面向对象的命题逻辑和分布式推理

熊 毅 左志宏 周明天
(电子科技大学计算机学院 成都610054)

摘 要 命题逻辑是人工智能和知识工程的基础之一,然而传统命题逻辑却没有体现面向对象的思想,并且难于适应目前的分布式计算环境。本文用对象封装命题变元,实现了传统命题逻辑和面向对象思想的结合;本文引入命题关联类/对象来封装产生式,不但显式地表征了事物之间的命题逻辑关系,还使系统形成一种网络拓扑结构,实现了知识的分布式存储;本文还提出了真值传递和规则触发机制,通过它们可以将推理分布在网络的各个节点完成,并且实现了知识的自索引:相关的知识都存于本地,因此搜索只发生在一个节点内部,避免了全局搜索。

关键词 命题类/对象,命题关联类/对象,分布式推理,真值传递,规则触发

Object Oriented Proposition Logic and Distributed Deduction

XIONG Yi ZUO Zhi-Hong ZHOU Ming-Tian
(Computer School, University of Electronic Science and Technology of China, Chengdu 610054)

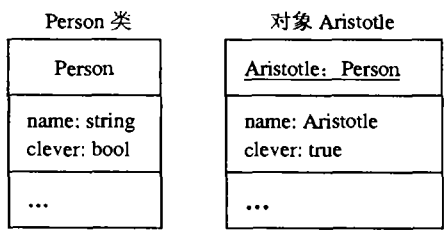
Abstract Proposition Logic is one of the foundations of Artificial Intelligence and Knowledge Engineering. However, traditional Proposition Logic is not designed in conformity to OO ideology, and has become inadaptable to modern distributed computing environment. The paper integrates traditional PL and OO methodology by encapsulating statement variables in objects so that OO technology is applicable to traditional Proposition Logic; and the paper introduces association classes/objects so as to encapsulate statement formulas, by which, the relations among entities are explicitly depicted and the distributed storage of knowledge is hence capable because of the network topology formed. Besides, the paper brings up a true value diffusion process and rule triggering mechanism to enable the implementation of deduction among distributed nodes in the network, thus, enables a knowledge self-indexing: since relevant knowledge has been stored locally, global search is unnecessary but a local one in stead.

Keywords Statement class/object, Statement association class/object, Distributed deduction, True value diffusion, Rule triggering

1 命题类/命题对象

在面向对象的命题逻辑(下简称 OOPL,Object Oriented Proposition Logic)中,命题变元被封装为对象的一个布尔型成员变量,该布尔变量的值对应了命题变元的真值。

例如,设"Aristotle 是聪明的"这一断言为命题变元 P,定义相应的类和对象:



则命题变元 P 在 OOPL 中被表示为 *Aristotle.clever*。

为方便以后的讨论,我们将拥有命题变元的对象称为命题对象,相应的类称为命题类。

2 命题关联对象/命题关联类

在 OOPL 中命题公式被封装在命题关联类或命题关联对象中。所谓命题关联类是这样的类:它的命题变元成员都来自于其它命题类(镜像关系),并且类中定义了这些成员之间必须满足的命题逻辑关系。因为这样的类实际上表征了参与关联的端类之间的命题逻辑关系,因此称其为命题关联类。自然地,该类的实例就称为命题关联对象,命题关联对象连接了多个命题对象,并反映了它们之间的命题逻辑关系。类定义的成员之间必须满足的关系称为类约束。

例:假设三个分别叫 Aristotle、Bakon、Comte 的学生,他们其中有且仅有一个聪明人,用 OOPL 表示为(UML 描述):

