马尔可夫决策过程两种抽象模式*)

王蓁蓁[1] 邢汉承[1] 张志政[1,2] 倪庆剑[1]
(东南大学计算机科学与工程学院 南京 210096)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 抽象层次上马尔可夫决策过程的引入,使得人们可简洁地、陈述地表达复杂的马尔可夫决策过程,解决常规马尔可夫决策过程(MDPs)在实际中所遇到的大型状态空间的表达问题。介绍了结构型和概括型两种不同类型抽象马尔可夫决策过程基本概念以及在各种典型抽象 MDPs 中的最优策略的精确或近似算法,其中包括与常规 MDPs 根本不同的一个算法:把 Bellman 方程推广到抽象状态空间的方法,并且对它们的研究历史进行总结和对它们的发展做一些展望,使得人们对它们有一个透彻的、全面而又重点的理解。

关键词 情景演算,因子化马尔可夫决策过程,逻辑马尔可夫决策过程,关系马尔可夫决策过程,Bellman 方程

Two Classes of Abstract Modes about Markov Decision Processes

WANG Zhen-zhen[1] XING Han-cheng[1] ZHANG Zhi-zheng[1,2] NI Qing-jian[1]
(School of Computer Science & Engineering, Southeast University, Nanjing 210096, China)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[2]

Abstract Using Markov decision processes on abstract level, one can compactly and declaratively represent complex Markov decision processes. And one can solve real-world problems that generally have very large state space with regular MDPs. This paper introduces central concepts of two classes of abstract MDPs: structured mode and generalized mode. Then precise or approximate algorithms for looking for optimal policy for abstract MDPs are reviewed, including an algorithm that is totally different from regular Markov decision processes: A relational upgrade of the Bellman update operation. Finally, the paper gives conclusions of history work and suggests future work. In this way, people are able to get an intensive, comprehensive and in-depth understanding of abstract MDPs.

Keywords Situation calculus, Factored Markov decision processes, Logical Markov decision processes, Relational Markov decision processes, Bellman equations

1 引言

现今,由马尔可夫随机过程理论和增强学习方法相结合得到的马尔可夫决策过程 MDP 模式,越来越在人工智能的理论和应用领域受到重视。它对处于随机环境中的决策者做出最优规划以及在机器学习中解决(增强)学习问题,都是一个自然而重要的框架。在这个框架下,系统是通过状态集合以及状态的随机演化描述的。基于实际状态上的(常规)MDPs 理论已经比较成熟,它有一系列的基本概念,并开发了许多有效算法,这些深刻、系统的理论不仅加深了人们对人工智能的理解,而且应用到实践,也获得了很大的成功。然而,应用常规 MDPs 的表达模式,在实际问题中一直存在着一个主要问题:在真实世界里,几乎大部分领域,它们的状态空间是非常巨大的,这引起常规理论中要清晰枚举状态的算法变得难以驾驭,甚至实际不可行。

许多学者从不同的方向研究怎样处理大型状态空间 MDPs 问题,其中最重要的而且能够解决问题的方向,应该是把常规 MDPs 提升到抽象层次上,利用抽象的概括能力和逻辑演算的表达能力,把"指数级"规模的状态空间缩减到"多项式"规模的抽象空间或较小缩减了的实际状态空间。这个方向也符合人类思维特征,人们总是喜欢首先在大的原则上探讨问题的解决方案,然后才付诸实践,具体实现之。

本文重点考察抽象层次上马尔可夫决策过程。所谓抽象马尔可夫决策过程指的是它所描述的状态,并不是实际具体的,而是包含变量,往往由谓词模式描述。我们把抽象层次分为结构型和概括型两类。前者主要立足于问题表达结构或者问题本身固有结构,在其上引进变量考虑状态的抽象描述。后者主要立足于抽象的概括功能,尽可能在抽象层面上讨论常规 MDPs 中一切理论。引入抽象状态和抽象状态空间后,新类型 MDPs 中也就出现了与传统 MDPs 不同的研究课题。本文也重点介绍其中一个课题:把 Bellman 方程推广到关系领域。它具有重大意义,在新类型 MDPs 的理论研究和实际运用方面都起到范例作用。

本文组织如下:第 2 节讨论结构型抽象,第 3 节讨论概括型抽象,讨论把 Bellman 方程推广到关系领域算法,最后是总结和展望。

2 结构型抽象马尔可夫决策过程

许多随机系统都具有显著的结构特征,在抽象表达层次上充分利用这些结构特征,对于解决大型 MDPs 是有效的。

*) 本课题得到国家自然科学基金会重大研究计划项目(90412014)和计算机软件新技术开放课题(A2007 07)资助。**王蓁蓁** 博士研究生,主要研究领域为马尔可夫决策过程、增强学习、群智能;**邢汉承** 教授,博士生导师,主要研究领域为人工智能、机器学习、模式识别;**张志政** 主要研究领域为知识表示和推理;**倪庆剑** 主要研究领域为群智能。

大体上，可以利用的结构分为两类：一是基于问题表述所适宜的逻辑体系的结构，另一是基于问题本身固有的结构。前者以情景演算为代表，后者以因子化 MDPs 为代表，下面分别叙述之。

2.1 在情景演算结构上的符号动态编程

本段材料取自于文献[1]。情景演算是公理动态系统(axiomatizing dynamic worlds)的一阶逻辑语言，其基本成分包括动作、情景、流和公理化领域理论等概念，它们都用一阶逻辑术语表达。动作是由某个具有一定目数的函数符号所表达的项。情景也是一个项，它表示一个动作序列，并用两目函数符号 $do(a,s)$ 表示增加动作 a 到动作序列(即情景) s 后所产生的动作序列，即新的情景项 $s'=do(a,s)$。流是一个关系，用谓词符号表示，它的真值随着情景的变化而变化。至于公理化领域理论，最主要是动作先决条件公理和后继状态公理：

• 动作先决条件公理：对于每一个动作函数 $A(\vec{x})$，都存在一个公理，其句法形式为 $Poss(A(\vec{x}),s)\equiv\Pi_A(\vec{x},s)$，这里 $\Pi_A(\vec{x},s)$ 是一阶逻辑公式，其中自由变量包含在 $\vec{x},s$ 中，它指明动作 A 的先决条件。

• 后继状态公理：对于每一个流 $F(\vec{x},s)$ 都有一个公理，其句法形式为 $F(\vec{x},do(a,s))\equiv\Phi_F(\vec{x},a,s)$，这里 $\Phi_F(\vec{x},a,s)$ 是一阶逻辑公式，其中自由变量包含在 $\vec{x},a,s$ 中，它表明流 F 下一个情景 $do(a,s)$ 下的真值仅仅依赖于流 F 在目前情景 s 下的真值和现实采取的动作 a，此即马尔可夫性。

另外，回归(Regression)是很重要的情景演算。一个公式 Ψ 通过一个动作 a 的回归是一个公式 Ψ'，Ψ' 在 a 被执行之前成立当且仅当在执行 a 后的 Ψ 成立。后继状态公理支持回归运算，如果流 F 的后继状态公理为 $F(\vec{x},do(a,s))\equiv\Phi_F(\vec{x},a,s)$，则 $Regr(F(\vec{x},do(a,s)))=\Phi_F(\vec{x},a,s)$。可以递归定义一个公式的回归运算，该公式中的情景项全都有 $do(a,s)$ 形式，例如 $Regr((\exists x)\Psi)=(\exists x)Regr(\Psi)$。

传统情景演算中的所有动作都是确定性的。因此，为了采用情景演算逻辑结构表达 MDPs，就必须采用某种技巧。例如以一种自然范式把随机动作分解成确定性的基本动作状况，继而运用这些确定性选择形式化情景演算领域公理，从而把抽象 MDPs 构建在情景演算结构上。这里所说的自然方式是指当 agent 执行一个随机动作时，agent 以某一指定的概率，选择一个确定性动作执行。例如随机动作 $A(\vec{x})$ 可能的选择结果为 $n_1(\vec{x}),\cdots,n_k(\vec{x})$，它们都是确定性动作，其形式表达为 $choice(A(\vec{x}),a)\equiv a=n_1(\vec{x})\vee\cdots\vee a=n_k(\vec{x})$。对于 $A(\vec{x})$ 的每一个自然选择 $n(\vec{x})$，用记号 $Prob(n(\vec{x}),A(\vec{x}),s)$ 表示 agent 在情景 s 时执行随机动作 $A(\vec{x})$ 时选择 $n(\vec{x})$ 实际执行时的概率。相应地，公理化领域理论也必须有所改变。这时对每一个确定性选择动作都有一个先决条件公理。而且，$A(\vec{x})$ 的自然选择 $n_j(\vec{x})$ 之间不必具有共同的先决条件，虽然它们经常如此。同样后继状态公理也要改变。

2.1.1 运用状况符号表达一阶 MDP(FOMDP)理论

(1)状态公式及状态空间的分割：一个状态公式 $\Psi(\vec{x},s)$，其中自由变量是非情景变量 $\vec{x}$ 和一个情景变量 s，它表示情景 s 的属性。一组状态公式集合 $\{\Psi_i(\vec{x},s)\}$ 分割整个状态空间是指当且仅当 $\vdash(\forall\vec{x},s)\Psi_i(\vec{x},s)\supset\neg\Psi_j(\vec{x},s)$，对 $\forall i,j\neq i$，以及 $\vdash(\forall\vec{x},s)\vee_i\Psi_i(\vec{x},s)$ 同时成立。

(2) 状况符号表达式及其“代数”运算：运用状况符号 $t=case[\phi_1,t_1;\cdots;\phi_n,t_n]$ 作为下面表达式的简写：$\vee_{i\leqslant n}\{\phi_i\wedge t=t_i\}$，其中 ϕ_i 是状态公式，而 t_i 是项。通常情况下，t_i 是常数，而诸 ϕ_i 分割整个状态空间。有时也把上述状况简写成 $case[\phi_i,t_i]$。在状况表达式上可以引入$\otimes$，$\oplus$，$\odot$和$\cup$等运算。现用“$\circledast$”代表$\otimes$，$\oplus$，$\odot$之一，则

$$case[\phi_i,t_i:i\leqslant n]\circledast case[\Psi_j,\upsilon_j:j\leqslant m]=case[\phi_i\wedge\Psi_j,t_i*\upsilon_j:i\leqslant n,j\leqslant m]$$

例如，如果$\circledast=\otimes$，则$*=\times$，即右边状况中，$t_i*\upsilon_j=t_i\times\upsilon_j$。

$$case[\phi_i,t_i:i\leqslant n]\cup case[\Psi_j,\upsilon_j:j\leqslant m]=case[\phi_1,t_1;\cdots;\phi_n,t_n;\Psi_1,\upsilon_1;\cdots;\Psi_m,\upsilon_m]$$

