用以表示规划状态，规划状态不是系统状态，它类似于程序设计语言中的跳转标记；

·w 表示当前系统状态；

·a 表示活动，即在状态 w 下要执行的活动；

·N 是前面所说的 n 的集合，即系统在状态 w 下执行活动 a 后下一个可能的规划状态的集合。

4 活动规划的实现

规划方式就是根据初始状态 w_0 及函数 *succ* 逐步展开而形成一规划图。对每一个状态 w，都有一目标公式 f 与其相对应，w_0 对应 f_0。在每一步规划展开时，都判断 f 是否已经满足，如果满足，得到一个完整规划；否则，随着系统状态的改变，系统的目标公式也要改变。系统可以在任何时候停止规划，这时系统的目标公式没有被全部满足，得到一个部分规划。系统目标公式的改变称为提升，包括两个方面的内容：安全和活性限制公式的提升。

4.1 活动规划与安全性限制

规划方式就是根据初始状态 w_0 及函数 *succ* 逐步展开而形成一规划图。下面给出安全性规划的一个简单的规划算法，称为算法一：

(1)给出初始状态 w_0 及相应的目标公式 f_0(f_0 不为 *false*)，那么对于由 $succ(w_0)$ 产生的每一条规划规则(a,d_l,d_h,W)，以及每一个状态 w' 属于 W，创建 w_0 的后继节点，其状态为 w'，目标公式为 f'。f' 是系统在 w' 状态下所要满足的目标公式，其得到的算法稍后介绍。

(2)对于(1)中产生的所有后继节点，如果其目标公式 f' 为 *true*，或者规划被终止，用回溯算法就可以从规划图中得到一个规划。否则根据(1)中方法继续展开。

在每步规划展开时，如果目标公式 f 不满足，需提升系统安全性限制目标公式。给定状态 w 及其相应目标公式 f，应用函数 *succ* 产生的规划规则后系统转为下一状态 w'，相应地，目标公式提升为 f'。提升算法如下，称为算法二：

```
输入：目标 f，状态 w，状态 w 转变到下一个状态 w' 所需的最少时间
      和最长时间 d_l 与 d_h，以及函数 π。
输出：提升后的 EMTL 公式 f'。
progress_goal(f,w,d_l,d_h,π){
  switch f
    case p: if π(p,w) then return true else return false endif;
    case f1: return ¬progress_goal(f1,w,d_l,d_h,π)
    case f1 ∧ f2: return progress_goal(f1,w,d_l,d_h,π) ∧ progress_
        goal(f2,w,d_l,d_h,π)
    case f1 ∨ f2: return progress_goal(f1,w,d_l,d_h,π) ∨ progress_
        goal(f2,w,d_l,d_h,π)
    case ◯≤t f1: if d_h≤t then return f1 else return false endif;
    case ◯≥t f1: if d_l≥t then return f1 else return false endif;
    case □≤t f1:
      if d_l>t then return progress_goal(f1,w,d_l,d_h,π)
      else return progress_goal(f1,w,d_l,d_h,π) ∧ □≤t-d_l f1 endif
    case □≥t f1:
      if d_h≤t then return □≥t-d_h f1
      else if t≠0 then return □≥0 f1
           else return progress_goal(f1,w,d_l,d_h,π) ∧ □≥0 f1 endif;
      endif
    case f1 U≤t f2:
      if d_h>t then return progress_goal(f2,w,d_l,d_h,π)
      else return progress_goal(f2,w,d_l,d_h,π) ∨
                  (progress_goal(f1,w,d_l,d_h,π) ∧ f1 U≤t-d_h f2) en-
                  dif;
    case f1 U≥t f2:
      if d_h≤t then return f1 U≥t-d_l f2
      else if t≠0 then return f1 U≥0 f2;
           else return progress_goal(f2,w,d_l,d_h,π) ∨
                       (progress_goal(f1,w,d_l,d_h,π) ∧ f1 U≥0 f2)
                       endif
      endif
}
```