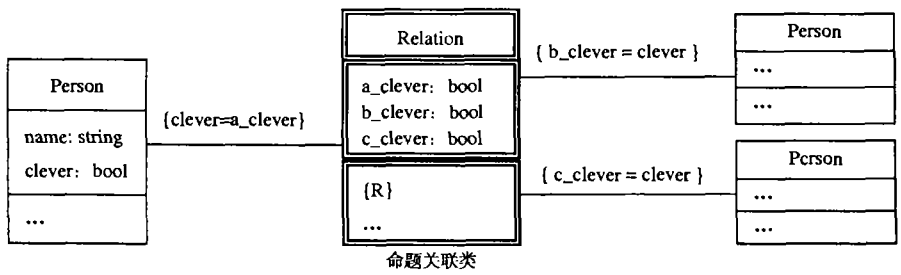


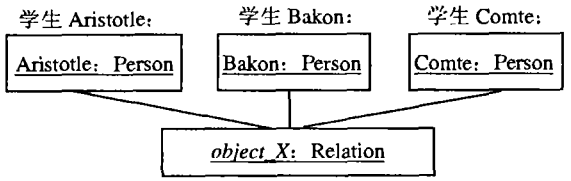
图1

熊 毅 硕士研究生,研究方向为语义互联网、ontology 应用;左志宏 副教授,研究方向为计算机智能、语义互联网;周明天 教授,博导,研究方向为计算机网络。

类 Relation 中：$R=(a\_clever \wedge \neg b\_clever \wedge \neg c\_clever) \vee (b\_clever \wedge \neg a\_clever \wedge \neg c\_clever) \vee (c\_clever \wedge \neg a\_clever \wedge \neg b\_clever)$；

在命题关联类 Relation 中，成员变量 a_clever、b_clever、c_clever 与另三个 Person 类的成员 clever 分别构成镜像关系；Relation 中定义了内部逻辑关系 R，凡是该类的对象都必须满足此命题公式。通过镜像变量和封装命题公式，类 Relation 定义了 Person 类三个对象之间的命题逻辑关系。

具体的命题关联对象为：



命题关联对象 "*object_X*"

图2

类完整性约束：系统通过所谓类完整性约束保证所封装知识的正确性：一旦 Aristotle、Bakon、Comte 之间的命题逻辑关系不满足类定义 R，系统就会抛出异常。

3 用命题关联类封装产生式

虽然命题关联类可以封装任何合法的命题逻辑公式，但为方便起见我们只考虑封装产生式，进而引出基于产生式的 OOPL 推理。我们将封装产生式的命题关联类暂时称为产生式关联类或规则关联类。

规则触发：Decision 的类中还将定义一些触发函数，一旦类约束中的产生式满足前件为真时，就将触发其后件为真；或当产生式满足后件为假时触发前件为假。

产生式的形式化约束：OOPL 的产生式限于以下两种形式：

形式 A：$P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow R$，即前件为一个合取式，后件为单一变元；

形式 B：$S \rightarrow Q_1 \vee Q_2 \vee \cdots \vee Q_n$，即前件为单一变元，后件为一个析取式；

形式 A 用于产生式的正向推理，当 P_1、P_2、…、P_n 全为真是可以触发 R 为真；形式 B 用于产生式逆向推理，当 $Q_1 \vee Q_2 \vee \cdots \vee Q_n$ 全为假时可以触发 S 为假。

