ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.3, March 2007, pp.490−504
DOI: 10.1360/jos180490

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 STRIPS 的领域知识提取策略*

吴向军[1]，姜云飞[2]，凌应标[3+]

[1](中山大学 软件学院,广东 广州 510275)

[2](中山大学 信息科学与技术学院 软件研究所,广东 广州 510275)

[3](中山大学 信息科学与技术学院 计算机科学系,广东 广州 510275)

Strategy of Extracting Domain Knowledge for STRIPS World

WU Xiang-Jun[1], JIANG Yun-Fei[2], LING Ying-Biao[3+]

[1](Software School, SUN YAT-SEN University, Guangzhou 510275, China)

[2](Software Research Institute, School of Information Science and Technology, SUN YAT-SEN University, Guangzhou 510275, China)

[3](Department of Computer Science, School of Information Science and Technology, SUN YAT-SEN University, Guangzhou 510275, China)

+ Corresponding author: Phn: +86-20-84036776, E-mail: isslyb@mail.sysu.edu.cn

Wu XJ, Jiang YF, Ling YB. Strategy of extracting domain knowledge for STRIPS world. *Journal of Software*, 2007,18(3):490−504. http://www.jos.org.cn/1000-9825/18/490.htm

Abstract: In this paper, a similarity relation between two predicates is defined first. To a given predicate, the set of action for the predicate can be obtained by the similarity relation. Then, the domain knowledge is extracted from the common fluent in preconditions and effects of all actions for each set of action, and the formalism for the domain knowledge is given. Finally, the contradictions in the initial states and the goal states in a particular planning problem with domain knowledge can be discovered. The strategy of extracting the domain knowledge is integrated in the planner StepByStep, and the domain knowledge is the necessary theory when one predicate by action is realized in the planner.

Key words: artificial intelligence; AI planning; planning domain; STRIPS; similarity relation; substitution; domain knowledge

摘 要: 提出了谓词之间的一种相似关系,并用该相似关系得到可实现某谓词的动作集.利用该动作集中所有动作的公共前提谓词和公共效果谓词,提取出隐含在动作描述中的领域知识,并给出了描述领域知识的一种形式化方法.最后,对具体的规划问题,可利用领域知识判断出初始状态或目标状态中存在的矛盾.该领域知识的提取策略已应用于智能规划器 StepByStep 之中,所获取的领域知识对选择待实现的谓词提供了必要的理论依据.

关键词: 人工智能;智能规划;规划领域;STRIPS;相似关系;置换;领域知识

中图法分类号: TP18 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60173039 (国家自然科学基金); the Special Foundation of 985 Project of SUN YAT-SEN University of China (中山大学 985 工程专项资金)

Received 2006-04-04; Accepted 2006-05-11

智能规划是人工智能领域里的一个极其活跃的研究方向.至今,研究人员已研究开发了几十种不同类型的规划器,其中,比较著名的、应用于具体领域中的有 RAX/PS(NASA 太空一号控制器)、HSTS(哈勃太空望远镜调度器)和 MACHINE(工业生产制造设计的非线性规划器)等.从规划器与规划领域相关性来看,可分为“与领域有关的”规划器(domain-independent planner)和“与领域无关的”规划器(domain-dependent planner).

“与领域有关的”规划器是在规划器中加入特定领域的领域知识、规则或常识,从而避免一些不必要的动作或推理,达到提高规划效率的目的.但除了具体的应用系统之外,研究者都希望开发出高效的、“与领域无关的”规划器.为了提高规划器的效率,从领域描述信息中自动获取领域知识就是规划策略中的重要内容.

在提取领域知识研究方面,早期的工作有:TIM[1],它是用于规划器 STAN 中的一个独立的领域分析工具.它从领域定义和问题实例的信息中分析出隐式的状态常量(state invariant),然后,在规划中直接使用这些隐含信息,从而达到改善规划器性能的目的[1,2],这种分析技术被称为状态分析技术(state analysis).

