一种基于逻辑程序的重复协商框架*)

陈 武[1,2] 张明义[3]

(贵州大学计算机科学与技术学院 贵阳 550025)[1] (西南大学计算机与信息科学学院 重庆 400715)[2]

(贵州科学院 贵阳 550001)[3]

摘 要 本文从基于信念修改的角度提出了两个AGENT之间的一个重复协商框架。在这个框架中,一个逻辑程序被当作一个协商的AGENT,每一个AGENT(逻辑程序)选择自己的一个回答作为自己最初的协商需求。两个AGENT之间的协商过程就是两个逻辑程序之间相互更新的过程,这个过程是通过协商的每一方接受对方的部分(或全部)需求和放弃自己部分协商需求来实现的。本文设计了协商双方必须遵守的一些协商规则,根据这些规则对这个协商框架进行了形式化描述,并给出了协商的终止条件。

关键词 AGENT,协商,扩展逻辑程序

A Repeated Negotiation Framework Based on Logic Programs

CHEN Wu[1,2] ZHANG Ming-yi[3]

(School of Computer Science and Technology, Guizhou University, Guiyang 550025, China)[1]

(School of Computer and Information Science, Southwest University, Chongqing 400715, China)[2]

(Guizhou Academy of Sciences, Guizhou 550001, China)[3]

Abstract An extended logic program is regarded as a negotiating agent. The negotiation process between two agents is then modelled as repeated encounters between two extended logic programs, each of which selects an answer set as its initial demand. The overall dynamics can be regarded as mutual updates between two extended logic programs. We design some negotiation rules which both extended logic programs must follow during the negotiation. These rules can guarantee the termination of the negotiation. On the basis of these rules, we propose a framework for repeated encounters between two agents, and examine some conditions of termination of such negotiation.

Keywords AGENT, Negotiation, ELP(Extended Logic Program)

1 引言

在现实社会中,多个AGENT之间的协商是非常普遍的。例如从国家之间达成的协议到家庭之间或朋友之间达成的一般谅解。协商通常也是一个重复的过程。在其中每一轮协商过程中,协商双方会互相做一些交易直到最终协议达成。协商问题是博弈论研究中一个较早被关注的问题,并且已经产生了各种各样的对协商进行形式化的方法。具体例子请参见参考文献[6,8]。

近年来,人们从信念修改的角度提出了另外一类关于协商的形式化的方法。它的基本思想就是当两个AGENT协商时,协商双方彼此提出一些需求,这些需求是用一些逻辑语言的公式集合来表示的。然后每一个AGENT通过猜测对方的需求来判断哪些需求是自己可以接受的,在必要的情况下为了达成协议也可以放弃(或收回)自己最初提出的部分需求。这些需求被解释为AGENT的信念是信念修改作为这种放弃或接受需求的理性机制的基础。每一轮协商的结果就是AGENT之间信念的相互修改的结果。此外,有一篇关于相互信念修改得很好的文献[9]能被用于对重复协商进行建模。

经典的博弈论的形式化方法和最近的基于逻辑的形式化方法是互补的。前者的优点就是具有对AGENT做出决策的高度抽象的观点而后者是具有获取那些关于AGENT推理的细节问题的能力。把基于逻辑的表示方法看作是博弈论的经典的宏观的AGENT表示的基石是比较恰当的。

文献[1]采用这种基于逻辑的方法研究了一次协商的情况,文献[2]考虑了正规博弈的多次协商。本文进一步精炼了这种基于逻辑的AGENT协商模型。

本文主要考虑两个AGENT之间的协商。本文把每一个AGENT具体模型化为一个扩展逻辑程序(ELP)。这个ELP的所有回答作就是这个AGENT自己的所有协商需求。在每一轮协商中,每一个AGENT都将选择自己的一个回答集作为当前的需求。然后每一方将通过接受对方回答集中的一些文字并且放弃自己回答集的部分文字来改变自己的程序。文献[10]证明了这样的修改程序的算法是可行的。由于双方相应的程序已经被修改,两个AGENT将带着新的需求进入下一轮协商。最终协商过程以达成协议或者失败结束。这种重复协商对应着经典的扩展博弈。本文与上述引用的文献的主要区别在于:这篇文章考虑了同时处理接受和放弃文字,而上述文献或者只考虑一般模型的公理协商或者仅考虑逻辑程序之间放弃文字。

现实的基于逻辑的协商模型还应该考虑AGENT策略,然后也许会涉及学习。我们推迟这些研究直到我们更好地理

*)本文受国家自然科学基金(60573009)和贵州省长基金(2005(12))资助。陈 武 工程师,博士研究生,主要研究方向为逻辑程序、非单调逻辑;张明义 研究员,博士生导师,主要研究方向人工智能、非单调逻辑等。

解不具有策略 AGENT 的 ELP 模型的结构和行为。因此在这篇文章我们仅仅抽象地描述了这样的细节，即一个 AGENT 怎样选择一个特殊的回答集作为自己的协商需求。同时，知道哪些文字是另外一个 AGENT 愿意接受或者拒绝的，这表明了本文的目的就是证明 ELP 作为协商模型是合理的，尤其可以获取类似一些标准的博弈论的概念。

本文第 2 部分简要地描述了 ELP 的基本概念和两个 ELP 之间一次协商的一些结论。第 3 部分我们提出了两个 ELP 之间的一个重复协商的框架。在这部分的第 1 小节介绍了重复协商的概念和一些协商规则。第 2 小节描述了重复协商的过程和协商中止的条件。总结部分概括了本文取得的主要工作和存在的不足，以及在未来的工作中弥补这些不足的计划。