4 基于产生式的 OOPL 推理

下面以"聪明人考试问题"为例说明基于产生式的 OOPL 推理。"聪明人考试问题"是这样的：

A(Aristotle)、B(Bakon)、C(Comte)三个学生一起参加了物理和化学两门考试。

三个人中，有且只有一个聪明人；聪明人是三人中唯一的一个通过这两门科目中某门考试的人；聪明人也是三人中唯一的一个没有通过另一门考试的人。

A 说：(1)如果我不聪明，我将不能通过物理考试；(2)如果我聪明，我将能通过化学考试。

B 说：(3)如果我不聪明，我将不能通过化学考试；(4)如果我聪明，我将能通过物理考试。

C 说：(5)如果我不聪明，我将不能通过物理考试；(6)如果我聪明，我将能通过物理考试。

考试结束后，证明三个人说的都是真话。

4.1 问题描述

为了进行基于产生式的 OOPL 推理，必须先建立一个对象/类网络，用 UML 描述如下：

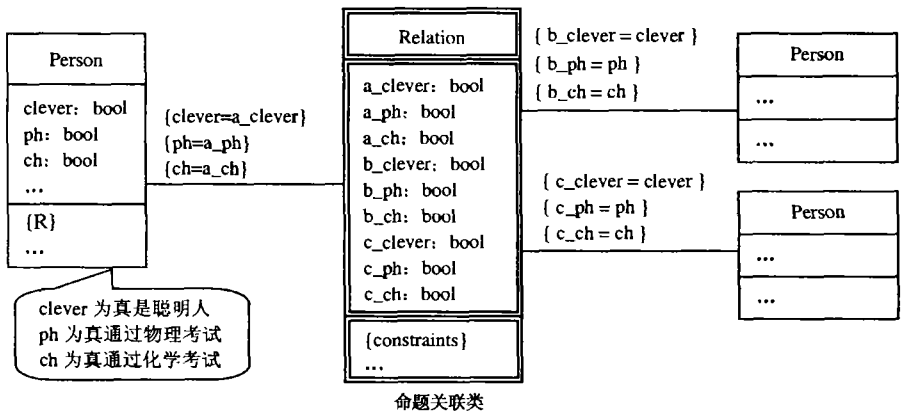


图3

Relation 类中的类约束{constraints}为：

//三人中有且只有一个是聪明人

r0：$\neg b\_clever \wedge \neg c\_clever \rightarrow a\_clever$

r1：$a\_clever \rightarrow \neg b\_clever \wedge \neg c\_clever$

r2：$b\_clever \rightarrow \neg a\_clever \wedge \neg c\_clever$

r3：$c\_clever \rightarrow \neg a\_clever \wedge \neg b\_clever$

//聪明人是三人中唯一的一个通过这两门科目中某门考试的人，聪明人也是三人中唯一的一个没有通过另一门考试的人

r4：$a\_clever \wedge a\_ph \rightarrow (\neg b\_ph \wedge \neg c\_ph) \wedge (\neg a\_ch \wedge b\_ch \wedge c\_ch)$

r5：$a\_clever \wedge a\_ch \rightarrow (\neg b\_ch \wedge \neg c\_ch) \wedge (\neg a\_ph \wedge b\_ph \wedge c\_ph)$

r6：$b\_clever \wedge b\_ph \rightarrow (\neg a\_ph \wedge \neg c\_ph) \wedge (\neg b\_ch \wedge a\_ch \wedge c\_ch)$

r7：$b\_clever \wedge b\_ch \rightarrow (\neg a\_ch \wedge \neg c\_ch) \wedge (\neg b\_ph \wedge a\_ph \wedge c\_ph)$

r8：$c\_clever \wedge c\_ph \rightarrow (\neg a\_ph \wedge \neg b\_ph) \wedge (\neg c\_ch \wedge a\_ch \wedge b\_ch)$

r9：$c\_clever \wedge c\_ch \rightarrow (\neg a\_ch \wedge \neg b\_ch) \wedge (\neg c\_ph \wedge a\_ph \wedge b\_ph)$

Person 类含有类约束{R}，此例中设三个实例分别为 Aristotle、Bakon 和 Comte，他们的约束分别为：

r10：Aristotle. $\neg$clever $\rightarrow$ Aristotle. $\neg$ph

r11:Aristotle. clever→Aristotle. ch
r12:Bakon. →clever→Bakon. →ch
r13:Bakon. clever→Bakon. ph
r14:Comte. →clever→Comte. →ph
r15:Comte. clever→Comte. ph
对象网络为：

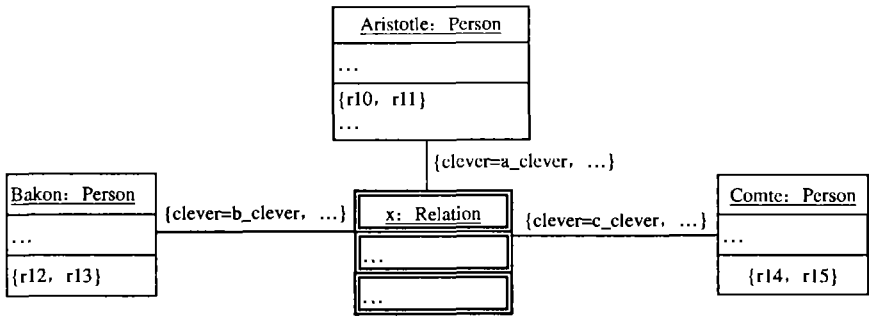


图4

4.2 三态逻辑

在OOPL中，对象的命题变元真值在某个时候可能是未知的。为了在真值未知的情况下使逻辑演算和推理顺利进行，这里引入三态逻辑：即除了真、假外再引入一个值"未知"，暂用NULL表示。三态逻辑的运算法则是：