图规划(GraphPlan)[3]方法是近年来智能规划研究中的一个重要方法.在规划求解时,利用动作和谓词之间的二类互斥关系来指导具体的求解过程.这些互斥关系是用动作描述和逻辑规则来获得的.但图规划方法存在着如下不足,即谓词或动作之间的互斥关系是与“具体的规划实例”相关的,不能反映该领域中的一般规律.

在表达领域知识方面,目前有直接在领域描述中添加领域知识的方法,如用 PDDL(planning domain definition language)中的 Axiom 子句[4]来表达领域知识;有用时态模态逻辑规则来表示领域知识的,如 TLPLAN[5−7]和 TALplan[8−10]等;也有把领域知识隐含在启发式函数中,并通过不同的“加权”来体现不同领域知识的作用,如 FF[11−13],AltAlt[14−16]和 SATPLAN[17,18]等.总之,为了提高规划器的效率,不同的规划器会以不同的方式来运用领域知识.

在研究规划领域的定义时,我们发现:领域设计者在编写领域的动作定义时,在动作的前提条件和动作效果中已隐式地表达了相关的领域知识.在规划求解时,这些领域知识能够准确地约束规划动作的执行.另一方面,STRIPS 是规划领域中最经典的领域类型[19],是研究智能规划的重要基础,很多智能规划器也都是以求解这类规划领域为目标的,如 Blockbox[20−22],GraphPlan,AltAlt,GRT[23−25],MIPS[26,27]等.所以,本文以 STRIPS 规划领域为研究对象,以领域中的动作定义为依据来提取相应的领域知识.

本文第 1 节简单介绍规划动作的描述方式,通过对动作信息的分析,反映出第 2~4 节研究内容之间的内在关系.第 2 节研究动作中的谓词与指定谓词之间的一致化问题,其置换思想与人工智能领域中的置换类似,但也有不同之处.第 3 节讨论动作参数的置换原理,主要解决如何把效果谓词的置换扩展为动作参数的置换问题.第 4 节给出领域知识的提取策略,并分析有关领域知识对领域状态的判断作用.

1 规划领域的动作描述

在本文的规划研究中,规划领域是用 PDDL 语言来描述的[28−30].PDDL 描述动作的一般形式如下:

(:Action *Act*
 :Parameters (P_1 P_2 … P_k)
 :Precondition (Pc_1 Pc_2 … Pc_s)
 :Effect (and Pd_1 Pd_2 … Pd_t)
)

其中:*Act* 是动作名,Pc_i 和 Pd_j 是领域中的谓词,i=1..s, j=1..t.

动作描述的含义是:在当前状态下,若前提条件 $Pc_1,Pc_2,\ldots,Pc_s$ 都为“真”,则可执行动作 *Act*,且用效果谓词 $Pd_1,Pd_2,\ldots,Pd_t$ 来修改当前状态,使“当前状态”变成“下一状态”.

对规划领域中的任意动作 *Act*,我们用 *Para*(*Act*),*PC*(*Act*)和 *E*(*Act*)分别表示其参数表(parameter)、前提条件(precondition)的谓词集合和动作效果(effect)的谓词集合.

下面根据领域动作的描述和功能来开展本文的研究.在规划问题中任取一个谓词,“如何选用领域动作来实现之?”是智能规划研究的最基本、也是最重要的问题.为了清楚地表达本文的研究内容,我们先用一个最著名的

规划领域 BlocksWorld 来加以说明.

假设在规划领域 BlocksWorld*中(完整的领域定义见附录 A),目前需要实现谓词(*holding x*).

在 BlocksWorld 规划领域中,其动作 *Unstack* 的描述如下:

(:Action *Unstack*
:Parameters (?*x* ?*y*)
:Precondition ((*clear* ?*x*)(*on* ?*x* ?*y*)(*arm-empty*))
:Effect (and (*not* (*clear* ?*x*))(*not* (*on* ?*x* ?*y*))(*not* (*arm-empty*))(*holding* ?*x*)(*clear* ?*y*))
)

在动作效果 *E*(*Unstack*)中有谓词(*holding* ?*x*),但该谓词与(*holding x*)之间存在某种差异,需要用置换{*x*/?*x*}来使它们达到一致化.对规划动作中的谓词与待实现谓词之间一致化的研究,在第 2 节中有详细描述.