2 ELP 及相关基础知识

本文只考虑命题逻辑语言环境。在这一节我们回忆文献[4,5,7]中的一些逻辑程序的基本概念。

一个 ELP 是下面这种形式的规则的有穷集合：

$L_0 \leftarrow L_1, \cdots, L_m, \text{not } L_{m+1}, \cdots, \text{not } L_n (0< m< n)$，

其中，每一个 $L_i (0\leqslant i\leqslant n)$ 是命题语言 L 下的一个文字，并且一个文字是一个正原子 a 或者负原子 $\neg a$。假设 r 是如同上述形式的一条规则，我们用 $Head(r)=\{L_0\}$，$Pos(r)=\{L_1, \cdots, L_m\}$ 和 $Neg(r)=\{L_{m+1}, \cdots, L_n\}$ 分别来表示 r 的头，正的规则体以及负的规则体。但是 $Head(r)$ 和 $Pos(r)\cup Neg(r)$ 不应该同时为空集 $\emptyset$。

我们称一个 ELP Π 是一个基础程序如果 $Neg(\Pi)=\emptyset$。假定 Lit 是出现在一个基础程序 Π 的所有文字的集合，一个文字集合 S 是 Π 的回答集，如果 S 是 Lit 的最小子集并且使得：

(1)对任意 $r\in\Pi$，如果 $Pos(r)\subseteq S$，那么 $Head(r)\subseteq S$；

(2)如果 S 包含一对互补的文字(形如 a，$\neg a$)，那么 $S=Lit$。

我们用 $Cn(\Pi)$ 表示基础程序 Π 的回答集集合。

现在假定 Π 是一个 ELP 并且 Lit 表示 Π 的语言中所有文字的集合。对于任意集合 $S\subseteq Lit$，我们可以通过从 Π 中删除以下规则或公式来获取一个基础程序 Π^S：

(1)每一个 $r\in\Pi$，如果 $Neg(r)\cap S\neq\emptyset$；

(2)在其余规则里所有的 not L。

S 是 Π 的回答集如果 S 是 Π^S 的回答集。一个程序的回答集是协调的，如果这个回答集不包含互补的文字对；否则这个回答集是不协调的。

举例来说，令 $\Pi_1=\{a\leftarrow\}$。很显然，Π_1 只有唯一的一个回答集$\{a\}$。令 $\Pi_2=\{a\leftarrow\text{not } b; b\leftarrow\text{not } a; c\leftarrow b\}$，$\Pi_2$ 有两个回答集$\{a\}$和$\{b, c\}$。令 $\Pi_3=\{a\leftarrow\text{not } a\}$，$\Pi_3$ 没有回答集。我们用 $ANS(\Pi)$来表示一个 ELPΠ 的所有回答集的集合。

本文将采纳一个对于 ELP 修改非常有用的方法。这个方法是在文献[1]里被首次提出。这个方法主要意思是如何放弃某个回答集中的某些文字，使这个回答集中除了放弃的这些文字外其余文字保持不变。令 Π 是一个 ELP，它表示一个 AGENT。假设从 Π 的一个回答集中放弃文字 L。放弃文字 L 的方法如下所述：

(1)对于 Π 中 $Head(r)=L$ 和 $L\in Pos(r')$ 的任意一对规则 r 和 r'，在 π 中增加规则 $Head(r')\leftarrow Pos(r), Pos(r')-\{L\}$，not $Neg(r)$，not $Neg(r')$；

(2)删除 $L\in Pos(r)$ 的所有规则 r；

(3)删除 $Head(r)=L$ 的所有规则 r；

(4)删除所有规则体中包含 not L 的所有规则。

修改后得到的新程序用 $SReduce(\Pi, L)$ 表示。类似地，$SReduce(\Pi, X)$ 表示 Π 用这个方法放弃文字集合 X 后获得的新程序。文献[1]提出了一个相关的结论如下：

命题 1 假定 S 是一个 ELP Π 的一个回答集。如果 $X\subset S$，那么 $S\backslash X$ 是 $SReduce(\Pi, X)$ 的一个回答集。

3 两个 ELP 之间重复协商的框架

在本节里，我们将提出两个 ELP 之间重复协商的一个框架。它包括以下四部分：

(1)一个协商集合，表示所有 AGENT 提出的需求空间；

(2)一些协商规则，定义了 AGENT 提出的合法的需求；

(3)一组策略，每个 AGENT 提出一个策略。这个策略决定了将要提出的需求；

(4)一些协商终止条件，它决定了达到协议的具体时间及已经达成什么协议。

重复协商是一个多轮的相互协商的过程。在这个协商过程中，每一个 AGENT 最大可能地接受另外一个 AGENT 的需求并且使自己放弃的需求尽可能达到最小。每一个 AGENT 提供一个回答集作为自己最初的需求。在本文中，令两个 ELPΠ_{10} 和 Π_{20} 代表协商的双方，并且假定协商有 n 轮。Π_{10} (Π_{20})选择自己的一个回答集 S_{10} (S_{20})作为初始的协商需求。两个 AGENT 互相之间的交易可以看作是这两个回答集之间的一种交换。在第 i $(0<i<n)$ 轮协商后，通过修改 Π_{10} (Π_{20})可以获得新程序 Π_{1i} (Π_{2i})。用 S_{1i} (S_{2i})表示从 Π_{1i} (Π_{2i}) 的一个从特殊的回答集集合的子集中选出的一个回答集。协商最后达成的协议是 $S_{1n}\cap S_{2n}$。