∨运算	0	1	NULL
0			NULL
1			1
NULL	NULL	1	NULL

∧运算	0	1	NULL
0			0
1			NULL
NULL	0	NULL	NULL

→运算	0	1	NULL
	1	0	NULL

4.3 基于OOPL的正向演绎推理

基于OOPL的正向演绎推理是通过逻辑真值在对象网络中不同对象之间进行真值传递和规则触发来完成的。所谓真值传递是指镜像变量之间相互传递真值的过程；而规则触发是指在对象内部根据产生式类约束由前件为真推出后件为真，或由后件为假推出前件为假的过程(前已叙及)。

在本例中，如假设已知B是聪明人，问A、B、C各自考试的情况是什么？推理过程如下：

1)首先系统进行初始化，它创建各个需要的对象实例并建立相关连接，在该过程中各个对象的命题变元成员变量的值被初始化为NULL；

2)用户将B(指对象Bakon，下同)的clever置为1；

3)在B中根据规则r13引起规则触发，以致B. ph=1；

4)此后引发B向命题关联对象x发送消息进行真值传递，将clever的真值传递给x的b_clever，将ph的真值传递给x的b_ph；

5)根据x中规则r2，b_clever为1引发规则触发，使a_clever、c_clever为0；随即根据x中规则r6，b_clever、b_ph为1触发a_ph=0、c_ph=0、b_ch=0、a_ch=1、c_ch=1；

6)上一步得到的结果a_clever=0、a_ph=0、a_ch1，b_ch=0，c_clever=0、c_ph=0、c_ch=1再一次引发真值传递，刷新各自对象A、B和C的镜像变量；

7)在A中，clever为0触发规则r10，得到的ph=0和x真值传递来的ph=0一致，说明推理正确，；同理，C中触发r14，得到的结果也是一致的；而B中ch=0不会再触发任何规则，因此只是简单地赋值；

8)此时系统中所有的命题变量都已得到明确的真值，真值传递和触发的过程到此结束，系统此时的状态如下：

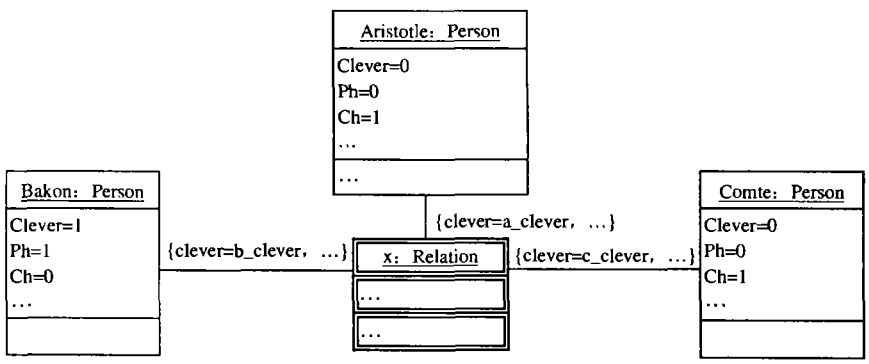


图5

4.4 基于OOPL的逆向演绎推理

基于OOPL的逆向演绎推理是一种基于逻辑的反证法，其基本思想是先假设结论为假，然后根据这一假设在网络中进行反向真值传递和触发，如果出现矛盾，则说明原来的假设错误并得到结论为真的断言。