(3)用状况符号表达 FOMDPs 中的概率、回报函数和值函数

①概率的状况表达式：设随机动作 $A(\vec{x})$ 的可能选择结果为 $n_1(\vec{x}),\cdots,n_k(\vec{x})$，现假定有关涉及这些结果的概率能用状况符号表达。特别地，对 $n_j(\vec{x})$ 的选择概率给出如下：

$$Prob(n_j(\vec{x}),A(\vec{x}),s)=case[\phi_1^j(\vec{x},s),p_1^j;\cdots;\phi_n^j(\vec{x},s),p_n^j]$$

其中 $\phi_i^j,i=1,2,\cdots,n$ 分割状态空间，而 p_i^j 为 agent 执行随机动作 $A(\vec{x})$ 选择 $n_j(\vec{x})$(关于在状态 $\phi_i^j(\vec{x},s)$ 条件下)实现的概率。

②回报函数和值函数：假设回报函数 $R(s)$ 也可以运用状况符号表达：$R(s)=case[\xi_1(s),r_1;\cdots;\xi_m(s),r_m]$，其中 $\xi_i(s)$ 分割状态空间，它们分别为关系流，r_i 为常数。同样，MDP 理论中的(状态)值函数，也可以运用状况符号表达如下：$V(s)=case[\beta_1(s),\upsilon_1;\cdots;\beta_n(s),\upsilon_n]$，其中 $\beta_i(s)$ 为关系流，它们分割状态空间，υ_i 为常数。

总结 正由于上述表达，人们才把 MDP 看成是定义在抽象层次上。例如我们不在一个个基本状态上枚举值函数，而是根据条件 β_i 去区分状态。凡是满足 β_i 的实际状态归入一类，并指定相同的值 υ_i。β_i 可看作为抽象状态，同样也能以这种方式表达 MDP 理论中的 Q 函数和策略。这种利用若干一阶公式分割状态空间的方法，可以避免大型实际状态或实际状态-动作组的枚举表达问题。

2.1.2 动态编程

常规 MDP 的最优策略可以通过值迭代算法得出，同样对于 FOMDP 也是如此。现在构造一阶值迭代算法，但首先要把常规 MDPs 中的决策理论回归(decision-theoretic regression)概念推广到抽象 MDPs，即 FOMDPs 中。

(1) 一阶决策理论回归：假设已知值函数 V，则通过动作类型 $A(\vec{x})$，值函数 V 的一阶决策理论回归(FODTR)是一个关于 $Q^V(A(\vec{x}),s)$ 的逻辑描述。

现在设 $A(\vec{x})$ 是一个随机动作，相应的自然选择为 $n_j(\vec{x}),j\leqslant k$。如果暂时忽略动作先决条件，$Q^V(A(\vec{x}),s)$(按照传统)定义为

$$Q^V(A(\vec{x}),s)=R(s)+\gamma\cdot\{\sum_{t\in s}\Pr(s,A(\vec{x}),t)\cdot V(t)\}$$

其中 $0<\gamma<1$ 为折扣因子。因为不同的后继状态仅仅通过不同的自然选择出现，于是上式可写为：

$$Q^V(A(\vec{x}),s)=R(s)+\gamma\cdot\sum_j Prob(n_j(\vec{x}),A(\vec{x}),s)\cdot V(do(n_j(\vec{x}),s)) \quad (1)$$

正如前述，函数 $R(s)$，$Prob(n,A,s)$ 和 $V(s)$ 全都可用状况符号叙述，它们分别用 $rCase(s)$，$pCase(n,s)$ 和 $vCase(s)$ 表示。现把这些表达式替换为式(1)中相应表达式，并利用回归 Regr 运算处理 $do(n_j(\vec{x}),s)$(即把它改写成仅指称当前情景 s

的形式)以及运用状况符号的代数运算,最后简化式(1)成为一个状况表达式,比如说是 $case[\alpha_i(\vec{x},s),q_i]$,它描述 Q^V 为动作 $A(\vec{x})$ 和 V 值的函数。这实际上是一个转换,可以证明,如果 V 的状况表达式分割了状态空间,则所得的 Q^V 表达式中诸 α_i 也分割了状态,这是本结构最关键的地方。

回忆上述推导曾经忽略了先决条件。为了处理先决条件,$Q^V(A(\vec{x}),s)$ 不再能够处理为函数,它必须用关系式 $Q^V(A(\vec{x}),q,s)$ 来表达,其意义是动作 A 的 Q^V 值在 s 处是 q。这个关系成立,仅当对于 A 的选择,至少有一个比如说是 $n_i(\vec{x})$,$Poss(n_i(\vec{x}),s)$ 成立,否则 Q^V 值无定义,即

$$Q^V(A(\vec{x}),q,s)=[\vee_i Poss(n_i(\vec{x}),s)]\wedge q$$
$$=case[\alpha_i(\vec{x},s),q_i]$$

因为 $\vee_i Poss(n_i(\vec{x}),s)$ 能够分配到状况表达式中(靠 α_i 与它的合取),所以上式右边可以简化为单一的状况表达式。

(2)符号动态编程(Symbolic Dynamic Programming):可以通过设置 $V^0=R$ 以及重复运用下面方程(2)和(3),直到适合的终止条件成立,来进行值迭代(常规 MDPs 的值迭代方法):

$$Q^n(a,s)=R(s)+\{\gamma\sum_{t\in S}\Pr(s,a,t)\cdot V^{n-1}(t)\} \quad (2)$$

$$V(s)=\max_a Q^n(a,s) \quad (3)$$

前面已经讨论用一阶决策理论回归可以递归处理每一个 $Q^n(a,s)$ 为状况表达式,只要 V^{n-1} 是状况表达式。因为 $V^0=R$ 是状况表达式,所以仅仅需要处理方程(3)。假设迭代已经计算到了第 n 步,即对每一个动作类型 A,已经得到关系 $Q^n(A(\vec{x}),q,s)$,下面省略了上标 n,令 $V(s)$ 为第 n 步要求的值函数,则方程(3)可写为

$$V(s)=v\equiv(\exists a)Q(a,v,s)\wedge(\forall b)Q(b,w,s)\supset w\leqslant v \quad (4)$$

如果某个随机动作(例如 no-op)能够在每个情景下执行,则 $V(s)$ 可以看作函数(否则,也很容易定义它为一个关系)。在很一般的假设下,通过在方程(4)右边实施一系列逻辑演算和状况表达式的代数运算,最后可以简化 V 值为优雅形式:

$$V(s)=case\begin{bmatrix}\gamma_1(s) & V_1\\ \gamma_2(s)\wedge\neg\gamma_1(s) & V_2\\ \vdots & \\ \gamma_k(s)\wedge\neg\gamma_1(s)\wedge\cdots\wedge\neg\gamma_{k-1}(s) & V_k\end{bmatrix}$$

其中 $\gamma_i(s)$ 都是关系流。终止条件可仿效传统中通常方法设置。

最后从最优值函数可以推出一组 Q 函数,于是直接从 Q 函数抽取最大值的动作便得到最优策略,该最优策略也是以状况表达式描述的。

2.2 因子化 MDPs 有效解算法

本节讨论一种具有特殊结构的马尔可夫决策过程,即问题领域的转移概率、回报函数乃至于值函数都可以分解为依赖若干变量的"函数"的组合。而状态空间由所有变量取值决定,该空间随着变量的数目增加呈指数级增长。具有该结构的 MDP,我们称之为因子化 MDP。本节材料取自于文献[2]。下面用大写字母表示随机变量,小写字母表示具体数值,粗体大写字母表示向量,小写粗体表示向量的值。

2.2.1 因子化 MDPs

在一个因子化 MDP 中,状态集合是通过一组变量 $\boldsymbol{X}=\{X_1,X_2,\cdots,X_n\}$ 来描述的,其中每一个 X_i 在某个有限领域 $Dom(X_i)$ 取值。一个状态 $\boldsymbol{x}$ 是指对每一个变量 X_i 都规定一个值 $x_i\in Dom(X_i)$,对于任一实例 $\boldsymbol{y}\in Dom(\boldsymbol{Y})$ 和 $\boldsymbol{Y}$ 的子集 $\boldsymbol{Z}\subset\boldsymbol{Y}$,用记号 $\boldsymbol{y}[\boldsymbol{Z}]$ 表示变量 $\boldsymbol{Z}$ 在实例 y 中的值。

在一个因子化 MDP 中,对每一个动作 a,运用动态贝叶斯网络(dynamic Bayesian network,DBN)描述状态转移模型 τ_a。令 X_i 表示变量 X_i 的当前状况,X_i' 表示该变量在下一步的状况,于是一个 DBN 的转移图是一个两层有向无环路图 G_a,它的结点是 $\{X_1,\cdots,X_n,X_1',\cdots,X_n'\}$。用记号 $Parents_a(X_i')$ 表示 X_i' 的父结点集。为了叙述上简单起见,我们假设 $Parents_a(X_i')\subseteq X$,因此 G_a 中所有的弧都是关于在相继时间片的变量之间的。每一个结点 X_i' 都联系一个条件概率分布 CPD,$P_a(X_i'\mid Parents_a(X_i'))$。现在定义转移概率为:

$$P_a(\boldsymbol{x}'\mid\boldsymbol{x})=\prod_i P_a(x_i'\mid u_i)$$

其中 $\boldsymbol{u}_i$ 是 $Parents_a(X'_i)$ 中变量在 $\boldsymbol{x}$ 中的值。

为了完全确定一个 MDP,我们还需要提供回报函数的一个简洁表达。为此,首先定义局部函数(localized functions)概念。

定义 1 如果函数 $f:Dom(\boldsymbol{C})\mapsto\mathcal{R}$,$\boldsymbol{C}\subseteq\boldsymbol{X}$,则称 $\boldsymbol{C}$ 为 f 的 Scope(范围),并记之为 $Scope[f]=\boldsymbol{C}$。

如果 f 的 $Scope$ 为 $\boldsymbol{Y}$ 且 $\boldsymbol{Y}\subset\boldsymbol{Z}$,我们运用记号 $f(\boldsymbol{z})$ 作为 $f(\boldsymbol{y})$ 的缩写,其中 $\boldsymbol{y}$ 是实例 $\boldsymbol{z}$ 的一个部分,该部分相应于 $\boldsymbol{Y}$ 中的变量。

现在假定回报函数是一组局部回报函数的和,每一局部回报函数仅仅依赖于少数几个变量。设 $R_1^a,\cdots,R_r^a$ 是一组函数,每个 R_i^a 的 Scope 局限于变量聚类 $\boldsymbol{U}_i^a\subset\{X_1,\cdots,X_n\}$。然后定义 agent 在状态 $\boldsymbol{x}$ 采取动作 a 后所收到的回报为 $R^a(x)=\sum_{i=1}^r R_i^a(\boldsymbol{u}_i^a)\in\mathcal{R}$,其中 $\boldsymbol{u}_i^a$ 是 $\boldsymbol{U}_i^a$ 中变量在 $\boldsymbol{x}$ 中的值。