3.1 重复协商和协商规则

两个 ELP 之间的重复协商是通过每一轮每个 AGENT 改变自己的需求(文字)来实现的。因此在讨论重复协商和协商规则之前，我们首先需要给出在每一轮协商中两个 ELP 之间接受文字的条件。

定义 1 令 Π_1 和 Π_2 表示两个 ELP。S_1 和 S_2 分别是 Π_1 和 Π_2 的一个非空的回答集。对任意 $X\subseteq S_2$，我们说 X 是可接受的当有一个集合 S'_1 满足下述条件：

$S'_1\in ANS(\Pi_1\cup X)$ 并且 $S_1\subseteq S'_1$

我们用 $S_1\rhd X$ 表示任意一个这样的集合 S_1'，并且进一步用$\{S_1\rhd X\}$表示所有的满足上述条件的回答集的集合。

直观地，我们把上述的 Π_1 和 Π_2 当作协商的双方并且 S_1 和 S_2 作为相应的初始需求。上面的定义说的是一个 AGENT 的部分需求能被对方接受只有当这一部分需求不损害对方的利益。换一句话说，如果对方接受这部分需求，对方的需求就会变为一个新的需求，这个新的需求包含原来旧的需求和新加入的这部分需求。可以用一个例子来说明上面的定义。

例 1 $\Pi_1=\{a\leftarrow\text{not } b;\ b\leftarrow a,\ c;\ d\leftarrow a,\ e\}$；$\Pi_2=\{c\leftarrow;\ \neg d\leftarrow;\ e\leftarrow\}$。$\Pi_1$ 有唯一的回答集 $S_1=\{a\}$；Π_2 也有唯一的回答集 $S_2=\{c, \neg d, e\}$。由于通过在 Π_1 中增加新的规则$\{c\leftarrow\}$获得的 Π_1' 没有相应的协调的回答集包含 S_1，文字 c 对于 S_1 是不能接受的，所以也不存在这样的集合 $S_1\rhd c$。文字 $\neg d$ 对于 S_1 是可以接受的，$\{a, \neg d\}$ 是 $\Pi_1\cup\{\neg d\cup\leftarrow\}$ 的相应的回答集 S_1'。但是文字$\{\neg d, e\}$对于 S_1 是不可以接受的，因为

$\Pi_1 \cup \{\rightharpoonup d\leftharpoondown; e\leftharpoondown\}$的相应的回答集 S_1'包含有一个互补的文字对 d 和$\rightharpoonup d$。

定义 1 和命题 1 产生了下述的两个 ELP 之间关于接受文字的一些结论。

命题 2 令 Π_1 和 Π_2 表示两个 ELP。S_1 和 S_2 分别是 Π_1 和 Π_2 的一个非空的回答集。对任意非空集合 $X \subseteq S_1$ 和 $Y \subseteq S_2$，如果存在 $S_1 \triangleright Y$，那么就有 $X \triangleright Y$；其中 $X \in ANS(SReduce(\Pi_1, S_1 \backslash X))$。

证明：根据命题 1 和定义 1，我们将证明存在这样的集合 $X \triangleright Y$，其中 $X \in ANS(SReduce(\Pi_1, S_1 \backslash X))$。可以分以下两种情形进行证明：$(S_1 \backslash X) \cap Y = \emptyset$ 和 $(S_1 \backslash X) \cap Y \neq \emptyset$。

情形 1：$(S_1 \backslash X) \cap Y = \emptyset$

由定义 1，我们知道由于存在这样的集合 $S_1 \triangleright Y$，因此 $S_1 \triangleright Y$ 是协调的并且 $S_1 \cup Y \subseteq S_1 \triangleright Y$。令 Π' 和 Π'' 分别表示 $SReduce(\Pi_1, S_1 \backslash X)$ 和 $\Pi_1 \cup Y$。由于 $(S_1 \backslash X) \cap Y = \emptyset$，$SReduce(\Pi''_1, S_1 \backslash X) = \Pi' \cup Y$。可以得出 $(S_1 \triangleright Y) \backslash (S_1 \backslash X) \in ANS(\Pi'_1 \cup Y)$ 并且 $X \cup Y \subseteq (S_1 \triangleright Y) \backslash (S_1 \backslash X)$。再一次由命题 1 知道，$(S_1 \triangleright Y) \backslash (S_1 \backslash X) = X \triangleright Y$。由于 $S_1 \triangleright Y$ 是协调的，所以 $X \triangleright Y$ 也是协调的。再一次运用定义 1，可以得出存在这样的集合 $X \triangleright Y$，其中 $X \in ANS(SReduce(\Pi_1, S_1 \backslash X))$。

情形 2：$(S_1 \backslash X) \cap Y \neq \emptyset$

假定 $Y' = (S_1 \backslash X) \cap Y$。由于 $(S_1 \backslash X) \cap Y \neq \emptyset$，$SReduce(\Pi'', S_1 \backslash X) = \Pi'' \cup (Y \backslash Y')$。由于 $(S_1 \backslash X) \cap (Y \backslash Y') = \emptyset$，根据情形 1 的证明，可以得出 $X \triangleright (Y \backslash Y')$，其中 $X \in ANS(SReduce(\Pi_1, S_1 \backslash X))$。根据 $SReduce$ 的方法，可以得出 $LIT(Y') \cap LIT(\Pi'_1 \cup (Y \backslash Y')) = \emptyset$。因此可以进一步得出存在这样的集合 $X \triangleright Y$，其中 $X \in ANS(SReduce(\Pi_1, S_1 \backslash X))$。