一个安全性限制公式被提升后，或者返回 *false*，或者返回 *true*，或者减少目标公式的时间范围。当时间减少到小于0的时候，目标公式返回 *false*，所以随着活动规划的展开，目标公式都会不一样，有限步的规划内就可以判断目标公式能否得到满足。而对于一个活性限制公式当如 $f_1U_{\geq t}f_2$，当 $t=0$ 时，返回 $progress\_goal(f_2,w,d_l,d_h,\pi)\vee(progress\_goal(f_1,w,d_l,d_h,\pi)\wedge f_1U_{\geq 0}f_2)$，已经与时间无关。如果 f_1 为 true，f_2 始终为 false，则形成一个环。如图1所示：

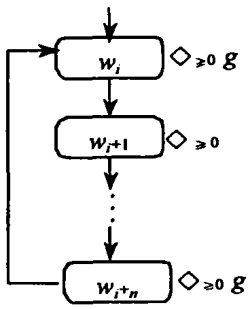


图1 不满足公式$\Diamond_{\geq 0}g$的环

例1 设一虚拟企业内有企业 A、企业 B、加工中心 C，企业 A，B 都有一种产品需要经过加工中心 C 加工，而加工中心一次只能为一个企业提供服务。加工中心应在企业提供申请后两个单位时间内为其安排加工服务。为了简单起见，本例中我们假设每个活动所需要的时间范围都是[1，1]。系统的状态、活动、规划规则，安全性限制公式等分别描述如下：

状态包括：

w_0：()；w_1：($request(A,C)$)；
w_2：($request(B,C)$)；w_3：($request(A,C)$，$request(B,C)$)；
w_4：($using(A,C)$)；w_5：($using(B,C)$)；
w_6：($using(A,C)$，$request(B,C)$)；w_7：($using(B,C)$，$request(A,C)$)；
w_0：初始状态，加工中心空闲。

注：$request(M,C)$表示加工中心有企业 M 提出的申请，$using$(N，C)表示加工中心正在为企业N提供服务。

安全性公式：

$f_0=\Box_{\geq 0}((request(A,C)\rightarrow\Diamond_{\leq 2}using(A,C))\wedge(request(B,C)\rightarrow\Diamond_{\leq 2}using(B,C))$

$f_1=f_0\wedge\Diamond_{\leq 1}using(A,C)\wedge\Diamond_{\leq 1}using(B,C)$

$f_2=f_0\wedge\Diamond_{\leq 0}using(B,C)$ $f_3=f_0\wedge\Diamond_{\leq 1}using(A,C)$ $f_4=f_0\wedge\Diamond_{\leq 1}using(B,C)$

活动包括：

$wait()$；$prepare(A)$；$prepare(B)$；

$prepare$(M)表示加工中心准备为企业M提供服务 $wait()$：等待

活动规划规则(用(w,a,d_l,d_h,W)五元组来表示，见定义4)：

$(w_0,wait(),1,1,(w_0,w_1,w_2,w_3))$

$(w_1,wait(),1,1,(w_1,w_3)),(w_1,prepare(A),1,1,(w_4,w_6))$

$(w_2,wait(),1,1,(w_2,w_3)),(w_2,prepare(B),1,1,(w_5,w_7))$

$(w_3,wait(),1,1,(w_3)),(w_3,prepare(A),1,1,(w_6)),(w_3,prepare(B),1,1,(w_7))$

$(w_4,wait(),1,1,(w_0,w_1,w_2,w_3))$

$(w_5, wait(), 1, 1, (w_0, w_1, w_2, w_3))$
$(w_6, wait(), 1, 1, (w_2, w_3)), (w_6, prepare(B), 1, 1, (w_5, w_7))$
$(w_7, wait(), 1, 1, (w_1, w_3)), (w_7, prepare(A), 1, 1, (w_4, w_6))$