事实上,我们已经定义因子化 MDP 为一个 4 元组 $(\boldsymbol{X},A,R,P)$,其中 $\boldsymbol{X}$ 是一个由 $|\boldsymbol{X}|=N$ 个状态组成的有限集合,其中每个状态是通过对变量集合进行"值"指派来描述。注意,这里又用变量集合 $\boldsymbol{X}$ 作为状态空间的符号,这通过上下文不会引起混淆。A 是 agent 能够采取的有限个动作的集合。对于每一个动作因为变量的转移通常都是仅仅依赖于少数几个变量,所以状态转移模型 $\tau_a=\langle G_a,P_a\rangle$ 用 DBN 来描述,并且把即时回报函数 $R:\boldsymbol{X}\times A\to\mathcal{R}$ 分解为依赖于个体变量或小规模变量聚量的局部回报函数之和。这样,用这种语言可以表达指数级大型 MDP 结构。下面的讨论都是在这个因子化 MDP 上。

2.2.2 因子化值函数

在 MDPs 理论中,寻求最优策略有三种最普通的传统方法:值迭代、策略迭代和线性编程。无论哪种方法,关键是计算值函数。回忆值函数是定义在整个状态空间上,现在状态总数为 N,故可把它看作是 N 维向量。当 N 很大时,精确表示值函数,往往很困难,但对于因子化 MDP,适宜用若干个基函数的线性组合近似表示值函数。1999 年,Koller 和 Parr 建议,采用和因子化 MDP 结构相容的特殊类型的基函数,即用一些仅仅牵涉很少几个变量的基函数,它们的线性组合也可能是值函数的很好近似。因为在很多情况下,值函数更接近有结构形态。

定义 2 设 $\boldsymbol{H}=\{h_1,h_2,\cdots,h_k\}$ 是一组基函数,每一个 h_i 的 Scope 都局限于变量的某个子集 $\boldsymbol{C}_i$。如果值函数 V 能够写为下述形式:$V(\boldsymbol{x})=\sum_{j=1}^k w_j h_j(\boldsymbol{x})$,其中 $\boldsymbol{x}$ 是状态,$\boldsymbol{w}=(w_1,w_2,\cdots,w_k)'$ 是系数,则称 V 为关于基函数 H 的因子化线性值函数,简称因子化值函数。

定义 $\mathcal{H}$ 是 $\mathcal{R}^N$ 中由基函数 H 扩展的线性子空间,并且规

定它是我们允许的值函数空间。如果定义 $N\times K$ 阶矩阵 $\boldsymbol{H}$，它的列由 k 个基函数组成，每个基函数看作一个 N 维向量，于是每个允许的(近似)值函数可以表达为 $\boldsymbol{Hw}$，其中 $\boldsymbol{w}$ 是某个系数向量。

在传统迭代过程中，如果产生的值函数 V 超出了空间 $\mathcal{H}$，则应在 $\mathcal{H}$ 中找到某个与 V 最接近的策略取而代之。为此，需要引入：

定义 3 投影 Π 是一个映射 $\Pi:\mathbb{R}^N\to\mathcal{H}$。如果 $\Pi V=H\boldsymbol{w}^*$，其中 $\boldsymbol{w}^*\in\arg\min_{\boldsymbol{w}}\|\boldsymbol{Hw}-V\|$，则称 Π 是关于范 $\|\cdot\|$ 的投影。

至于定义中的范 $\|\cdot\|$，文献[2]运用的是最大范 max-norm $\mathcal{L}_\infty$。这样做的优点是：相当于 $\mathcal{L}_\infty$ 范之投影运算能够与近似运算引起的误差分析紧密联系。

因子化值函数是在指数级尺寸状态空间上进行有效计算的关键。关于它，有两类重要运算：一步前向运算(lookahead)或者称为反向投影(backprojection)和线性编程 LP(linear program)指数级约束简便表示。

(1) 一步前向运算：令 $g_i^a(\boldsymbol{x})=\sum_{\boldsymbol{x}'}P(\boldsymbol{x}'\mid\boldsymbol{x},a)h_i(\boldsymbol{x}')$，称 g_i^a 为基函数 h_i 通过转移模型 P_a 的反向投影，简记为 $g_i^a=P_ah_i$。注意到 P_a 是因子化的(用 DBN 表达)，现在 h_i 的 Scope 也局限于较少变量上，因此 g_i^a 能非常有效地计算出来。事实上，如果 h 的 Scope 局限于 $\boldsymbol{Y}$，则它的方向传播 g 的 Scope 将局限于 $\boldsymbol{Y}'$ 的父辈，即 g 也可以是因子化的。

(2)表达指数级约束：在求最大范投影运算中，一般地我们往往要解出形如下式方程：$\boldsymbol{w}^*\in\arg\min_{\boldsymbol{w}}\|\boldsymbol{Cw}-\boldsymbol{b}\|_\infty$。1960 年 Stiefel 提出的算法，可以通过下述线性编程 LP 求解：

Variables: $w_1,w_2,\cdots,w_k,\phi$;

Minimize: ϕ

Subject to $\phi\geqslant\sum_{j=1}^k w_jc_j(\boldsymbol{x})-b(\boldsymbol{x})$ and $\phi\geqslant b(\boldsymbol{x})-\sum_{j=1}^k w_jc_j(\boldsymbol{x})$

在上述线性编程中，约束实质上：对 $\forall i,\phi\geqslant|\sum_{j=1}^k c_{ij}w_j-b_i|$，或者等价地，即 $\phi\geqslant\|\boldsymbol{Cw}-\boldsymbol{b}\|_\infty$，而目标是最小化 ϕ。如果它的解为 $(\boldsymbol{w}^*,\phi^*)$，则 ϕ^* 便是 $\mathcal{L}_\infty$ 的投影误差。

在 LP 中，变量虽然只有 $k+1$ 个，但约束为 $2N$ 个。如果 LP 约束中每个函数，其 Scope 都局限到少数几个变量上，文献[2]提供了一个算法，它通过引进新变量和新约束，用一种类似变量消去方法来求解上述 LP 方程，该算法称之为因子化 LP 技术。当把它运用于因子化 MDP 中，主要是基于值函数的因子化表达的。文献[2]表明，随着问题尺寸增加，因子化 LP 技术引进的变量和约束数量级为 $O(n^2)$，而用清晰枚举状态方法其数量级为 $O(n2^n)$。

2.2.3 近似策略迭代

运用最大范投影的近似策略迭代算法比较复杂，它需要一个额外假设：每一个动作仅仅影响少数几个状态变量。

①预设动作模型和预设回报模型：在许多情况下，变量有它预设演化模型。因子化 MDPs 的预设转移模型为 $\tau_d=\langle G_d,P_d\rangle$。对于每一个动作 a，定义 $Effects[a]\subseteq\boldsymbol{X}_i'$ 为下一个状态的变量集合，它们的局部转移模型不同于 τ_d，即这样的变量 X_i'，使得 $P_a(X_i'\mid Parents_a(X_i'))\neq P_d(X_i'\mid Parents_a(X_i'))$。

在动态转移中，我们也能定义预设回报模型，即存在一组回报函数，使得 $\sum_{i=1}^r R_i(\boldsymbol{U}_i)$ 与预设动作 d 相连。每一个动作 a 可能有一个额外的回报函数 $R^a(\boldsymbol{U}^a)$，其 *Scope* 局限于 $Rewards[a]=\boldsymbol{U}^a\subset\{X_1,\cdots,X_n\}$。于是，与 a 相联的整个回报由 $R^a+\sum_{i=1}^r R_i$ 表示。这里 R^a 也能够因子化作为更小(即依赖更少变量)的函数项的线性组合，其中每一个有更简洁的表达。现在在此基础上构建完整的算法。

②计算贪婪策略：在策略改良步骤计算值函数 $V^{(t-1)}$ 的 greedy 策略：$\pi^{(t)}=Greedy(V^{(t-1)})$。在算法里，值函数的估计为 $\boldsymbol{Hw}$。因此，greedy 策略给定如下：$Greedy(\boldsymbol{Hw})(\boldsymbol{x})=\arg\max_a Q_a(x)$，其中

$$\begin{aligned}Q_a(\boldsymbol{x})&=R(\boldsymbol{x},a)+\gamma\sum_{\boldsymbol{x}'}P(\boldsymbol{x}'\mid\boldsymbol{x},a)V(\boldsymbol{x}')\\&=R(\boldsymbol{x},a)+\gamma\sum_{\boldsymbol{x}'}P(\boldsymbol{x}'\mid\boldsymbol{x},a)\sum_i w_ih_i(\boldsymbol{x}')\\&=R(\boldsymbol{x},a)+\gamma\sum_i w_i\sum_{\boldsymbol{x}'}P(\boldsymbol{x}'\mid\boldsymbol{x},a)h_i(\boldsymbol{x}')\\&=R(\boldsymbol{x},a)+\gamma\sum_i w_ig_i^a(\boldsymbol{x})\\&=R^a(\boldsymbol{x})+\sum_{i=1}^r R_i(\boldsymbol{x})+\gamma\sum_i w_ig_i^a(\boldsymbol{x})\end{aligned}$$

对于预设动作 d，其 $Q_d(\boldsymbol{x})=\sum_{i=1}^r R(\boldsymbol{x})+\gamma\sum_i w_ig_i^d(\boldsymbol{x})$。

实际上，在加权组合中，大部分成分是相等的，即 $g_i^a=g_i^d$。因为 g_i^a 相应于基函数 $h_i(\boldsymbol{C}_i)$ 的反向投影，它仅仅在动作 a 影响 $\boldsymbol{C}_i$ 中的变量时才与 g_i^d 不同。形式地，如果 $Effects[a]\cap C_i=\emptyset$，则 $g_i^a(\boldsymbol{x})=g_i^d(\boldsymbol{x})$，这是由于 $\boldsymbol{C}_i$ 中所有变量在 τ_a 和 τ_d 转移模型中是相同的。换言之，当决定 a 是否比预设动作 d 更好时，Q_a 中这些第 i 个成分是不相关的。定义 $I_a=\{i\mid Effects[a]\cap\boldsymbol{C}_i\neq\emptyset\}$，即当 $i\in I_a$ 时，则 $g_i^a\neq g_i^d$。

考虑动作 a 与预设动作 d 之间差异。令

$$\begin{aligned}\delta_a(\boldsymbol{x})&=Q_a(\boldsymbol{x})-Q_d(\boldsymbol{x})\\&=R^a(\boldsymbol{x})+\sum_{i\in I_a}w_i[g_i^a(\boldsymbol{x})-g_i^d(\boldsymbol{x})]\end{aligned}$$

$\delta_a(\boldsymbol{x})$ 是一个函数，它的范围局限于 $T_a=\boldsymbol{U}^a\cup[\bigcup_{i\in I_a}\Gamma_a(\boldsymbol{C}'_i)]$，其中 $\Gamma_a(\boldsymbol{C}'_i)$ 表示 $\boldsymbol{C}'_i$ 关于动作 a 的父辈集合。