还是以"聪明人考试问题"为例，但现取消B为聪明人的已知条件，问谁是聪明人？解决该问题的思路是分别假设A、B、C为聪明人，然后验证是否存在矛盾。

先假设A为聪明人，推理过程如下：

1)首先系统进行初始化，将各个对象的命题变元成员变量的值初始化为NULL；

2)用户将A. clever置为1；

3)在A中根据规则r11触发A. ch=1；

4)此后引发A向命题关联对象x发送消息进行真值传

递，将 clever 的真值传递给 x 的 a_clever，将 ch 的真值传递给 x 的 a_ph；

5)根据 x 中规则 r1，a_clever 为1触发 b_clever、c_clever 为0；随即根据 x 中规则 r5触发 b_ch＝0、c_ch＝0、a_ph＝0、b_ph＝1、c_ph＝1；

6)上一步得到的结果再一次引发真值传递，刷新对象 A、B 和 C 各自的镜像变量；

7)此时在 C 中，根据真值传递应得 clever＝0、ph＝1、ch＝0；然而 clever＝0将触发规则 r14使 ph＝0，于是产生一个矛盾；

8)C 检测到该矛盾，向 x 返回一个错误信息，x 收到后又向 A 返回错误消息；

9)最后 A 收到矛盾的消息就向用户返回，用户便得出结论 A.clever 不能为真。

然后用户假设 C.clever 为真，通过同样的步骤进行反证推理同样得出矛盾，于是断定为假。

最后，用户根据 A 和 C 不是聪明人将 A.clever、C.clever 置为0，然后再进行真值传递和触发过程就可以得到 B.clever＝1这一结果。这样就完成了本问题。

需要说明的是，如果用户假设 B.clever 为1并进行反证不会得出任何矛盾，但不能就此断定结论的正确；在这种情况下应该理解为推理得不出任何有用结果，必须进行进一步推理。

4.5 基于 OOPL 的规则证明

在基于 OOPL 的正向推理和逆向推理基础上就可以进行基于 OOPL 的规则证明，其方法是假设规则的前件为真且后件为假，然后进行真值传递和触发推理，如果在推理过程中出现矛盾则假设不成立，从而得到规则为真的断言。重要的是，在证明之前必须将欲证明的规则封装在一个命题关联类中，然后生成该类的实例再进行推理。