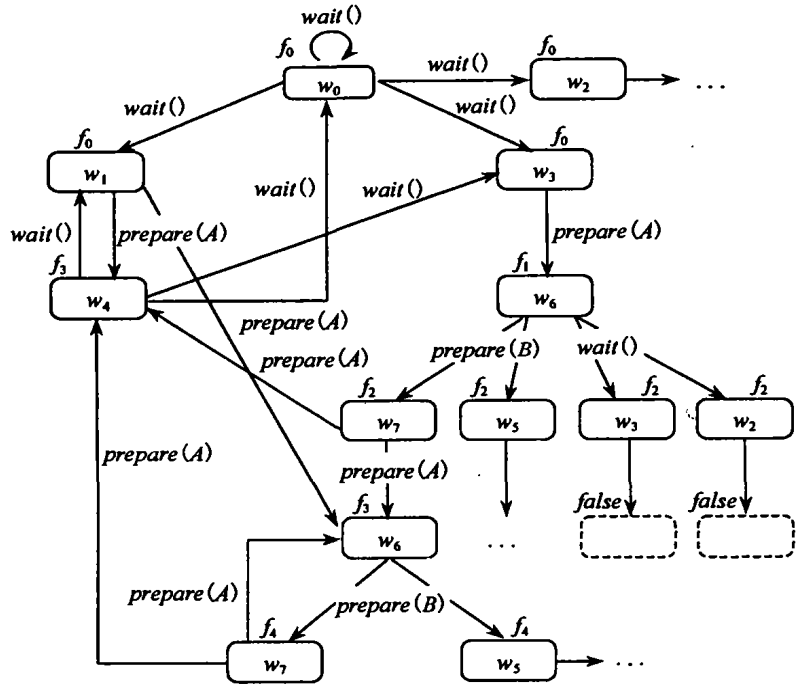


图2 一个部分规划图

展开后的部分规划图如图2所示。从该图中部分规划结果用SCR(见定义6)表示为：

(0 w_0 *wait*()(0,1,2)) (1 w_1 *prepare*(A)(3,5))
(2 w_3 *prepare*(A)(5))(3 w_4 *wait*()(0,1,2))
(4 w_5 *wait*()(0,1,2)) (5 w_6 *prepare*(B)(4,6))
(6 w_7 *prepare*(A)(3,5))

上面的例子中没有活性限制公式。如果有活性限制公式，算法二有可能得不到一个规划结果。下面讨论活性限制公式采用的规划算法。

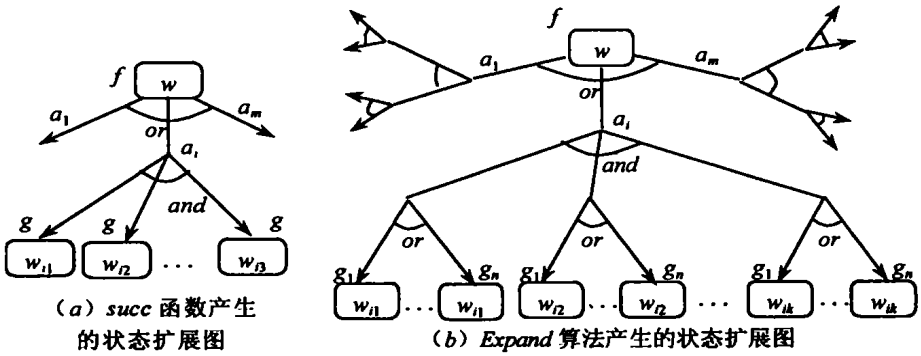
4.2 活动规划与活性限制

给定一个状态 w 及其应满足的公式 f，根据安全性规划的规划算法，可以形成一个规划图。在规划图中，每个结点记为 *node*，其相应的状态记为 *node*. w，应满足的目标公式记为 *node*. f。另外，在此引入公式集 *eventualities*，该公式集中的公式是状态 w 所要满足的活性限制公式，形如 $f_1 U_{\geqslant 0} f_2$ 或 $f_1 U_{>0} f_2$，记为 *node*. *eventualities*。