现在给出 $\pi^{(t)}$ 的决策列表形式。对于每一个动作 a，相对于 Q_a 函数，定义一组条件 $\langle t,a,\delta\rangle$，其中 t 是关于变量 T_a 的某个指派，以及 $\delta=\delta_a(t)$，然后对所有动作，依据 δ 值下降次序排列所有条件为

$$\langle t_1,a_1,\delta_1\rangle,\langle t_2,a_2,\delta_2\rangle,\cdots,\langle t_l,a_l,\delta_l\rangle \tag{5}$$

如果 agent 处在状态 $\boldsymbol{x}$ 上，设 t_i 为第一个与 $\boldsymbol{x}$ 相一致的项，则 agent 采取动作 a_i，就有较大的 δ_i 值。

注意到 T_a 中的变量个数很少，所以上述 Greedy 算法避免了传统算法中无法解决的指数级大型状态空间问题。

③计算近似值函数

在近似值函数确定步骤，算法计算

$$\begin{aligned}w^{(t)}&=\arg\min_{\boldsymbol{w}}\|\boldsymbol{Hw}-(R_\pi(t)+\gamma P_\pi(t)\boldsymbol{Hw})\|_\infty\\&=\arg\min_{\boldsymbol{w}}\|(\boldsymbol{H}-\gamma P_\pi(t)\boldsymbol{H}))-R_\pi(t)\|_\infty\end{aligned}$$

如果令 $C=\boldsymbol{H}-\gamma P_\pi(t)\boldsymbol{H}$，$b=R_\pi(t)$，则它是方程 $\boldsymbol{w}^*\in\arg\min\|C\boldsymbol{w}-\boldsymbol{b}\|_\infty$ 的实例。

如果 $P_\pi(t)$ 因子化，则 C 矩阵里每一列也相应是 Scope 有限的函数。又目标函数 $R_\pi(t)$(回报函数)也是因子化的，于是能应用前述因子化 LP 方法去估计策略 $\pi^{(t)}$ 的值函数。然而用决策列表表达策略 $\pi^{(t)}$，一般地导致转移模型 $P_\pi(t)$ 不能用简洁的 DBN 表达。因此，为了 $P_\pi(t)$ 因子化，我们必须分别考虑每一分支，相应每一分支运用 DBN 技术和因子化 LP 构架(construction)。设 $\pi^{(t)}$ 已经表达为式(5)中形式，令 $S_i=\left\{\boldsymbol{x}\,\middle|\,\begin{array}{l}\boldsymbol{x}\text{ 是状态，}\boldsymbol{x}\text{ 与 }t_i\text{ 一致，}\\\text{但对 }\forall j<i,\boldsymbol{x}\text{ 与 }t_j\text{ 不一致}\end{array}\right\}$。如同上面叙述，LP 定义了一组约束，例如 $\phi\geqslant\sum_j w_jC_j(\boldsymbol{x})-b(\boldsymbol{x})$，$\forall\boldsymbol{x}\in S_i$。因为对

于 S_i 中每一状态，agent 将采取动作 a_i，因此这里可以运用 P_{a_i}（由前述可知它是因子化转移模型）去代替非因子化的 $P_\pi(t)$。类似地，对于 S_i 状态子集，回报函数成为 $R^{a_i}(\boldsymbol{x})+\sum_{i=1}^{r} R_i(\boldsymbol{x})$。从而，相应于第 i 个分支，线性约束具有如下形式：

$$\phi \geqslant R(\boldsymbol{x},a_i)+\sum_l w_l(\gamma g_l^{a_i}(\boldsymbol{x})-h_l(\boldsymbol{x})+\sum_{j<i}-\infty 1(x=t_j),\quad \forall x\sim[t_i]$$

其中 $\forall x\sim[t_i]$ 定义为 $\boldsymbol{x}$ 的指派与 t_i 一致，特征函数 $1(x=t_j)$ 的引入是为了使得对所有的 $j<i$ 的 S_j 中的状态，将平凡地满足上式。注意特征函数也是 T_j 的其 *Scope* 有限的函数，因此它可以利用前面介绍的因子化 LP 技术容易得到近似值函数的估计。

通过策略评估（即计算值函数）和策略改良（即计算 Greedy 策略），我们就可以迭代直到满足类似传统算法的一些条件终止算法，并计算相应的最优策略。

文献[2]还讨论了另一种近似解算法——近似线性编程算法，该算法不仅优雅而且容易执行。该文除了利用由因子化 MDP 导致的上述所谓附加结构，还利用存在 MDP 本身领域里所谓内容确定（context-specific）结构去开发算法，例如讨论了以规划为基础的因子化 LP 技术。这些都把指数级大型线性编程减少到等价的多项式尺寸的 LP。并用理论分析和实验数据证明了文献所开发出的算法是很有前途的。

2.3 评论

在运用抽象层次上，利用结构是解决大型 MDPs 的有效途径。上面介绍的两种结构具有不同的特点。对于 MDPs，先前基于计算逻辑的框架是很少的。2001 年，Boutilier 等人报道了他们关于结合 MDPs 与 Reiter 的情景演算的工作（文献[1]）。如果问题领域适宜情景演算框架，则利用情景演算结构在抽象层次上进行逻辑和代数演算，能够精确地、优美地表达大型 MDPs 状态空间上的值函数。可惜的是，逻辑演算比较复杂，其中增强学习技术依赖于问题模型。

如果说情景演算框架表达的结构本质上是逻辑表达层次上的结构，那么可以说，因子化 MDPs，其抽象层次上所运用的结构本质上是问题领域上的结构，特别是当我们运用 context-specific 结构时更是如此。正因为如此，抽象层次上的状态表达概括力不强，故文献[2]采用近似求解方法，大大提高了解决大型 MDPs 的能力。虽然如此，这些算法都基于近似值函数的表达，一般把值函数看作为一组基函数的线性组合，并把允许的值函数偏置空间定义为如此线性组合所得到的空间。这样基函数的选择就很重要，即便它是依赖于问题领域，但更好选择它，对于理论和实践都是关键，例如 Koller 和 Parr 于 1999 年建议，采用和因子化 MDP 结构相容的特别类型的基函数，即用 Scope 特别有限的函数作为基函数。

3 概括型抽象马尔可夫决策过程

2003 年，Kristian Kersting 和 Luc De Raedt 首先提出逻辑马尔可夫决策过程，简写成 LOMDPs，该过程把马尔可夫决策过程（MDPs）与逻辑编程结合起来。2004 年，Martijn Van Otterlo 提出关系因子化马尔可夫决策过程，简写为 RMDPs，该过程在抽象状态-动作空间上建立抽象关系模式 CARCASS，它作为一个表达工具，构成了 RMDPs 的骨架。同年，LOMDPs 和 RMDPs 提出者，合作共同开发了一个算法，即关系 Bellman 改良操作 REBEL（relational Bellman update operator），把 Bellman 方程推广到关系领域。本节材料主要取自于文献[3-6]但部分概念和符号与原文有所不同。这些轻微变动仅仅是为了使本节内容统一连贯，便于阅读。本节考虑的实例是积木世界，除非特别声明，用小写字母表示具体的积木或实际状态，大写字母表示抽象对象或变量。

3.1 马尔可夫决策编程（Markov Decision Programs）

3.1.1 一阶字母表 Σ 和有关的逻辑概念

用 Σ 表示一个一阶字母表，它是由变量、若干关系符号 r（具有目 $m\geqslant 0$）和若干函数符号 f（具有目 $n\geqslant 0$）组成的集合。当 $n=0$ 时，函数符号 f 被称为常数。如果 $m=0$，则关系符号 r 被称为命题。形式 $r(t_1,t_2,\cdots,t_m)$ 称之为原子，其中 r 为关系符号，$t_i(i=1,2,\cdots,m)$ 为项。所谓项，它或者是一个变量，或者形如 $f(t_1,t_2,\cdots,t_n)$，其中 f 是函数符号。当 $n=0$ 时，f 是常数；当 $n\neq 0$ 时，$t_i(i=1,2,\cdots,n)$ 是项。一般地，我们只讨论 $n=0$ 时的函数符号，即今后的项或者是变量或者是常数。至于关系符号 r，可以分为两类：谓词模板和动作模板，由它们分别构造抽象状态和抽象动作。

原子的集合称之为合取，合取被假定是存在量化的。一个置换 $\theta=\{X_1/t_1,\cdots,X_n/t_n\}$ 是一个指派，即把项 t_i 指派给变量 $X_i,i=1,2,\cdots,n$。两个合取 A,B 之间，如果存在一个置换 θ，使得 $B\theta\subset A$ 成立，则称 A 是关于 θ 概括（或包容）于 B，记之为 $A\leqslant_\theta B$。两个合取项 A,B 的最大下界（greatest lower bound）glb 是一个被 A,B 所包容的最一般的合取项。即如果 $C\leqslant_\theta A$，$C\leqslant_\theta B$，并且对于任意一个满足下式的 D 而言：$D\leqslant_\theta A$，$D\leqslant_{\theta'} B$，皆存在一个置换 θ'，使得：$\theta\theta''=\theta'$，且 $D\leqslant_{\theta'} C$，则称 C 为 A,B 的最大下界 glb，写为 $\mathrm{glb}(A,B)=C$。注意，包容概念和 glb 概念也可以在子句上定义。

一个项、一个原子或一个语句 E 称之为基本的，如果它不包含变量。我们用符号 $mgu(a,b)$ 表示两原子 a,b 之间最一般的合一。用符号 hb_Σ（或 hb^Σ）表示由 Σ 的 Herbrand 基，它是由 Σ 中的关系符号和常数构造的所有基本原子的集合。

3.1.2 抽象状态

一个抽象状态是合取 $\mathbb{Z}$，它由若干谓词模板所构成的若干逻辑原子所组成。实质上，抽象状态的语义是它为基本状态的一个集合。用积木世界为例，一个抽象状态 $\mathbb{Z}$，比如 $on(X,Y),bl(Y),bl(X),cl(X)$，它表达所有（在给定的字母表 Σ 上）如此实际状态的集合：该状态有两个积木块，一块在另一块之上。例如，基本状态 $Z,on(a,b),on(b,fl),bl(a),bl(b),cl(a)$，它就包含在抽象状态 $\mathbb{Z}$ 所代表的基本状态集合中。更精确地说，一个抽象状态 $\mathbb{Z}$ 表达所有状态 Z。对于 Z，存在一个置换 θ，使得 $\mathbb{Z}\theta\subseteq Z$ 成立。上面例子中，所需的置换 $\theta=\{X/a,Y/b\}$。今后我们用 $S(\mathbb{Z})$ 表明抽象状态所表达的基本状态所组成的集合。由所有抽象状态组成的空间称之为抽象状态空间。