对于动作 *Unstack*,虽然用置换{*x*/?*x*}实现了谓词(*holding* ?*x*)和(*holding x*)之间的一致性,但(*holding* ?*x*)只是要该动作所涉及的一个谓词,其他谓词也必须与其一起进行相应的变化,即需要把置换{*x*/?*x*}扩展到动作参数(?*x* ?*y*)上,从而使该动作中的所有谓词都进行相应的变化.第 3 节研究了规划动作参数进行置换的问题.

对规划领域中的一个谓词,一般情况下会有多个动作来实现.若规划领域中的每个谓词都只有一个规划动作来实现,那么,该规划问题可能会蜕变成比较简单的求解问题.

在规划领域 BlocksWorld 中,动作(*Pickup* ?*x*)和(*Unstuck* ?*x* ?*y*)都有可能用来实现谓词(*holding x*).

令$\sigma_1=\{x/?x\}$,$\sigma_2=\{x/?y\}$,则(*holding x*)$\in E(\sigma_1 Pickup)$,(*holding x*)$\in E(\sigma_2 Unstack)$**.

所以,动作$\sigma_1 Putdown$ 和$\sigma_2 Unstuck$ 都可实现谓词(*holding x*).由这些动作在实现谓词(*holding x*)时所具有的共同谓词可反映出该规划领域的一些内在特征,如:不论采用何种领域动作实现谓词(*holding x*)时,都必须先有谓词(*arm-empty*),且在实现谓词(*holding x*)后,谓词(*arm-empty*)也一定不在规划状态之中.因此,谓词(*holding x*)和(*arm-empty*)之间存在着某种互斥关系.与此相似的、隐含在动作描述中的领域知识都可以从一组相关的动作描述中提取出来.从相关领域动作中自动提取领域知识的提取方法在第 4 节中给出了明确的表达.

综上所述,本节从领域动作出发,概括说明了通过谓词的一致化置换可使动作的效果谓词和待实现谓词变为同一形式.但由于动作中谓词的变化需要通过动作参数的置换才能实现,所以,需要把谓词的一致化置换扩展到动作参数上.最后,对可实现某个谓词的所有动作,利用它们的共同谓词来获取相应的领域知识,并运用这些领域知识对规划问题进行必要的判断.

2 谓词的置换原理

由上一节可知:在用动作实现某个谓词时,需要使动作中相应的效果谓词与待实现的谓词之间进行置换,而谓词置换是针对谓词参数表所进行的,所以,本节先讨论参数表的置换,然后再映射到谓词参数表上.

2.1 参数表的相似关系

为了形式化研究参数表之间的变换,下面定义几个符号和概念.

定义 **2.1**. 用 *Para* 表示参数表(P_1 P_2 … P_k),用 *V*(*Para*)表示参数表 *Para* 中所有变量组成的集合,称 *V*(*Para*)中的参数为变量参数.

例如:*Para*=(?*x A* ?*y*),*V*(*Para*)={?*x*,?*y*},?*x* 和?*y* 是变量参数, *A* 是某个具体的常量.

定义 **2.2**. 假设有参数表 $Para_1=(P_1\ P_2\ \dots\ P_k)$和 $Para_2=(Q_1\ Q_2\ \dots\ Q_k)$,对应参数 P_i 和 Q_i,i=1..k,若它们满足下列条件:

(1) 若 Q_i 是常量,则 $P_i=Q_i$;

(2) 若 Q_i 是变量,则 P_i 是常量,或是同类的变量;

* 本文是以 BlocksWorld 领域为例来说明有关概念,但讨论的内容和所得的结论对基于 STRIPS 的规划领域都有效.

** 对动作进行置换的含义见第 3 节中的"定义 3.2",本节是从宏观上介绍本文内容的组织结构.

(3) $Q_i=Q_j \Rightarrow P_i=P_j$,

则称参数表 $Para_2$ 相似于 $Para_1$,记为 $Para_1 <\sim Para_2$.

定义 2.2 中的条件(1)说明,$Para_2$ 中的常量参数对应相同的常量参数(在某些规划领域的定义中存在常量问题);条件(2)说明,$Para_2$ 中的变量参数可对应常量或同类的变量参数;条件(3)说明,$Para_2$ 中相同的参数在 $Para_1$ 中对应相同的参数.