活性限制条件的规划过程中，系统的状态结点扩展算法如下：

```
Expand(node, succ, π){
  对于 succ(w)中的每一个(a, d_l, d_h, W){
    g = progress_goal(node. f, node. w, d_l, d_h, π);
    将 g 转化为一般否定形式 g';
    将 g' 转化为一般否定形式 g'' = g_1 ∨ g_2 ∨ … ∨ g_n;
    for(i = 1, i≤n, i++) //有 n 个析取因式{
      对于W 中的每个状态 w'{
        node = create_new_node(); node. goal = g_i;
        node'. eventualities = progress_eventualities ( node,
            node', d_l, d_h, π)
      }}}}
```

图3示例说明了状态结点的扩展。其中(*a*)子图是由 *succ* 函数产生的状态结点扩展图，(*b*)子图是由 *Expand* 算法产生的状态结点扩展图。图3(*a*)中的 w 是 w_0 分裂为 n 个初始结点中的一个。



(*a*) *succ* 函数产生的状态扩展图　　(*b*) *Expand* 算法产生的状态扩展图

图3 状态扩展示例图

系统初始状态为 w_0，目标公式为 f，原始结点记为 $node_0$。先将公式 f 转化为一般否定形式 f'，再将 f' 转化为一般析取式 $f'' = f_1 \vee f_2 \vee \cdots \vee f_n$，这样，$node_0$ 分裂为 n 个初始结点，每个初始结点的目标公式分别为 f_1、f_2、…、f_n，它们的 *eventualities* 公式集都为空。

eventualities 集里的公式都形如 $f_1 U_{\geqslant 0} f_2$ 或 $f_1 U_{>0} f_2$，且是状态结点目标公式的一个析取因式。公式集 *eventualities* 的提升算法 *progress_eventualities* 如下：

```
progress_eventualities(node, node', d_l, d_h, π){
    node'. eventualities = ∅ //初始时赋空值
    if node. eventualities = ∅ then
```

for *node.goal* 的每一个合取因式 f_i
if(f_i 形如 $gU_{\geq 0}h$ 或 $gU_{>0}h$)且
公式 f_i 不蕴涵于结点 *node* 的目标公式中
then insert f_i *into node'.eventualities endif*
else
for *node.eventualities* 中的每一个公式 f // f 形如 $gU_{\geq 0}h$ 或 $gU_{>0}h$
if 公式 f 不蕴涵于结点 *node* 的目标公式中
then copy f *from node.eventualities to node'.eventualities*
endif
endif
}

4.3 活动规划算法

考虑安全性限制条件和活性限制条件，给定一个初始状态 w_0，函数 *succ*，活性限制条件 f，以及命题判断函数 π，系统的活动规划算法如下：

planner(w_0, f, *succ*, π){
将 f 转化为一般否定式 f'；
将 f' 转化为一般析取否定式 f''；
将 f''的所有析取因式放入目标集 G 中；
对于集合 G 中的每一个公式 g{
$node_0$=*create_new_node*(); //创建一初始结点
$node_0.state=w_0$; $node_0.goal=g$;
$node_0.eventualities=\phi$;
search($node_0$, *succ*, π); //在 *expand* 算法生成的状态扩展图中搜索
if 找到一满足目标公式的规划 then *exit endif*;
}}