3.1.3 抽象动作和抽象转移

一个抽象动作是一个由有限个动作规则组成的集合：$\mathbb{H}_i \xleftarrow{p_i:\mathbb{A}} \mathbb{B}$，$i=1,2,\cdots,l$，其中 $\mathbb{A}$ 是一个原子（由动作模板构成），它既是动作的名字，也是该动作的一个组成部分。$\mathbb{B}$ 是一个抽象状态，它指示执行 $\mathbb{A}$ 的先决条件。$\mathbb{H}_i$ 是 $\mathbb{A}$ 的第 i 个可能结果，其概率为 p_i，并要求 $\sum_{i=1}^{l} p_i=1$。有时也把每一个动作规则 $\mathbb{H}_i \xleftarrow{p_i:\mathbb{A}} \mathbb{B}$ 称之为一个抽象转移 T，其中 $act(T)=\mathbb{A}$ 是抽象动作名称，$P(T)=p_i$ 为转移概率，并且定义 $body(T):=\mathbb{B}$ 和 $head(T):=\mathbb{H}_i$，它们分别称为转移 T 的体和头。自然要求 $vars(\mathbb{H}_i)\subseteq vars(\mathbb{B})$，$vars(\mathbb{A})\subseteq vars(\mathbb{B})$，这里例如 $vars(\mathbb{B})$ 表示 $\mathbb{B}$ 中所有变量的集合。当用抽象转移术语时，用

记号$\mathbb{B}$表示所有抽象转移的$\mathbb{B}$所组成的集合，用记号$\mathbb{T}$表示所有抽象转移T所组成的集合，用记号$\mathbb{A}$表示所有抽象动作名称$\mathbb{A}$的集合，并要求下面等式成立：

$$\forall \mathbb{B}\in\mathbb{B},\forall \mathbb{A}\in A(\mathbb{B})\sum_{\substack{T\in\mathbb{T}\\ body(T)=\mathbb{B}\\ act(T)=\mathbb{A}}}P(T)=1 \qquad (6)$$

其中$A(\mathbb{B})$表示所有先决条件为$\mathbb{B}$的抽象动作名称的集合，它与上述关于抽象动作的要求一致。

一个抽象转移$\mathbb{H}_i\xleftarrow{p_i:\mathbb{A}}\mathbb{B}$的语义是：如果 agent 处于状态$Z$，使得$\mathbb{B}\theta\subseteq Z$，当 agent 执行动作$\mathbb{A}\theta$时，则 agent 将以概率$p_i$走到状态$Z':=[Z\backslash\mathbb{B}\theta]\cup\mathbb{H}_i\theta$。

3.1.4 抽象回报

抽象回报模型R指定 agent 进入抽象状态时所获得的回报，在 REBEL 算法里，把它作为初始抽象状态值函数V_0。在 LOMDPs 里，抽象回报与抽象转移一起定义，即把上述一个抽象转移T扩充为形式：$\mathbb{H}\xleftarrow{p:r:\mathbb{A}}\mathbb{B}$，其中$R(T):=r\in\mathcal{R}$表达该转移所获得的回报。虽然这里回报$R$是作为抽象转移的函数，但它与仅作为抽象状态函数的回报R在实质上并无多大区别。除非特别声明，R作为抽象状态函数。

3.2 逻辑马尔可夫决策过程 LOMDPs

3.2.1 定义

一个逻辑马尔可夫决策过程(LOMDP)是一个4元组$\mathbb{M}=(\sum,A,\mathbb{T},R)$，其中$\sum$是一阶逻辑字母表，$A$是一个抽象动作名称集合，$\mathbb{B}$是形如$\mathbb{H}\xleftarrow{p:r:\mathbb{A}}\mathbb{B}$抽象转移$T$的集合，$R$是定义在$\mathbb{T}$上的回报函数，并且上面等式(6)成立。

为了解决抽象转移的矛盾冲突，文献[3]采用类似 Prolog 简单易行方法，首先在$\mathbb{B}$中按动作-体排序，即所有转移建立一个全序关系$\prec$，并按前向搜索在动作-体组中找到第一个匹配的体。

3.2.2 最优策略的求法

文献[3]中规定在$\sum$上的一个抽象策略$\tilde{\pi}$为一些形如$\mathbb{A}\leftarrow\mathbb{L}$决策规则的集合，其中$\mathbb{A}$是一个抽象动作，$\mathbb{L}$是一个抽象状态。抽象决策规则$\mathbb{A}\leftarrow\mathbb{L}$的语义是：如果 agent 处于状态$Z$，使得$\mathbb{L}\theta\subseteq Z$，则 *agent* 将要执行动作$\mathbb{A}\theta$，它用$\tilde{\pi}(Z)$表示。

如果抽象策略$\tilde{\pi}$由多个决策规则组成，则在规则之间建立一个全序$\prec$。令$\mathbb{L}=\{\mathbb{L}_1,\cdots,\mathbb{L}_m\}$是按$\prec$序排列的策略$\tilde{\pi}$的体所组成的集合，称它为策略$\tilde{\pi}$的抽象层次。如果$\mathbb{L}$能覆盖 *LOMDP* 所有可能实际状态，则它能形成实际状态空间的一个分割。并用$[\mathbb{L}_1],\cdots,[\mathbb{L}_m]$表示相应的等价类集合，即$[\mathbb{L}_1]=S(\mathbb{L}_1)$，对于$i=2,\cdots,m$，有$[\mathbb{L}_i]=S(\mathbb{L}_i)\backslash\bigcup_{j=1}^{i-1}[\mathbb{L}_j]$。

文献[3]证明了一系列定理和命题。

定理1 每一个 LOMDP $\mathbb{M}=(\sum,A,\mathbb{T},R)$确定一个实际的 MDP $M(\mathbb{M})=(S,A,\boldsymbol{T},R)$。

以定理1为基础，可得到如下论断：任意抽象策略$\tilde{\pi}$确定一个在基本状态空间上的非确定策略π。因此，当$\mathbb{M}$是 LOMDP，而$M(\mathbb{M})$是由它产生的 MDP，我们就可以定义$\forall\mathbb{L}\in\mathbb{L}$的期望回报为所有在$[\mathbb{L}]$中状态的期望回报，即

$$V_{\tilde{\pi}}(\mathbb{L})=E_{[\mathbb{L}]}[E_\pi\{\sum_{k=1}^{\infty}\lambda^k r_{t+k}\mid Z_t=Z\}] \qquad (7)$$

其中r_i表示在$M(\mathbb{M})$中遵循基本层次策略π(它由$\tilde{\pi}$产生)在时间i所获得的回报，λ为折扣因子，$0<\lambda<1$，中括号里的期望E_π是基本空间上系统从时间t处于状态$Z\in S$(记之为$Z_t=Z$)开始的长期累积折扣期望，中括号外的期望$E_{[\mathbb{L}]}$是关于所有$[\mathbb{L}]$中的元素而求的。

定理2 设$\mathbb{M}=(\sum,A,\mathbb{T},R)$是一个 LOMDP，$\tilde{\pi}$是一个抽象策略，其抽象层次$\mathbb{L}=\{\mathbb{L}_1,\cdots,\mathbb{L}_m\}$，则存在一个有限的 MDP $L=(\{l_1,\cdots,l_m\},A_L,\mathbf{T}_L,R)$和一个关于$L$的策略$\pi$，使得$V_{\tilde{\pi}}(\mathbb{L}_i)=V_\pi(l_i)$，$i=1,2,\cdots,m$。

在定理2的证明过程中，说明了马尔可夫决策过程L的具体构造。于是在原则上，对于在 LOMDP 中抽象策略$\tilde{\pi}$的评估可以转化到在 MDP 中L的策略π的评估，也就是说我们得到了抽象值函数$V_{\tilde{\pi}}$的计算方法，从而在L中最优策略的计算便可以运用到在 LOMDP 中关于抽象层次$\mathbb{L}$上最优策略的计算。因为它把抽象层次的最优策略问题归约到实际状态的最优策略问题，所以我们称这种方法为“转换法”。

然而在实际马尔可夫决策过程L中，其转移概率T_L的计算要牵涉到$M(\mathbb{M})$中的状态空间的分割和概率分布问题，即使计算是可能的，也是很复杂的。因此文献[3]给出一个随机迭代动态编程，并结合了传统的 Q 学习，引入一个(模型自由的)逻辑 Q 学习，简称 LQ 学习，此方法解决上述寻求抽象最优策略的问题(见文献[3]第50页)。

LQ 学习是在固定的抽象层次$\mathbb{L}=\{\mathbb{L}_1,\cdots,\mathbb{L}_m\}$上应用迭代技术寻找抽象$Q$函数，其主要步骤是：

①初始化$Q_0(\mathbb{L},\mathbb{A})$，即对每一个$\mathbb{L}\in\mathbb{L}$，$\mathbb{A}\in A$，给定任意一个值。

②在每一个迭代纪元(episode)，任选一个基本状态Z，1)令$\mathbb{L}\in\mathbb{L}$为唯一匹配Z的抽象状态，按传统 Q 学习方法选择抽象动作$\mathbb{A}$，然后根据$\mathbb{A}$和Z及其匹配的$\mathbb{L}$，在所有可能实施的基本动作中均匀选择一个动作，比如说a。于是在Z上，采取动作a，如果观察到系统进入下一个状态Z'，并获得r。再令$\mathbb{L}'\in\mathbb{L}$为唯一的与Z'匹配的抽象状态。2)令$visits_n(\mathbb{L},\mathbb{A})$为抽象状态-动作对$(\mathbb{L},\mathbb{A})$直到第$n$次迭代(包括第$n$次)，被访问到的次数。设置学习因子$\alpha_n=(1+visits_n(\mathbb{L},\mathbb{A}))^{-1}$，则可学习到$Q_n(\mathbb{L},\mathbb{A})=(1-\alpha_n)Q_{n-1}(\mathbb{L},\mathbb{A})+\alpha_n(r+\lambda\max_{\mathbb{A}'}Q_{n-1}(\mathbb{L}',\mathbb{A}'))$。3)令$Z\leftarrow Z'$，$n\leftarrow n+1$，并重复上述1)、2)两步，直到$Z$为终止状态。

③重复执行第②步(即在纪元上迭代)，直到收敛或$Q_n(\mathbb{L},\mathbb{A})$满足一定的精度，整个迭代过程结束。

在一般(类似常规 MDP 的动态迭代)的条件下，上述迭代过程最终会收敛，即$Q_n(\mathbb{L},\mathbb{A})\to Q(\mathbb{L},\mathbb{A})$，它为最优抽象$Q$函数，运用它可得抽象层次$\mathbb{L}$上的最优抽象策略，即$\forall\mathbb{L}\in\mathbb{L}$，令$\tilde{\pi}(\mathbb{L})=\arg\max_{\mathbb{A}}Q(\mathbb{L},\mathbb{A})$

3.3 关系因子化马尔可夫决策过程 RMDPs

3.3.1 定义

一个关系因子化马尔可夫决策过程(Relationally factored MDP，RMDP)$\mathbb{M}$是一个5元组$\langle P,A,D,T,R\rangle$，其中D是一个(有限)对象领域，P是一个(有限)谓词模板的集合，A是一个(有限)动作模板的集合。状态空间$S=S(\mathbb{M})$是所有在P和D上的一阶解释组成的集合的一个特殊子集。实际动作集合$A=hb^{\langle A\cup D\rangle}$。给定$S$和$A$，转移函数$T$和回报函数$R$按通常马尔可夫决策过程 MDPs 定义，即$\langle S,A,T,R\rangle$构成一个基本 MDP。