命题 3 令 π_1 和 π_2 表示两个 ELP。S_1 和 S_2 分别是 π_1 和 π_2 的一个非空的回答集。对任意非空集合 $X \subseteq S_2$ 和 $Y \subseteq S_2$，如果存在 $S_1 \triangleright X$ 使得 $Y \subseteq S_1 \triangleright X$，那么就有 $S_1 \triangleright (X \cup Y)$。

证明：令 Π' 和 Π'' 分别表示 $\Pi_1 \cup X$ 和 $\Pi_1 \cup X \cup Y$。由于存在这样的集合 $S_1 \triangleright X$ 使得 $Y \subset S_1 \triangleright X$ 并且 $\Pi'' = \Pi' \cup Y$，因此可以得出 $Cn(\Pi''^{S_1'}) = S'_1$ 和 $S'_1 \in ANS(\Pi'')$，其中 $S'_1 = S_1 \triangleright X$。根据定义 1，$S_1 \triangleright X$ 是协调的。由于 $S_1 \subseteq S_1 \triangleright X$，$X \subseteq S_1 \triangleright X$ 和 $Y \subset S_1 \triangleright X$，可以得出存在这样的集合 $S_1 \triangleright (X \cup Y)$ 并且它就是 S'_1。

推论 1 令 Π_1 和 Π_2 表示两个 ELP。S_1 和 S_2 分别是 Π_1 和 Π_2 的一个非空的回答集。对任意非空集合 $X \subseteq S_2$ 和 $Y \subseteq S_2$，对任意 $S_1 \triangleright X$ 使得 $Y \subseteq S_1 \triangleright X$ 并且对任意 $S_1 \triangleright Y$ 使得 $X \subseteq S_1 \triangleright Y$，那么就有 $S_1 \triangleright X = S_1 \triangleright Y$。

证明：明显地，由命题 3 的证明，有 $S_1 \triangleright X = S_1 \triangleright (X \cup Y)$ 并且 $S_1 \triangleright Y = S_1 \triangleright (X \cup Y)$。因此可以得出 $S_1 \triangleright X = S_1 \triangleright Y$。

为了协商，协商双方必须遵守一些大家都同意的协商规则。实际上，对每一个 AGENT 而言，每一轮协商是一种从当前双方的需求到修改后的新程序中获得的新的需求的关系。这种新的需求是通过接受对方回答集的一些文字并且同时放弃自己的部分文字而获得的。因此，如果要定义重复协商，首先我们必须定义一种遵守一些规则的关系。这些规则包括封闭性、包含性、扩充性等特性。

定义 2 令 Π_{10} 和 Π_{20} 表示两个 ELP。S_{10} 和 S_{20} 分别是 Π_{10} 和 Π_{20} 的一个非空的回答集。关系 $RN \subseteq (2^L \times 2^L) \times (2^L \times 2^L)$ 定义如下：

(1) $RN(\langle S_{10}, S_{20} \rangle, \langle S_{10}, S_{20} \rangle)$；

(2) $RN(\langle S_{10}, S_{20} \rangle, \langle S_{11}, S_{21} \rangle)$ 满足下述规则：

(RN1)封闭性：

$S_{1i} = Cn((\Pi_{1i})^{S_{1i}})$；$S_{2i} = Cn((\Pi_{2i})^{S_{2i}})$

$(i = 0, 1)$

其中，Π_{1i} 和 Π_{2i} 是两个 ELP。

(RN2)包含性：

$S_{1i} \subseteq Head(\Pi_{1i} \cup \Pi_{2i})$；$S_{2i} \subseteq (Head(\Pi_{1i} \cup \Pi_{2i})(i = 0, 1)$

(RN3)一致性扩张：

如果存在这样的集合 $S_{10} \triangleright S_{20}$ 和 $S_{20} \triangleright S_{10}$，那么

$S_{10} \cup S_{20} \subseteq S_{11}$；$S_{10} \cup S_{20} \subseteq S_{21}$

(RN4)单调性收缩：

$S_{10} \cup S_{20} \subseteq S_{11} \cup S_{21}$

(RN5)可接受性扩张：

(a)对所有集合 $X \subseteq S_{20} \backslash S_{10}$ 和 $Y \subseteq S_{20} \backslash S_{10}$，如果存在这样的集合 $S_{10} \triangleright X$ 和 $S_{10} \triangleright Y$ 并且 $|X| < |Y|$，那么 $Y \subseteq S_{11} \cap S_{21}$；

(b)对所有集合 $X \subseteq S_{10} \backslash S_{20}$ 和 $Y \subseteq S_{10} \backslash S_{20}$，如果存在这样的集合 $S_{20} \triangleright X$ 和 $S_{20} \triangleright Y$ 并且 $|X| < |Y|$，那么 $Y \subseteq S_{21} \cup S_{11}$。

(RN1)说的是每一轮协商的结果是修改后的程序的新的需求。(RN2)说的是协商每一方在每一轮协商后获得的结果是两个协商双方 ELP 头文字并集的子集。换句话说就是协商的范围不会超出双方可能的协商空间。(RN3)说的是如果每一方能接受对方的需求，协商双方将达成一个交易。这个交易的结果就是每一方将保留当前自己的需求并且增加从对方获得的可以接受的需求直接进入到下一轮。直观地，(RN4)表示的是上一轮达成的协议必须在下一轮继续保留。换句话就是协商的任何一方都不愿意放弃已经达成的协议。(RN5)说的是为了避免协商失败，每一个 AGENT 将尽量多地接受对方的需求，只要这些需求不损害自己的利益。