还是以“聪明人问题”为例，假设现在欲证明“A 和 C 通过的考试科目总是一样的”，将其用规则形式描述得到：

r16：A.ph→C.ph

r17：C.ph→A.ph

r18：A.ch→C.ch

r19：C.ch→A.ch

在 OOPL 中首先新建一命题关联类 Equivalency：

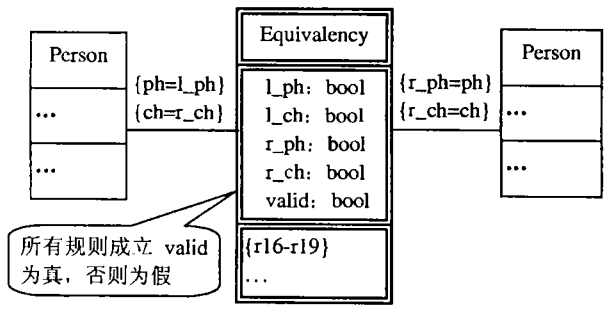


图6

然后建立类的实例，由于是推断 A、C 之间的逻辑关系，故须创建一个 A、C 之间的关联对象 ac(见图7)：

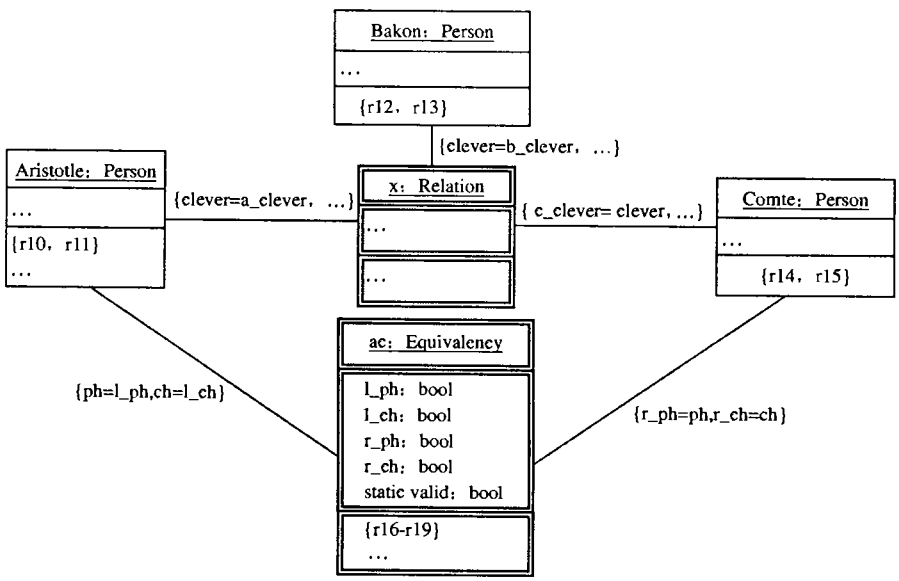


图7

最后进行推理过程：

1)首先系统进行初始化，将各个对象的命题变元成员变量的值初始化为 NULL；

2)用户向 ac 提出推理请求，ac 收到后启动一个对应的例程，该例程将依次验证本对象规则的合法性；

3)先验证规则 r16：推理例程将规则 r16的前件置为真、后件置为假，得到 l_ph＝1、r_ph＝0，然后引发真值传递；

4)A 中由 ph＝1和 r10触发 clever＝1(由后件假推出前件假)，并使 x.a_ph＝1、x.a_clever＝1；

5)x 中由 x.a_clever＝1和规则 r1触发 b_clever、c_clever 为0，并进行相应真值传递；

6)B 中由 b_clever＝0和规则 r12触发 B.ch＝0，并使 x.b_ch＝0；

7)x 中由 a_ph＝1、a_clever＝1和规则 r 触发 b_ch＝1；但这和第6)步 x.b_ch＝0存在矛盾，因此断定假设不成立，从而规则 r16为真。

8)接下来验证规则 r17到 r19，其方法与上面相同，这里不再赘述；

9)最终推理例程验证完所有的规则，并得到它们均为真的结果，于是 ac 将赋值其成员 valid 为真并返回消息给用户

(否则它将置 valid 为0然后返回)。

参 考 文 献

1 Bonacina M P. On the reconstruction of proofs in distributed theorem proving: a modified Clause-Diffusion method. Journal of Symbolic Computation, 1996, 21: 4～6, 507～522

2 Bonacina M P, Hsiang J. The Clause-Diffusion methodology for distributed deduction. Fundamenta Informaticae, 1995, 24: 177～207

3 Bonacina M P, Hsiang J. Parallelization of deduction strategies: an analytical study. Journal of Automated Reasoning, 1994, 13: 1～33

4 胡春华，吴波，杨叔子. 基于多自主体的分布式智能控制系统研究. 中国机械工程，1999，9(7)

5 王万森. 人工智能原理及其应用. 电子工业出版社，2000

6 杨炳儒. 知识工程与知识发现. 冶金工业出版社，1998

7 http://www.tcd.ie/Psychology/Ruth_Byrne/mental_models/critics.html

8 http://www.prdm.net/papers/knowledge/KM_20for_20CE.htm

9 http://www.cs.cf.ac.uk/Dave/AI2/node102.html

(上接第81页)

* 10<50，同步粒度也可以接受。

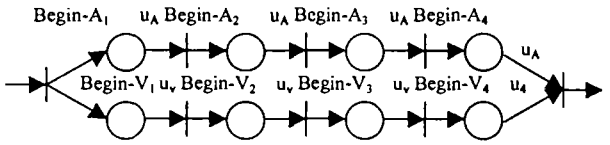


图4 当 n=4时的唇同步

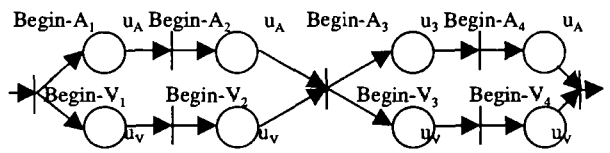


图5 当 n=2的唇同步

5 异常处理

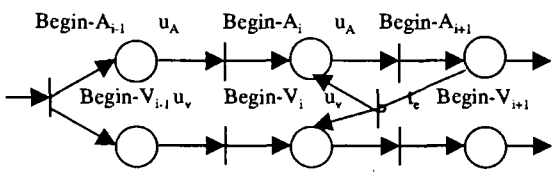


图6 STPN 的异常处理

为了解决对象间因同步等待引起的“时间狭缝”(gap problem)[5]，“restricted blocking”提供了一种解决的办法。通常该方法被用在同步点时刻，实际上最终难以接受的扭曲主要是流内的偏移量超出 QoS 给定的最大抖动而导致的，因此我们可借鉴文[8]思想来解决此类问题，但需要说明的是，我们的模型，对音频和视频采取了不同的处理方法，如图6上边的媒体流为音频。