由定义 2.2 可知:参数表的相似关系"<~"不具有对称性,即 $Para_1<\sim Para_2$,不一定有 $Para_2<\sim Para_1$.

例如:$(?x_1\ ?y_1)<\sim(?x_2\ ?y_2)$,$(?x_1\ A\ ?x_1)<\sim(?x_2\ A\ ?x_2)$,$(?x_1\ A)<\sim(?x_2\ ?y)$,但$(?x_2\ ?y)<\sim(?x_1\ A)$不成立,因为常量 A 对应前面参数表中的变量参数"?y",其中,变量$?x_1$和$?x_2$,$?y_1$和$?y_2$是同类变量.

2.2 谓词的相似关系

在研究谓词的变换时,我们称形如$(Name_1\ P_1\ P_2\ \dots\ P_s)$的谓词为正谓词,$(not\ (Name_2\ Q_1\ Q_2\ \dots\ Q_t))$为负谓词,称谓词的正负属性为谓词的类型,其中,$Name_1$和 $Name_2$ 是谓词名.

用 $Type(Pd)$,$Name(Pd)$和 $Para(Pd)$分别表示谓词 Pd 的类型、谓词名和参数表.

例如:谓词 $Pd=(on\ ?x\ A)$,$Type(Pd)$="正",$Name(Pd)$="on",$Para(Pd)=(?x\ A)$.

定义 **2.3**. 假设有两个谓词Pd_i和Pd_j,若 $Type(Pd_i)=Type(Pd_j)$,$Name(Pd_i)=Name(Pd_j)$,且 $Para(Pd_i)<\sim Para(Pd_j)$,则称谓词 Pd_j 相似于 Pd_i,记为 $Pd_i<\sim Pd_j$.

定义 **2.4**. 假设有两个谓词 Pd_i和 Pd_j,$Name(Pd_i)=Name(Pd_j)$,$Para(Pd_i)=Para(Pd_j)$,那么,

(1) 当 $Type(Pd_i)=Type(Pd_j)$时,称谓词 Pd_i和 Pd_j 相等,记为 $Pd_i=Pd_j$;

(2) 当 $Type(Pd_i)\neq Type(Pd_j)$时,称谓词 Pd_i和 Pd_j 互反,记为 $Pd_i=(not\ Pd_j)$.

2.3 谓词的置换原理

置换是人工智能中一个常用的概念[19],它用来把若干个谓词变换成同一个谓词的形式.但本文所讨论的置换有其自身的特点.虽然置换作用与前者基本一致,但在置换的内容和作用方式上是不同的.本文所讨论的置换是针对谓词参数表和动作参数表进行的.

定义 **2.5**. 假设变换集合 $H=\{V_1/U_1,V_2/U_2,\dots,V_k/U_k\}$,若变换集合 H 满足下列条件:

(1) $\forall V_i/U_i\in H$,都有:$U_i\neq V_i$;

(2) $\forall V_i/U_i\in H$,都有:U_i 是变量,V_i 是常量,或是与 U_i 同类的变量;

(3) $\forall V_i/U_i,V_j/U_j\in H$,都有:$U_i=U_j \Rightarrow V_i=V_j$,

则称变换集合 H 为一个置换,记为σ,并用 $U(\sigma)$表示置换σ中被替换的变量集合$\{U_1,U_2,\dots,U_k\}$.

定义 2.5 中的条件(1)和条件(2)说明置换σ是把变量 $U_i(i=1..k)$置换成常量,或另一个同类的变量 V_i;条件(3)说明置换σ不能把变量 U_i 置换成两个不同的值 V_i 和 V_j.

由定义 2.5 可知:$\sigma=\varnothing$是一种特殊的置换,我们称之为空置换.