定义 3 令 Π_{10} 和 Π_{20} 表示两个 ELP。S_{10} 和 S_{20} 分别是 Π_{10} 和 Π_{20} 的一个非空的回答集。$RN(\langle S_{10}, S_{20} \rangle, \langle S_{1n}, S_{2n} \rangle)$ 是一个有 n 轮的重复协商当且仅当 $RN(\langle S_{1(i-1)}, S_{2(i-1)} \rangle, \langle S_{1i}, S_{2i} \rangle)(0 < n; 0 < i(n)$。

重复协商的概念自然导出一个关于它的产生过程和终止属性的解释。

3.2 重复协商的过程和终止条件

重复协商是一个关于两个 ELP 的需求(回答集)互相修改的过程。我们把这个修改过程分为两个部分：收缩和扩张，类似于文献[3]中关于信念修改的原始概念描述。回答集的收缩指的是放弃该回答集自身的部分文字；回答集的扩张指的是把对方回答集的一些可以接受的文字增加到自己的回答集中。扩张是通过把对方回答集中可以接受的文字作为新的事实加入到自己的程序中来实现的。重复协商的过程依靠每一轮每个 AGENT 的选择，可以用下面的抽象的函数来形式化描述。

两个这样的抽象函数用于这个目的。第一个函数 δ，称之为选择函数，该函数的功能是从两个非空的回答集集合中选择出两个回答集。第二个函数 γ，称之为接受放弃函数，该函数用于分别从两个回答集中选出部分子集。

定义 4 令 Π_1 和 Π_2 表示两个 ELP。函数 $\delta: 2^{ANS(\Pi_1)\backslash\Phi} \times 2^{ANS(\Pi_1)\backslash\Phi} \rightarrow ANS(\Pi_1)\backslash\Phi \times ANS(\Pi_1)\backslash\Phi$ 被称为关于 (Π_1, Π_2) 的选择函数当且仅当对于任意的 $A_i \in 2^{ANS(\Pi_i)\backslash\Phi}$，$\delta(A_1, A_2) = (S_1, S_2)$ 使得 $S_i \in A_i (i = 1, 2)$。

直观地，上面的定义告诉我们在每一轮协商开始时，协商的每一方将选择自己的一个特殊的非空回答集做为自己的新的需求。

定义5 令 Π_1 和 Π_2 表示两个 ELP。函数 γ:ANS(Π_1)×ANS(Π_2)→($2^L\times 2^L$)×($2^L\times 2^L$)被称为一个关于(Π_1,Π_2)的接受放弃函数当且仅当对于任意 $S_i\in ANS(\Pi_i)(i=1,2)$，$\gamma(S_1,S_2)=((R_1,G_1),(R_2,G_2))$使得 $R_1\cup G_1\subseteq S_1\setminus S_2$，$R_1\cap G_1=\emptyset$，$R_2\cup G_2\subseteq S_2\setminus S_1$并且 $R_2\cap G_2=\emptyset$。

上述定义直观意思是协商的每一方将从自己的需求中选出能被对方接受的部分需求。同时，为了避免协商失败，每一方也将放弃自己的部分需求。

下面的定义给出了每一轮协商达成的交易的形式化描述。

定义6 令 Π_1 和 Π_2 表示两个 ELP。Π_1 和 Π_2 分别选择自己的一个非空的回答集 S_1 和 S_2 作为自己的需求。假定 γ 是一个关于(Π_1,Π_2)的接受放弃函数。我们称在(S_1,S_2)上的交易是一对(R_1,R_2)当且仅当存在这样的集合 $S_1\setminus G_1\rhd R_2$，$S_2\setminus G_2\rhd R_1$，其中 $\gamma(S_1,S_2)=((R_1,G_1),(R_2,G_2))$，$S_1\setminus G_1\in ANS(SReduce(\Pi_1,G_1))$并且 $S_2\setminus G_2\in ANS(SReduce(\Pi_2,G_2))$。

在(S_1,S_2)上所有可能的交易被称为 S_1 和 S_2 上协商集。

令 Π_1 和 Π_2 表示两个 ELP。Π_1 和 Π_2 分别选择自己的一个非空的回答集 S_1 和 S_2 作为自己的需求。Π_1 和 Π_2 之间的重复协商过程描述如下：

(1)每一个 AGENT 希望对方能接受自己回答集的所有文字，所以协商的每一方都提供整个回答集作为自己的需求。双方的最初需求分别是 S_1 和 S_2；

(2)如果每一方能接受对方提出的需求，每一个 AGENT 将对方的需求作为新的事实加入自己的程序中。那就有集合 $S_1\rhd S_2$ 和 $S_2\rhd S_1$。如果每一方没有因此而产生除了对方的需求以外的其它文字，则双方将达成一个交易并且协商终止。协商的结果是($S_1\cup S_2$,$S_2\cup S_1$)，双方达成的相应的协议是 $S_1\cup S_2$。如果产生了其余的文字，双方也将达成一个交易并且协商进入下一轮。