与 LOMDPs 不同，Van Otterlo 还引入背景知识 BK 概念。BK 为若干 Horn 子句$\mathbb{H}\leftarrow\mathbb{B}$的集合，其中$\mathbb{B}$是一个合取或者是$\emptyset$，$\mathbb{H}$是原子，它称之为头(Head)，并要求$vars(\mathbb{H})\subseteq vars(\mathbb{B})$。由此，抽象状态$\tilde{s}$便定义为由$P$所构成的原子和 BK 中的 head 上的合取，而抽象动作$\tilde{a}$为由A构成的原子。

并用符号 $S(\tilde{s})$ 表示由抽象状态 $\tilde{s}$ 形成的所有基本状态的一个集合(聚类),即:$S(\tilde{s})=\{s \mid s\in S(\mathbb{M}), s \vdash_\theta \tilde{s}\}$,其中 $\vdash_\theta$ 表示蕴含。例如两个合取 A,B,$A \vdash_\theta B$ 表示合取 A 蕴含 B,如果存在一个置换 θ,并且运用 BK 作为公理从 A 能够证明(即推出)B 成立的话。$\vdash_\theta$ 相当于 LOMDPs 中的$\leqslant_\theta$。

如果抽象状态集合$\{\tilde{s}_1,\tilde{s}_2,\cdots,\tilde{s}_n\}$完全包含了状态空间 $S(\mathbb{M})$,文献[5]也采取在诸 $\tilde{s}_i$ 中排序的方法(不妨设就是上面的序),引入 $S(\mathbb{M})$ 上的一个分割。即定义$[\![\tilde{s}_1]\!]=S(\tilde{s}_1)$,对于 $i=2,\cdots,n$,定义$[\![\tilde{s}_i]\!]=S(s_i)\backslash(\bigcup_{j=1}^{i-1}[\![\tilde{s}_j]\!])$。

代替 LOMDPs 中的抽象转移,RMDP 定义抽象动作规则概念。一个抽象动作规则 $\tilde{t}$ 是形如$\tilde{s}\rightarrow\tilde{a}$,有时也写成$(\tilde{s},\tilde{a})$形式,其中 $\tilde{s}$ 是抽象状态,$\tilde{a}$ 是抽象动作,并且 $vars(\tilde{a})\subseteq vars(\tilde{s})$。规则 $\tilde{t}$ 的语义是,如果 agent 处于某个状态 $s\in[\![\tilde{s}]\!]$,则由所选择的抽象动作 $\tilde{a}$,它将非确定性选择某个动作 $a=\tilde{a}\theta$,其中 $s \vdash_\theta \tilde{s}$。为了简单起见,该文假定选择某个 a 的概率为$\frac{1}{n}$,n 指不同置换 θ(满足 $s \vdash_\theta \tilde{s}$ 关系)的数目。于是由某个抽象动作规则(也称为抽象转移规则)可聚集一些基本状态-动作对,以组成集合$[\![\tilde{s},\tilde{a}]\!]=\{(s,a) \mid s\in[\![\tilde{s}]\!]\wedge s \vdash_\theta \tilde{s}\wedge a=\tilde{a}\theta\}$。由此,$\forall s\in[\![\tilde{s}]\!]$,$[\![\tilde{s},\tilde{a}]\!]^s=\{a \mid (s,a)\in[\![\tilde{s},\tilde{a}]\!]\}$便表示相对于某个抽象转移规则在某个基本状态 s 上可以运用的所有基本动作的集合。这些动作相对于抽象层次上的应用而言,都是"类似"的,我们在抽象意义上不能区分它们。从而,由所有动作规则组成的集合称之为一个策略 $\tilde{\pi}$,自然在规则之间如同在上述抽象状态之间一样,规定一个全序。

最后,设 $M=[P,A,D,T,R]$是一个 RMDP,用记号 $\tilde{C}(\mathbb{M})$表示关于$\mathbb{M}$的一个 CARCASS(Compact Abstraction using Relational Conjunctions for Aggregation of State-action Spaces),它定义为如此一个结构$(\widetilde{sa},\leqslant,\mathrm{BK})$,其中$\widetilde{sa}$是一个集合$\{(\tilde{s},\widetilde{A})\}$,该集合每一个元素$(\tilde{s},\widetilde{A})$是由一个抽象状态 $\tilde{s}$ 连同所有在$\tilde{s}$ 可应用的抽象动作组成的集合 $\widetilde{A}$ 构成的序对,$\leqslant$是一个在抽象状态上的全序,BK 是 Horn 子句组成的集合。并且对所有$(\tilde{s},\widetilde{A})\in\widetilde{sa}$和 $\tilde{a}\in\widetilde{A}$,$var(\tilde{a})\subseteq var(\tilde{s})$。今后用记号 $\tilde{S}(\tilde{C})$表示 $\tilde{C}(\mathbb{M})$中的抽象状态,$\forall\tilde{s}\in\tilde{S}(\tilde{C})$,它为 P 和 BK 中的 *heads* 上的一个合取。显然,CARCASS $\tilde{C}(\mathbb{M})$是 RMDP M 的骨架,它分割状态空间 $S(\mathbb{M})$,并形成状态-动作空间上不同的聚类,从而把"相似"的状态以及"相似"的动作分别聚集在一起,以简约的方式思考问题,特别是背景知识 BK 的应用,更能提高抽象状态的表现能力。

通过上面的叙述,不难看出 LOMDPs 和 RMDPs 在本质上是一样的,尽管它们的出发点有所不同。

3.3.2 最优策略的求法

作为抽象状态-动作对的抽象 $\tilde{Q}$ 函数,主要是应用 CARCASS 工具来学习的。其基本精神也是在抽象状态和实际状态之间通过 CARCASS 进行转换,并运用实际状态上 Q 学习方法来学习 $\tilde{Q}$ 函数(见文献[5]Algorithm 1)。

此算法及上面的 LQ 学习都是模型自由方法,文献[5]还给出了以模型为基础增强学习技术,该技术还运用 PS(Prioritized Sweeping)优先扫描技术,使得传统的 Q 学习方法得到改进。值得注意的是,文献[5]还用实例指出,在抽象层次上,决策问题可能不再具有马尔可夫性。为此,作者还给出聚类 RMDP $\tilde{C}(M)$仍然保留马尔可夫性的先决条件。

3.4 推广 Bellman 方程到关系领域

上面描述的有关最优策略的求法问题,实质上仍然以实际 MDP 中传统算法为基础,并不是"纯粹"地在抽象状态空间上寻找(抽象)值函数和(抽象)动作值函数,即 V 值和 Q 值函数。而由文献[6]开发的算法 REBEL 却不同,它运用约束的逻辑编程语言简洁地表达了关系领域上的马尔可夫决策过程,并在抽象层次上直接进行值迭代,根本不涉及到基础状态空间。下面我们介绍该算法。但首先要把 V 值和 Q 值改写为规则形式,并引入完全约束概念。

• 一个抽象状态值函数 V 是有限个形如 $c\leftarrow\mathbb{B}$ 值规则的列表,其中$\mathbb{B}$是抽象状态,而 $c\in\mathscr{R}$ 为实数。

对于任意抽象状态 Z,值函数 $V(Z)$定义为所有匹配 Z 的值规则 $c\leftarrow\mathbb{M}$ 中最大的值 c,而值规则匹配的意思是指 $Z\leqslant_\theta\mathbb{B}$ 成立。考虑 $V_0(=\mathscr{R})$,例如它为:

$10.0\leftarrow on(a,b)$, $0.0\leftarrow$true

注意,这里是把 $on(a,b)$作为目标状态,并在最后值规则中运用 true,它保证了所有状态都能指定值。

• 一个抽象状态-动作值函数 Q 是有限个形如 $c:\mathbb{A}\leftarrow\mathbb{B}$ 所谓 Q-规则的集合,其中$\mathbb{B}$是一个抽象状态,$\mathbb{A}$是一个抽象动作,而 $c\in\mathscr{R}$。

对于任意的抽象状态-动作"队" $\mathbb{B}$和$\mathbb{A}$,Q 指定包含$\mathbb{A}\leftarrow\mathbb{B}$的所有抽象状态动作规则的最大值 c 为其值。

在增强学习(RL)中,阶段(episodic) 任务将要用一个特殊的吸收状态编码。文献[6]用一个人为确定的动作例如 $on(a,b)\xleftarrow{1.0:absorbing}on(a,b)$来做到这一点,它表明所有状态如果它被 $on(a,b)$所包含,则它仅仅转移到自身,并收取零回报。

• 一个完全约束(integrity constraints)及其解决方法

一个完全约束是由领域知识所强加的。我们用集合 C 表示完全约束的集合,而每一个完全约束是一个 Horn 子句。例如在积木世界里,下面两个 Horn 子句 false$\leftarrow on(X,Y),cl(Y)$以及 $X\neq Y\leftarrow on(X,Y)$分别表明两个完全约束,即如果一个积木的顶部有另一个积木,则它就不是"清空的"和一个积木不可能在自己的上面。

于是一个抽象状态$\mathbb{Z}$的完备(completion)描述是指关于 $C\cup\{Z\}$的最小不动点(least fix-point),即由从 $C\cup\{Z\}$推断出的所有事实。例如 $on(a,b)$并没有指出 a 不同于 b,运用上述规则,这个状态可以完备为 $on(a,b),a\neq b$。特别地,如果"完备"中包含 false,它表明状态不满足约束,因此它是一个不合法的状态。为了处理完全约束问题,我们还必须修改有关动作定义和一般化的概念。例如动作定义现在应该约束,使得它们不能导致不合法状态。文献[6]把完全约束作为背景理论,并且同样可以证明,这里介绍的马尔可夫决策编程可以产生一个(可能无限)MDP。

下面我们介绍关系值迭代 REBEL 算法:对于任意一个马尔可夫决策编程,给定一个抽象回报模型 R,则把它作为初始抽象状态值函数 V_0,然后相继计算下一个抽象状态值函数 $V_t,t=1,2,\cdots$。

其中主要思想是把传统的 Bellman 方程回溯运算提升到抽象层次,即引入关系 Bellman 回溯运算 REBEL。每次迭代包括以下三个步骤:

①从 V_t 回归所有"前象"(即前提)状态。

②在回归的状态上计算 Q_{t+1}值。

③用最大化方法计算 V_{t+1}。

下面依次讨论每一步。

3.4.1 回归算法

首先给出一个重要定义：如果S是一个抽象状态，当遵循某个动作规则$\mathbb{H}_i \xleftarrow{p_i:\mathbb{A}} \mathbb{B}$后能导致到抽象状态$\mathbb{S}'$，则称它为$\mathbb{S}'$关于动作$\mathbb{A}$的第$i$个结果的一个最弱前置条件状态，用记号$wp_i(\mathbb{A},\mathbb{S}')$表示所有$\mathbb{S}'$关于$\mathbb{A}$的第$i$个结果的最弱前置条件状态集合。文献[6]给出 WEAKESTPRE 程序，它返回关于动作规则$\mathbb{H}_i \xleftarrow{p_i:\mathbb{A}} \mathbb{B}$和抽象状态$\mathbb{S}'$(给定了一个完全约束集合 C)的最弱前置条件状态集合wp_i。下面我们举例说明该算法。