在实时多媒体通信中，多媒体数据通常采用非可靠传输协议(如 UDP，RTP 等)进行传输的，数据包的丢失是不可避免，因此，有必要研究同步模型对象 LDU 丢失对同步的影响，可现有的 Petri 网同步模型均没有考虑对象丢失的情况。当 j≥η 时，即媒体对象的第 m 个 LDU 受到较大的延时，超出规定的时间范围，通常在要求不高的时候，原则上应该丢弃，但本模型对音频、视频采用了不同的处理方法。由于人的听觉对音频变化比较敏感，因而不能随意丢弃音频流的 LDU，可以引入抑止弧来进行如图6所述的异常处理。对于视频媒体流来说，丢失有限几个的媒体单元不会对用户带来明显的影响，少量的可直接丢弃。由于 RTS 和表现时间 u_i 已知，因而当 LDU_k(k 为自然数，指第 k 个 LDU)丢失时，只要在库所 p_k 插入相同数目与 LDU_k 有相同时间属性的延迟节点，就不会影响后续媒体的同步。可见该模型对于一定程度的对象丢包具有鲁棒性。

结论 本文针对时间流 Petri 网的不足，引入了 STPN 模型，有效地解决了多媒体通信、播放的同步问题。它是细粒度模型，支持并保证服务质量，并根据不同的 QoS 要求，提供不同的同步响应，具有很强的适应性。另外，该模型对于一定程度的对象丢包具有鲁棒性，不会影响到后续媒体的同步。

今后，我们需要进行仿真试验，验证 STPN 模型用于面向 Internet 的实时多媒体通信时的实际特性，需要进一步研究对该模型的性质分析(如可达性、活性、公平性[11])，以及对该模型的多媒体同步机制问题等。

参 考 文 献

1 Little T D C, Ghafoor A. Synchronization and Storage Models for Multimedia Objects. IEEE Journal on Selected Area in Communications, 1990, 8(3): 413～427

2 Woo M, Quzi N U, Ghafoor A. A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information. IEEE Network, 1994, 8(1): 52～61

3 Tan Kun, Shi Yuan-chun, Xu Guang-you. Supporting weak synchronization over world wide Web. Journal of Software, 2000, 11(7): 853～862

4 Senac P, Diaz M. Modelling logical and temporal synchronization in hypermedia systems [J]. IEEE Journal on Selected Area in Communications, 1996, 14(1): 257～273

5 Steinmetz R. Synchronization properties in multimedia systems. IEEE Journal on Selectde Area in Communications, 1990(April): 401～412

6 单志广，杨扬，王任，于兴业. 扩展的时间流 Petri 网多媒体模型，计算机研究与发展，2000，37(2)：223～273

7 Zhou Y, Murata T. Fuzzy-timing Petri net model for distributed multimedia synchronization. In: Proc. of the 1998 IEEE Int. Conf. on Systems, Man and Cybernetics, Lolla, California, 244～249

8 Han Ying-jie, Sun Yong-qiang, Wu Zhe-hui. Fine-grained distributed multimedia synchronization model-Enhanced Fuzzy-timing Petri net. Journal of Shanghai Jiaotong University, 2001, E-6(1): 62～66

9 Prabhakaram B, Raghavan S V. Synchronization models for multimedia presentation with user participation. Multimedia Systems, 1994, 2(2): 53～62

10 Allen J F. Maintaining knowledge about temporal intervals. Communication of the ACM, 1983, 26(11): 832～843

11 吴哲辉. 有界 Petri 网的活性和公平性的分析与实现. 计算机学报，1989，12(4)：267～278