(3)如果每一方都不能接受对方的需求，为了使对方能尽可能多地接受自己的需求，每一方将选择自己当前的部分需求作为自己新的协商需求，分别用 R_1 和 R_2 表示；同时，为了避免协商失败，每一方将放弃部分当前需求从而使得协商双方获得最大利益。存在如下两种情形：

a)存在一个关于(Π_1,Π_2)的接受放弃函数 γ 使得(R_1,R_2)是(S_1,S_2)上的一对交易，其中 $\gamma(S_1,S_2)=((R_1,G_1),(R_2,G_2))$。如果每一方将不因为接受对方的部分需求而产生除这些需求以外的新的文字，并且 $S_1\setminus S_2=R_1\cup G_1$，$S_2\setminus S_1=R_2\cup G_2$，则协商将终止。协商结果是$((S_1\setminus G_1)\cup R_2,(S_2\setminus G_2)\cup R_1)$，达成的相应的协议是$(S_1\setminus G_1)\cup(S_2\setminus G_2)$。否则，双方也将达成一个交易并且协商进入下一轮。

b)如果不存在一个关于(Π_1,Π_2)的接受放弃函数 γ 使得(R_1,R_2)是(S_1,S_2)上的一对交易，其中 $\gamma(S_1,S_2)=((R_1,G_1),(R_2,G_2))$，则协商破裂。协商的结果是($S_1$,$S_2$)，并且达成相应的协议是 $S_1\cap S_2$。

上述非形式化的过程描述可以通过下面的定义形式化表示：

定义7 令两个 ELP Π_{10} 和 Π_{20} 表示重复协商的双方，δ_0 表示关于(Π_{10},Π_{20})的最初选择函数。$n(n\geqslant 0)$轮重复协商$\langle S_{1n},S_{2n}\rangle$的结果如下定义：

(1)$(S_{10},S_{20})=\delta_0(ANS(\Pi_{10}),ANS(\Pi_{20}))$；(2) 对于 $n>0$，如果 $RN(\langle S_{1(n-1)},S_{2(n-1)}\rangle,\langle S_{1n},S_{2n}\rangle)$，那么存在一个关于($\Pi_{1(n-1)}$,$\Pi_{2(n-1)}$)的接受放弃函数 $\gamma_{(n-1)}$ 和一个关于(Π_{1n},Π_{2n})的选择函数 δ_n 使得

$$(S_{1n},S_{2n})=\begin{cases}(S'_{1n},S'_{2n}),\text{如果存在 } S_{1(n-1)}\rhd S_{2(n-1)}\\ \text{和 } S_{2(n-1)}\rhd S_{1(n-1)};\\ (S''_{1n}S''_{2n}),\text{如果存在}(S_{1(n-1)}\setminus G_{1(n-1)})\\ \rhd R_{2(n-1)}\text{和}(S_{2(n-1)}\setminus G_{2(n-1)})\rhd R_{1(n-1)};\\ (S_{1(n-1)},S_{2(n-1)}),\text{否则。}\end{cases}$$

其中：

$\gamma_{(n-1)}(S_{1(n-1)},S_{2(n-1)})=((R_{1(n-1)},G_{1(n-1)}),(R_{2(n-1)},G_{2(n-1)}))$，$S_{1(n-1)}\setminus G_{1(n-1)}\in ANS(SReduce(\Pi_{1(n-1)},G_{1(n-1)}))$，$S_{2(n-1)}\setminus G_{2(n-1)}\in ANS(SReduce(\Pi_{2(n-1)},G_{2(n-1)}))$，$(S'_{1n},S'_{2n})=\delta_n(S_{1(n-1)}\rhd S_{2(n-1)},S_{2(n-1)}\rhd S_{1(n-1)})$，$(S''_{1n},S''_{2n}=\delta_n((S_{1(n-1)}\setminus G_{1(n-1)})\rhd R_{2(n-1)},(S_{2(n-1)}\setminus G_{2(n-1)})\rhd R_{1(n-1)})$

用下面一个例子来说明这个定义：

例：令两个 ELP Π_{10} 和 Π_{20} 表示重复协商的双方，其中 $\Pi_{10}=\{a\leftarrow;d\leftarrow b;f\leftarrow b,\text{ not } e\}$，$\Pi_{20}=\{b\leftarrow;e\leftarrow a;\ c\leftarrow a,\text{ not } d\}$。可以知道 $ANS(\Pi_{10})=\{\{a\}\}$，$ANS(\Pi_{20})=\{\{b\}\}$。在协商的第一轮，存在一个关于(Π_{10},Π_{20})的选择函数 δ_0 使得 $(S_{10},S_{20})=\delta_0(ANS(\Pi_{10}),ANS(\Pi_{20}))=(\{a\},\{b\})$。根据定义，存在集合$\{a\}\rhd\{b\}$和 $\{b\}\rhd\{a\}$，可以得出$\{\{a\}\rhd\{b\}\}=\{\{a,b,d,f\}\}$和$\{\{b\}\rhd\{a\}\}=\{\{a,b,c,e\}\}$。因此，存在一个关于($\Pi_{11}$,$\Pi_{21}$)的选择函数 δ_1 使得$(S_{11},S_{21})=\delta_1(\{\{a\}\rhd\{b\}\},\{\{b\}\rhd\{a\}\})=(\{a,b,d,f\},\{a,b,c,e\})$。由于 $S_{10}\cup S_{20}\subset S_{11}$并且 $S_{10}\cup S_{20}\subset S_{21}$，协商将进入下一轮。在第二轮协商中，存在一个关于(Π_{11},Π_{21})的接受放弃函数 γ_1 使得存在这样的集合$\{a,b,d\}\rhd\{e\}$和 $\{a,b,e\}\rhd\{d\}$，其中 $\gamma_1(\{a,b,d,f\},\{a,b,c,e\})=((\{d\},\{f\}),(\{e\},\{c\}))$，$\{a,b,d\}\in ANS(SReduce(\Pi_{11},\{f\}))=ANS(\{a\leftarrow,b\leftarrow,d\leftarrow b\})$，$\{a,b,e\}\in ANS(SReduce(\Pi_{21},\{c\}))=ANS(\{a\leftarrow,b\leftarrow,e\leftarrow a\})$。可以得出