例如，考虑动作 move：

$$\begin{array}{ll} on(X,Y), & cl(X),cl(Y), \\ cl(X),cl(Z), & \xleftarrow{0.9:move(X,Y,Z)} \quad on(X,Z), \\ X\neq Y, & X\neq Y, \\ Y\neq Z,X\neq Z & Y\neq Z,X\neq Z \end{array}$$

$$\begin{array}{ll} on(X,Z), & cl(X),cl(Y), \\ cl(X),cl(Y), & \xleftarrow{0.1:move(X,Y,Z)} \quad on(X,Z), \\ X\neq Y, & X\neq Y, \\ Y\neq Z,X\neq Z & Y\neq Z,X\neq Z \end{array}$$

它有两个结果：第一个结果是该动作转移 X 到 Y 上，其概率为 0.9；而动作失效为第二个结果，概率为 0.1，即它并没有改变状态的概率为 0.1。令 $S\equiv(cl(a),cl(b),on(a,c),on(b,d))$(省略了不等式关系约束)，$S'\equiv on(a,b)$。显然 $S\in wp_1(move(X,Y,Z),S')$，但是 $S\notin wp_2(move(X,Y,Z),S')$。

为了计算(例如)$wp_1(move(X,Y,Z),S')$，我们能够假定从 S 可以移到 S'。因此，①动作规则的条件必须在 S 状态下能够满足；②S'是部分由该动作的第一个结果导致的。第①点是明显的。为了阐明第②点，现在考虑 $S'=on(a,b)$，可以有两种情况：其一，移动导致 $on(a,b)$。例如当我们处于状态 $S_1\equiv(cl(a),cl(b),on(a,Z),a\neq b,a\neq Z,b\neq Z)$时，$moveX=a$到$Y=b$上。其二，移动并未导致 $on(a,b)$，我们移动 X 到 Y 上但不是 a 到 b 上。例如当我们处于状态 $T\equiv\begin{pmatrix}cl(X),cl(Y),on(X,Z),\\ on(a,b),X\neq Y,X\neq Z,Y\neq Z\end{pmatrix}$，它满足我们并未移动 a 到 b 上的条件，换言之，我们动作并未涉及 $on(a,b)$，即 $on(X,Y)\neq on(a,b)$以及 $on(X,Z)\neq on(a,b)$。只是因为 T 中原来就包含 $on(a,b)$，而约束条件又保证了当我们在 T 中应用 $move(X,Y,Z)$时又保留了 $on(a,b)$。于是在这种情况下，前置条件状态的定义可以简化为：$S_2\equiv(T\wedge X\neq a)$，$S_3\equiv(T\wedge X\neq a\wedge Z\neq b)$，$S_4\equiv(T\wedge Y\neq b\wedge X\neq a)$以及 $S_5\equiv(T\wedge Y\neq b\wedge Z\neq b)$，而所有 $S_i(i=2,3,4,5)$都可以完备成相同的状态，即 $S_6\equiv cl(A),cl(B),on(a,b),on(A,C)$，其中所有变量和常量都不相同。从而抽象状态 S_1，S_6 在一起逻辑地定义了 $wp_1(move(X,Y,Z),S')=(S_1\vee S_6)$。

直到现在，我们考虑的只是仅有一个“效果”，即 $S'=(on(a,b))$的情况，对于一般情况。可能有多种组合“效果”，也可以类似处理。

有了最弱前置条件的概念，我们可以大体描述关系 Bellman 后向传播运算。假设 V_t 是目前得到的抽象状态值函数，比如说 V_0，并且考虑抽象动作 *move*。那么作为一次 Bellman 反向传播，所有抽象状态 S 当执行动作 *move*，它能导致 V_0 的一个条件，都应该被计算。但在计算 V_{t+1} 之前，我们要计算 Q_{t+1}，即进入第二个步骤。

3.4.2 计算抽象状态动作值函数

给定上述回归的抽象状态和目前抽象状态值函数 V_t，文献[6]给出程序 2 QRULES，它是一个计算抽象状态-动作值函数 Q_{t+1} 的算法。

当给定了回报模型 R，目前值函数 V_t 和折扣因子 γ，该算法返回一个动作$\mathbb{A}$的 Q 规则。为了计算一个动作的 Q 规则，必须：①把一个动作$\mathbb{A}$的每一个结果都看成是单一动作，并计算相应的抽象状态-动作值函数。②对于动作$\mathbb{A}$，我们结合所有结果的值为一个抽象状态-动作值函数。下面举例说明这两步。在说明时，直到下一段之前，我们都省略掉约束条件的陈述。

关于第①步，考虑前面关于 *move* 第一个结果的例子，这时 $wp_1(move(X,Y,Z),S')\equiv S_1\vee S_6$。回忆回报函数模型(即 V_0)，S_6 是吸收的，所以执行动作 *move*。我们指定该抽象状态-动作值为 10，即 $10:move(X,Y,Z)\leftarrow S_6$。至于 S_1，有关计算必须依赖于 $V_t(S')$，即我们例子中的 $V_0(S')$。现假设折扣因子是 0.9，这导致 $R(S)+p_1\cdot 0.9\cdot V_0(S')=0+0.9\times 0.9\times 10=8.1$，即 $8.1:move(a,b,Z)\leftarrow S_1$。对于 V_0 中所有其他规则都做了相同事情以后，我们得到：

〈a〉 $10:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(a,b),on(X,Z)$

〈b〉 $8.1:move(a,b,Z)\leftarrow cl(a),cl(b),on(a,Z)$

〈c〉 $0.0:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(X,Z)$

因为首先考虑的是关于动作 *move* 第一个结果，那么由 *move* 第一个结果所产生的〈a〉，〈b〉，〈c〉就作为一个“暂时结果”储存，对于第一个结果无须执行第②步。考虑 *move* 动作的第二结果，同样由第①步得到：

〈d〉 $1.0:move(a,X,b)\leftarrow cl(a),cl(X),on(a,b)$

〈e〉 $1.0:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(a,b),on(X,Z)$

〈f〉$0.0:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(X,Z)$

现在我们执行第②步。关于第②步，我们注意到这些规则的每一个都描述如此情景，即我们如果处于该状态，那么我们可能得到某些值，它是关于动作$\mathbb{A}$的第 i 结果方面的。因此这些信息就必须结合起来，成为关于$\mathbb{A}$的抽象状态-动作的值。为了做到这一点，我们从〈a〉—〈c〉里选择一个规则，比如说〈b〉，并且从〈d〉—〈f〉里选择一个规则，比如说〈f〉，然后检查我们是否能同时处于那两个状态，以及是否能够运用相同的动作。换言之，我们计算这些逻辑子句(在两个值规则里)的最大下界(glb)。如果 glb(其中动作必须合一)存在，并且它是合法状态，则它就作为新规则储存，但该新规则的值是结合规则的值的和。回到〈b〉和〈f〉，这导致 $8.1:move(a,b,Z)\leftarrow cl(a),cl(b),on(a,Z)$。作为对比，〈b〉和〈d〉就不能结合给出新规则。

在上述积木世界例子里，当 QRULES 运用到 V_0 和 *move*，*absorbing* 两动作，则产生下述抽象状态-动作值函数：

〈1〉$10:absorbing\leftarrow on(a,b)$

〈2〉$10:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(a,b),on(X,Z)$

〈3〉$8.1:move(a,b,X)\leftarrow cl(a),cl(b),on(a,X)$

〈4〉$0.0:move(X,Y,Z)\leftarrow cl(X),cl(Y),on(X,Z)$。

3.4.3 计算抽象值函数

由 QRULES 程序得到的 Q-规则(Q-rules)集合使得我们能够去计算下一步抽象状态值函数 V_{t+1}。和传统情况相比，表示抽象状态-动作值的 Q-rules 可能重叠，例如上面 Q-rules 里的〈1〉和〈2〉。为了计算抽象状态值函数，我们根据传统做

法，利用下面事实 $V_{t+1}(\mathbb{S})=\max_A Q_{t+1}(\mathbb{S},\mathbb{A})$。鉴于此，文献[6]运用一个简单的分而治之(Separate-and-Conquer)规则学习方法(见文献[6]程序 3 VRULES)。如果对于所有动作都从 V_t 计算了 Q-rules，则通过程序 VRULES 返回值函数 V_{t+1}。

程序首先在 Qrules 中搜索到具有最大 Q-值的 Q-rule m，然后分离出那些覆盖的 Q-rules，再递归处理剩下来的 Q-rules，靠选择更多的规则直到没有 Q-rules 保留。注意最主要的地方是，仅仅当在 Qrules 里没有其他如此 Q-rule 留下，该 Q-rule 与 Q-rule m 有相同的值，且它的体(body)包含了 m 的体，这时才选择 m 并把它加到 V_{t+1} 中去。

在上面运行的例子，我们从规则〈1〉开始，因为它没有被任意其他具有相同值的规则所包含，所以我们增加 $10\leftarrow on(a,b)$ 到 V_1 中去。而且，由于它包含了规则〈2〉，我们从 Qrules 中移去〈2〉，于是剩下来有最高值的规则是〈3〉，再迭代下去。最后，通过完备化，这样产生了新的值函数(再一次列出约束)：

$10\leftarrow on(a,b), a\neq b$

$8.1\leftarrow cl(a), cl(b), on(a,X), a\neq b, a\neq X, b\neq X$

$0\leftarrow cl(X), cl(Y), on(X,Z), X\neq Y, X\neq Z, Y\neq Z$

总结上面 3 个步骤组成的算法称之为 REBEL 算法(Relational Bellman Backup Operator)，它把 Bellman 方程推广到抽象关系领域，从而可以“纯粹”地脱离实际状态空间，独立寻找抽象状态空间的最优策略，这是该算法的可贵之处。

3.5 评论

在本质上，LOMDPs 和 RMDPs 中的 CARCASS 是相同的，即由它们表达的抽象结构可以“外化”成具体的 MDPs 的实际状态空间或实际的状态-动作空间。而该抽象结构实际上是相应的实际结构的概括，并且可以依赖抽象结构对实际结构进行分割划分，以致于把一些状态看作是类似的，因而可在它们之上实施类似的动作。但是，就我们所介绍的 LOMDPs 和 RMDPs 而言，二者还是有区别的。在 LOMDPs 中抽象转移 $T\ \mathbb{H}\xleftarrow{p;r;\mathbb{A}}\mathbb{B}$ 的定义中，其转移概率 $P(T)=p$ 以及回报 $R(T)=r$ 怎样由实际状态转移概率与实际回报函数分别确定，并未明确交待，或者换言之，它们之间的关系并不清楚。而 RMDPs 却回避了这个问题，因为它只定义抽象动作规则，并且在〈P,A,D,T,R〉的定义中，T 与 R 是实际 MDP (S,A,T,R) 中的转移概率和回报函数。