其中，*search* 函数是在由 *expand* 算法生成的状态扩展规划图中搜索一有限子图，该子图要满足下列条件：

·不存在目标公式为 *false* 的状态；

·该子图中任意一个环中，最少包含一个结点，其 *eventualities* 公式集为空；

·在该子图中，每个结点都有且仅有一个含有执行某一活动后产生的后继结点集；

·在该后继结点集中，含有相同状态的结点中有一个在该子图中有后继结点。

这样的子图我们称之为实现图。

例2 在例1考虑了活性后的一个部分规划图。图3是一个部分节点扩展图。

$f=\Box_{\geq 0}((request(A,C)\rightarrow\Diamond_{<2}using(A,C))\wedge(request(B,C)\rightarrow\Diamond_{<2}using(B,C))$
f_0为 f 的一般析取否定式
$f_1=f_0\wedge\Diamond_{\geq 0}using(A,C)\wedge\Diamond_{\geq 0}using(A,C)$
$f_2=f_0\wedge\Diamond_{\geq 0}using(B,C)$
$f_3=f_0\wedge\Diamond_{\geq 0}using(A,C)$

图中的状态和例1中的状态相同。

eventualities 公式集
$E_0=\phi$（ϕ 表示为空） $E_1=\{\Diamond_{\geq 0}using(B,C)\}$ $E_2=\{\Diamond_{\geq 0}using(A,C)\}$

从此图中可以看出，规划 $w_0\rightarrow w_3\rightarrow w_7\rightarrow w_3\rightarrow w_6\rightarrow w_3\rightarrow w_7\rightarrow w_3$虽然规划中出现了环，但是因为存在 *eventualities* 公式集为空的节点，所以这个规划是可以实现的。$w_0\rightarrow w_3\rightarrow w_7\rightarrow w_3\rightarrow w_7$中出现了环，且不存在 *eventualities* 公式集为空的节点，所以这个规划是不可以实现的。

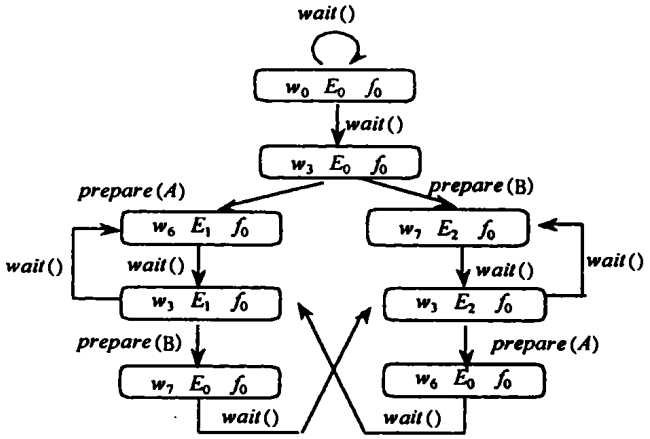


图4 一个部分节点扩展图

总结 本文首先介绍了 ETML 时态逻辑和虚拟企业的活动模型的抽象描述。然后分别考虑虚拟企业活动规划的安全性和活性的活动规划算法，最后给出了完整的活动规划算法。由于有安全性和活性限制，使得系统的规划是有效的和合理的。

参考文献

1 林东豪. 基于 agent 技术的虚拟企业信息基础集成框架研究:[浙江大学博士论文]
2 高济，林东豪. 基于 agent 技术的虚拟组织集成框架 IFVO. 计算机研究与发展，1999，36(12)：1409～1416
3 Jennings N R, et al. Using Archon to Develop Real-World DAI Applications. IEEE Expert Intelligent Systems & their Applications，1996，11(6)
4 Jin Y, et al. The Virtual Design Team: Modeling organizational behavior of concurrent design teams. Artificial Intelligence for Engineering Design. 1995，9：145～158
5 Rao, George. A model-theoretic approach to the verification of situated reasoning systems. In :Proc. of Thirteenth Intl. Joint Conf. on Artificial Intelligence (IJCAI-93), France，1993，318～324
6 Kraus S, et al. Multiagent Reasoning with Probability, Time, and Beliefs. Intl. journal of intelligent systems，1995，10：459～499
7 马华东，刘慎权. 基于时序逻辑的动画描述模型. 计算机学报，1995，18(11)：814～821