$\{(S_{11}\setminus\{f\})\rhd\{e\}\}=\{\{a,b,d,e\}\}$，$\{(S_{21}\setminus\{c\})\rhd\{d\}\}=\{\{a,b,d,e\}\}$，$\Pi_{12}=\{a\leftarrow,b\leftarrow,d\leftarrow b,e\leftarrow\}$ 和 $\Pi_{22}=\{a\leftarrow,b\leftarrow,e\leftarrow a,d\leftarrow\}$。因此存在一个关于($\Pi_{12}$,$\Pi_{22}$)的选择函数 δ_2 使得$(S_{12},S_{22})=\delta_2(\{\{a,b,d,e\}\},\{\{a,b,d,e\}\})=(\{a,b,d,e\},\{a,b,d,e\})$。由于 $S_{12}=S_{22}$，协商终止，相应达成的协议是$\{a,b,d,e\}$。

由于协商的空间是有穷的，根据协商规则，每一轮新达成的协议都必须包含上一轮已经达成的协议，因此重复协商就一定会终止。

定理1 令 Π_{10} 和 Π_{20} 表示两个 ELP。对于任意的非空的 $S_{i0}\in ANS(\Pi_{i0})(i=1,2)$，存在 $n>0$ 使得$\langle S_{1n},S_{2n}\rangle$是 n 轮重复协商的结果并且这个结果恰好是下列三种情形之一：

(1)$S_{1(n-1)}\cup S_{2(n-1)}=S_{1n}$并且 $S_{1(n-1)}\cup S_{2(n-1)}=S_{2n}$；

(2)$(S_{1(n-1)}\setminus G_{1(n-1)})\cup R_{2(n-1)}=S_{1n}$，$(R_{1(n-1)}\cup G_{1(n-1)})=S_{1(n-1)}\setminus S_{2(n-1)}$，$(S_{2(n-1)}\setminus G_{2(n-1)})\cup R_{1(n-1)}=S_{2n}$并且$(R_{2(n-1)}\cup G_{2(n-1)})=S_{2(n-1)}\setminus S_{1(n-1)}$；

(3)$S_{1n}=S_{1(n-1)}$并且 $S_{2n}=S_{2(n-1)}$。

证明：考虑 $n=1$ 和 $n>1$：

如果 $n=1$，由定义7，考虑下述三种情形。

情形1：存在一些这样的集合 $S_{10}\rhd S_{20}$ 和 $S_{20}\rhd S_{10}$。

(下转第239页)

层相互交互通信。

应用层：是关于权利所管理的内容使用方面的应用，为用户提供使用内容时的界面，包括内容选择、个人诚信和资金交易等。

协商层：是建立服务级的信任关系，包括接收许可证的请求、终端设备能力的评估等。协商层的服务级信任关系内容提供商所发送的内容类型或内容的实质。

权利表述与解析层：是最重要的一层，提供所需的数字版权管理服务的最小集，以连接上下层。该层的功能包括许可证的创建、保护的内容、权利受控的内容分发，以及许可证。更具体地讲，权利表述功能负责获取上层的权利信息，以在特定应用中提供版权保护的功能，采用权利表述语言(REL)来实现。权利解析用于解析不同的权利。另外，不同的权利表述语言具有不同的权利集，因此权利解析负责权利的转换、权利的重建、内容和权利的重新打包，像 MPEG-21 权利字典，就是用于不同版权管理的权利表述的转换。

DRE 上层：主要负责权利的管理、内容的访问控制和加密协议，包括高级的权利增强技术，如加密、数字水印和数字证书。

DRE 下层：是关于操作系统级的技术，主要负责低级的权利管理，包括设备驱动认证、防止恶意程序的非法内存访问等。微软的“安全音频路径技术”(SAP)就是一例。

物理层：即用户端设备。

分层的数字版权互操作架构，其突出的优点是在各个层的内部解决该层所要提供的功能，然后考虑各个层的接口功能，以完成较大的数字版权管理功能。

但是，分层的数字版权互操作架构思想目前还需要实际的检验。

(5) 面向服务架构的数字版权互操作系统

文献[10]面向服务架构的数字版权互操作系统的思想，是由五种服务构成服务驱动的架构，包括打包服务、更新服务、定购服务、许可证服务和策略服务。

结束语 数字版权管理系统的互操作问题，要考虑现有的封闭的数字版权管理系统。同时，安全[11]问题已经成为数字版权互操作所要特别考虑的问题。不能因为各类数字版权管理系统的互操作，却引发安全问题。因此，这将是数字版权管理互操作的一个新的研究和开发重点。另外，文中提出的几种数字版权管理系统的互操作方案，需要在实际中进一步验证，以不断完善改进。这里分层的数字版权管理互操作系统，具有一定研究前景，开发出实用系统并为用户所认可是下一步需要推进的。

参考文献

[1] Ku W, Chi Chi-Hung. Survey on the technological aspects of digital rights management[J]. Lecture Notes in Computer Science, 2004, 3225: 391-403

[2] Wang Xin. On DRM Interoperability and Compatibility [EB/OL]. http://www.ieee-ccnc.org/2005/conf.html, 2005

[3] Koenen R H, Lacy J, MacKay M, et al. The Long March to Interoperable Digital Rights Management [J]// Proceedings of the IEEE. 2004, 92(6): 883-897

[4] Polo J, Prados J, Delgado J. Interoperability Between ODRL and MPEG-21 REL[EB/OL]//Proceeding of ODRL International Workshop. 2004. http://odrl.net/workshop2004/program.html (2007-01-28)