虽然具有概括性功能是一切抽象结构的共同特征之一，但是 LOMDPs 和 RMDPs 的 CARCASS 的精妙之处就是立足于概括性，实现对实际结构的分割或聚类，从而利用抽象层次上的结果指导具体层次上的行为。特别当实际结构过于复杂庞大，这种指导作用就更为显著和有效。通过概括功能，庞大复杂的实际问题可以在简约的抽象层次上考虑，但是它也引入了下述几个问题：

①LOMDPs 中关于抽象策略 $\tilde{\pi}$ 的抽象层次 $\mathbb{L}$ 的确定问题。$\mathbb{L}$ 的确定除了要满足能够覆盖实际空间的条件外，还应该满足其它什么条件才能使得在该层次上得到的最优抽象策略具有对实际问题更好的指导意义？

②在 RMDPs 中，运用 CARCASS 工具学习抽象 Q 函数，虽然它并未事前规定抽象层次，而是动态地迭代确定下一个抽象状态-动作对，并对它计算 Q 值。同样，运用这种技术得到最优策略对实际问题具有怎样的指导意义？

③一般地，在抽象层次上得到的确定性最优策略，归约到实际状态而形成的策略往往是不确定的。因此直觉上，在平均意义上，如此得到的实际策略是最优的，然而无论是在 LOMDPs 或是 RMDPs 理论中，该结论并未得到严格证明。

把 Bellman 方程推广到关系领域，作为逻辑编程的一部分，这个框架为关系增强学习领域带来了一个新的方向。众所周知，基于 Bellman 最优方程引入的传统的值迭代 VI(value iteration)技术要求所有状态及其值都必须清晰地表达在一个表中。但这在关系领域，由于状态数目可能非常大，甚至无限，这种技术是无法实现的。因此，Bellman 方程推广到逻辑编程抽象层次上，由此得到的(抽象)值迭代 VI 技术，就更具有重大实践意义。

虽然 REBEL 方法无论是在实践上或是理论上都很重要，可是它远没有在实际问题中所遵循的 Bellman 方程来得简单。问题是否出在我们追求的抽象层次过于概括，过于一般性？例如在积木世界里，无论是 LOMDPs 或是 RMDPs，里面叙述的抽象状态都达到这样普遍性，即不论积木块的数目多少都适用。这样的普遍性致使某些抽象动作看似完全不同，而实质却是一样的情况出现，这当然会引起不必要的麻烦[6]。

结束语 关于在随机系统里作出决策理论规划(DTP)，MDPs 是一个自然模型。然而，经典 MDP 框架需要清晰地枚举状态和动作空间，该空间随着状态变量(或领域特征)数目的增加而呈指数级增长。然而，大部分人工智能问题(包括 DTP)能够运用逻辑表达，从而获得简洁表述。因此，为了表达和解决大型状态空间 MDPs，学者们通过对给定随机系统领域特征的研究，开始运用高层次表达语言。1994 年 Steve Hanks，1995 年 Nicholas Kushmeri et. al 运用命题逻辑表达动态系统，即延伸 STRIPS(the Stanford Research Institute Planning System)为 Probabilistec (概率) STRIPS，简写为 STRIPS。本文介绍的文献[2]里许多思想都与他们的思想一致。1989 年，Thomas Dean 和 Keiji Kanazawa 运用动态贝叶斯网 DBN 表达动态领域(即状态随时间变化的领域)。运用 DBN 表述一个马尔可夫动态系统时，状态的转移模型可以运用 DBN 里的条件概率表(CPT)来计算。文献[2]也运用了 DBN 工具。此外，1993 年 J. Ross Quinlan 提出决策树，R. Iris Bahar 等人提出代数决策图(ADD)，1995 年 David Poole 提出逻辑规则，这些方法都能简洁表达转移和回报函数模型。上述基于命题逻辑对动态系统结构的表达，里面的思想和方法都为 MDPs 在更抽象的层次上描述开了先导。

2000 年 Boutilier，Dearden & Goldszmidt 提出运用因子化 MDPs 简洁表达大型有结构的 MDPs。在这个框架下，状态可以由对状态变量的指派来描述，而转移模型可以由 Dean & Kannazawa 1989 年提出的动态 Bayesian network (DBN) 简洁表达。2003 年，Carlos Guestrin 等人[2]系统描述了在这个框架下近似求解技术，他们基于 Koller 和 Parr 在 1999 年和 2000 年的工作，运用因子化值函数以及在因子化 MDP 中的 additive 和 context- specific 结构，提出许多新算法和新技巧，分别在策略迭代和线性编程方面开发了十分有效的近似算法。

最近，人工智能研究者对于在 MDP 中运用更丰富、更有力的表达语言的兴趣越来越高涨。Boutilier 等人于 2001 年(文献[1])基于结构动态编程在关系 MDPs 中提出了第一个精确解技术。在文献[1]中，他们运用(允许随机动作的)情景

(下转第 48 页)

[18] Deng J, Han R, Mishra S. Secure code distribution in dynamically programmable wireless sensor networks// ACM/IEEE Conference on Information Processing in Sensor Networks. Nashville, TN, April 2006: 292-300

[19] Barr K, Asanovic K. Energy aware lossless data compression// The First International Conference on Mobile Systems, Applications, and Services. San Francisco, CA, May 2003

[20] Kim D H, Gandhi R, Narasimhan P. Exploring symmetric cryptography for secure network reprogramming // International Workshop on Wireless Ad-hoc and Sensor Networks. New York, NY, June 2007

[21] Perrig A, Szewczyk R, Tygar J D, et al. SPINS: Security protocols for sensor networks. Wireless Networks, 2002, 8(5): 521-534

[22] Perrig A, Canetti R, Song D, et al. Efficient and secure source authentication for multicast// Proceedings of Network and Distributed System Security Symposium. San Diego, CA, February 2001: 35-46

[23] Krontiris I, Dimitriou T. Authenticated in-network programming for wireless sensor networks// International Conference on Ad-Hoc Networks and Wireless. Ottawa, Canada, August 2006

[24] Lamport L. Constructing digital signatures from a one-way function. Technical Report CSL-98. SRI International Computer Science Laboratory, Palo Alto, 1979

[25] Merkle R C. A certified digital signature// Proceedings on Advances in cryptology (CRYPTO '89). New York: Springer-Verlag, Inc., 1989: 218-238

[26] Lanigan P E, Narasimhan P, Gandhi R. Tradeoffs in Configuring Secure Data Dissemination in Sensor Network: An Empirical Outlook. CMU-CyLab-07-006. CyLab, Carnegie Mellon University, PA, 2007

[27] Wood A D, Stankovic J A. Denial of Service in Sensor Networks. IEEE Computer, Oct. 2002: 48-56

(上接第 14 页)

演算动态表达一个 MDP，并且通过一个称为决策理论回归(decision-theoretic regression)运算，精确产生最优值函数和最优策略的逻辑描述，它是 Reiter's 情景演算(situation calculus)的概率延伸，即是一阶决策理论回归。虽然情景演算语言富有表达力，然而它比较复杂。2003 年，Kristian Kersting 和 Luc De Raedt 提出逻辑马尔可夫决策过程，2004 年 Martijn Van Otterlo 提出关系马尔可夫决策过程，这些都是基于逻辑编程的系统。在某种意义上说，LOMDPs 和 RMDPs 和文献[1]中的思想是一致的，但比较简单些。也是基于文献[1]中的思想，Kersting, Van Otterlo & De Raedt，于 2004 年成功地把 Bellman 方程推广到关系领域，从而在逻辑层次上精确求解抽象值函数和抽象最优策略。

在近似求解和模型自由(model free)方面，Kerstian Kersting 和 Luc De Raedt 在 LOMDPs 中、Martijn Van Otterlo 在 RMDPs 中都提出了算法。但是这些算法都是基于传统的近似值函数算法，2006 年，Alan Fern, Sungwook Yoon & Robert Givan[7] 中运用类似于"可能学习近似正确假设"的机器学习方法，直接在偏置的策略空间中学习近似最优策略。文献[7]是值得阅读的，因为它提出的算法与传统不同，是直接在偏置(用一种表达策略的语言而得到)的策略空间里搜寻最好策略，从而为抽象近似策略迭代提供一个范例。当然，结构型里有关情景演算和因子化 MDPs 以及概括型里有关 LOMDPs 和 RMDPs 的文献及其最优策略的算法是较多的，但为了使我们的介绍既简单而又不流于空泛，在我们阅读许多文献后，根据自己的研究，把自己认为最主要的概念及代表不同风格和趋向的算法精选出来，加以较详细的介绍。至于其他作者的工作大都在我们所引的文献中有所涉及，故略而不述，有兴趣的读者可在我们所列的文献中找到有关他们的信息。

最后，对于 LOMDPs 和 RMDPs 今后的发展，我们简单提出几点看法。

①所有文献都指出抽象层次上的最优策略是有价值的，但这个论点至今尚未看到清晰的、完整的证明。我们认为，既然 LOMDPs 和 RMDPs 都基于这样一个事实，即逻辑层次上的表述能把基础层次上的状态和动作按某种类似性分类，那么我们就可以运用现代概率论中的条件数学期望概念来论证在抽象层次上的最优策略和实际层次上的具体最优策略之间的关系。我们认为，可以证明抽象层次上最优策略在平均意义上是基础层次上的最优策略。

②所有文献都把抽象层次和基础层次交织在一起，这为寻求抽象最优策略带来了很大麻烦。我们认为，也可以把这两个层次"完全"分离，转变为两个层次上的马尔可夫决策过程，这样在抽象层次上，利用它比实际状态数目较少的优点，较简单地运用传统方法求抽象最优策略，再按某种演算具体在实际状态空间实施，可能起到事半功倍之效。这个思路更符合人类思维特点，因为人们总是首先在大体上思考问题解决方案，然后再付诸实践，具体实现之。

参 考 文 献

[1] Boutilier C, Reiter R, Price B. Symbolic Dynamic Programming for First-order MDPs// Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01). Seattle: USA, 2001: 690-700

[2] Guestrin C, Koller D, Parr R, et al. Efficient Solution Algorithms for Factored MDPs. Journal of Artificial Intelligence Research, 2003, 19: 399-468

[3] Kersting K, Raedt L D. Logical Markov Decision Programs// Working Notes of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data (SRL-03). Acapulco, Mexico, 2003: 63-70

[4] Kersting K, Raedt L D. Logical Markov Decision Programs and the Convergence of Logical TD(λ)// Proceeding of The 14th International Conference of Inductive Logic Programming. Porto, Portugal, 2204: 180-197

[5] van Otterlo M. Reinforcement Learning for Relational MDPs// Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands. Brussels, Belgium, 2004: 138-145

[6] Kersting K, van Otterlo M, Raedt L D. Bellman goes to Relational// Proceedings of the 21st International Conference on Machine Learning. Banff, Canada, 2004

[7] Fern A, Yoon S, Givan R. Approxiamte Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. Journal of Artificial Intelligence Research, 2006, 25: 75-118