[5] Prados J, Rodriguez E, Delgado J. Interoperability Between Different Rights Expression Languages and Protection Mechanisms [C]// Proceedings of the First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05) on Automated Production of Cross Media Content for Multi-Channel Distribution. 2005: 145-152

[6] Bradley B W, Maher D P. The NEMO P2P Service Orchestration Framework [C]//Proceedings of the 37th HICSS - Hawaii International Conference on System Sciences. 2004: 4641-4650

[7] Heileman G L, Jamkhedkar P A. DRM interoperability analysis from the perspective of a layered framework [C]//Proceedings of the 5th ACM workshop on Digital rights management table of contents. 2005: 17-26

[9] Jamknedkar A, Heileman L. DRM as a layered system [C]// Proceedings of the 4th ACM workshop on Digital rights management. 2004: 11- 21

[10] Filho F M F, de Albuquerque J P, de Geus P L. A Service-oriented Framework to Promote Interoperability Among DRM Systems[J]. Lecture Notes in Computer Science, 2006(4267): 124-127

[11] Taban G, Cardenas A A, Gligor V D. Towards a Secure and Interoperable DRM Architecture [C]// Proceedings of the ACM workshop on Digital rights management. 2006: 69 - 78

(上接第 212 页)

由于存在一些这样的集合 $S_{10} \triangleright S_{20}$ 和 $S_{20} \triangleright S_{10}$，因此可以得出 $S_{10} \cup S_{20} \subseteq S_{11}$ 和 $S_{10} \cup S_{20} \subseteq S_{21}$。假设存在 $X \subseteq S_{11} \setminus (S_{10} \cup S_{20})$ 或 $Y \subseteq S_{21} \setminus (S_{10} \cup S_{20})$，协商将进入下一轮，则 $n>1$，与 $n=1$ 矛盾。可以推导出 $S_{10} \cup S_{20} = S_{11} = S_{21}$。

情形 2：存在这样的集合 $(S_{10} \setminus G_{10}) \triangleright R_{20}$ 和 $(S_{20} \setminus G_{20}) \triangleright R_{10}$，其中 $(S_{10} \setminus G_{10}) \in ANS(SReduce(\Pi_{10}, G_{10}))$，$(S_{20} \setminus G_{20}) \in ANS(SReduce(\Pi_{20}, G_{20}))$。

由于存在如上所述的这些集合，由此可以得出 $(S_{10} \setminus G_{10}) \cup R_{20} \subseteq S_{11}$ 并且 $(S_{20} \setminus G_{20}) \cup R_{10} \subseteq S_{21}$。由情形 1 可知，$(S_{10} \setminus G_{10}) \cup R_{20} = S_{11} = S_{21}$。类似情形 1，也不存在任何集合 $X \subseteq (S_{10} \setminus S_{20}) \setminus (R_{10} \cup G_{10})$ 或者 $Y \subseteq (S_{20} \setminus S_{10}) \setminus (R_{20} \cup G_{20})$，否则协商将进入下一轮使得 $n>1$ 与 $n=1$ 矛盾。所以可以得出 $R_{10} \cup G_{10} = S_{10} \setminus S_{20}$ 和 $R_{20} \cup G_{20} = S_{20} \setminus S_{10}$。

情形 3：不存在任何关于 (Π_{10}, Π_{20}) 的接受放弃函数 γ 使得有这样的集合 $(S_{10} \setminus G_{10}) \triangleright R_{20}$ 和 $(S_{20} \setminus G_{20}) \triangleright R_{10}$。根据定义 7，有 $S_{11} = S_{10}$ 和 $S_{21} = S_{20}$。

如果 $n>1$，证明过程类似 $n=1$ 的情形。

结束语 本文把一个 ELP 看作是一个协商 AGENT，提出了一个关于两个 ELP 之间重复协商的形式化框架。作为下一步的工作，我们将引入带有优先关系文字的 ELP，对协商策略进行研究，从而进一步完善此框架。

参考文献

[1] Foo N, Meyer T, Zhang Y, et al. Negotiating logic programs// Proceedings of NRAC05, Sixth Workshop on Nonmonotonic Reasoning, Action and Change. Edinburgh, 2005

[2] Foo N, Meyer T, Brewka G. LPOD Answer Sets and Nash Equilibria. Lecture Notes in Computer Science, Springer-Verlag, 2004, 3321: 343-351

[3] Gardenfors P. Knowledge in Flux: Modelling the Dynamics of Epistemic States. MIT Press, 1988

[4] Gelfond, Lifschitz V. Classical negation in logic programs and disjunctive databases. New Generation Computing, 1991: 365-385

[5] Lifschitz V. Foundations of logic programming, in Principles of Knowledge Representation. CSLI Publications, 1996: 69-127

[6] Littman M L, Stone P. Implicit Negotiation in Repeated G-ames// Proceedings of The Eighth International Workshop on Agent Theories, Architectures, and Languages, August 2001: 393-404

[7] Lloyd J W. Foundation of logic programming. 2nd ed. New York: Springer-Verlag, 1987

[8] Sabourian H, Lee J. Complexity and Efficiency in Repeated G-ames with Negotiation// Econometric Society 2004 North American Summer Meetings. 2004: 58

[9] Zhang D, Foo N, Meyer T, et al. Negotiation as Mutual Belief Revision//Proceedings of AAAI04, Nineteenth National Conference on Artificial Intelligence: AAAI Press. The MIT Press, 2004: 317-322

[10] Zhang Y, Foo N. Solving Logic Program Conflict through Strong and Weak Forgettings. Artificial Intelligence, 2006, 170: 739-778