定义 **2.6**. 假设有参数表 $Para=(P_1\ P_2\ \dots\ P_k)$,置换$\sigma=\{V_1/U_1,V_2/U_2,\dots,V_m/U_m\}$,用$\sigma Para$ 表示置换σ对参数表 $Para$ 中的变量参数进行变换所得到的结果,具体的替换规则如下:

对$\forall P_i\in V(Para)$,进行如下操作:

(1) 若$\exists V_j/U_j\in\sigma$,且 $P_i=U_j$,则用 V_j 替换 $Para$ 中的变量参数 P_i;

(2) 若$\forall V_j/U_j\in\sigma$,都有 $P_i\neq U_j$,则变量参数 P_i 不作任何改变.

定义 **2.7**. 假设有谓词 Pd 和置换σ,用σPd 表示置换σ对谓词 Pd 的参数表进行置换所得的谓词.

由定义 2.7 可知:σPd 是对谓词 Pd 的参数表进行替换所得到的新的谓词形式,除改变谓词的参数表之外,谓词σPd 和 Pd 的类型与谓词名是一致的.

引理 **2.1**. 假设有谓词 Pd 和置换σ,则 $Para(\sigma Pd)=\sigma Para(Pd)$.

由定义 2.6 和定义 2.7 不难证得.

当$\sigma=\varnothing$时,由引理 2.1 和定义 2.4 可知:$Para(\varnothing Pd)=Para(Pd)$,$\varnothing Pd=Pd$.

定理 **2.1**. 假设有谓词 Pd_1 和 Pd_2,若 $Pd_1<\sim Pd_2$,则存在置换σ,使得:$Pd_1=\sigma Pd_2$,$U(\sigma)\subseteq V(Para(Pd_2))$.

证明:

由"$Pd_1<\sim Pd_2$"可知:$Type(Pd_1)=Type(Pd_2)$,$Name(Pd_1)=Name(Pd_2)$,$Para(Pd_1)<\sim Para(Pd_2)$.

再设:$Para(Pd_1)=(P_{11}\ P_{12}\ \dots\ P_{1k})$,$Para(Pd_2)=(P_{21}\ P_{22}\ \dots\ P_{2k})$.

(1) 构造一个变换集合 H

令:$H=\varnothing$.

对参数表$(P_{21}\ P_{22}\ \dots\ P_{2k})$中的每个参数 $P_{2i}(i=1..k)$,分两种情况加以分析.

(1.1) P_{2i} 是常量

由"$Para(Pd_1)<\sim Para(Pd_2)$"和定义 2.2 可知:$P_{1i}=P_{2i}$.

所以,参数 P_{1i} 和 P_{2i} 之间无须进行变换.

(1.2) P_{2i} 是变量

由"$Para(Pd_1)<\sim Para(Pd_2)$"和定义 2.2 可知:$P_{1i}$ 是常量,或是同类变量.

① 若 $P_{1i}=P_{2i}$,则变量参数 P_{1i} 和 P_{2i} 之间也无须进行变换;

② 若 $P_{1i}\neq P_{2i}$,则 $H\leftarrow H\cup\{P_{1i}/P_{2i}\}$.

综合情况(1.1)和情况(1.2)可知:对参数表 $Para(Pd_2)$中的所有变量参数按情况(1.2)的②进行处理,即可获得一个变换集合 H,且 $U(H)\subseteq V(Para(Pd_2))$.

(2) 证明:变换集合 H 就是置换σ

(2.1) $\forall P_{1i}/P_{2i}\in H$

由情况(1.2)可知:$P_{1i}\neq P_{2i}$,且 P_{2i} 是变量,P_{1i} 是常量,或是与 P_{2i} 同类的变量.

(2.2) $\forall P_{1i}/P_{2i},P_{1j}/P_{2j}\in H$

假设 $P_{2i}=P_{2j}$,即:变量 P_{2i} 是谓词 Pd_2 的第 i 个参数和第 j 个参数.

由"$Para(Pd_1)<\sim Para(Pd_2)$"和定义 2.2 可得:$P_{1i}=P_{1j}$.

综合情况(2.1)、情况(2.2)和定义 2.5 可得:变换集合 H 就是一个置换σ,且 $U(\sigma)\subseteq V(Para(Pd_2))$.

(3) 证明:$Pd_1=\sigma Pd_2$

令:$\sigma Para(Pd_2)=\sigma(P_{21}\ P_{22}\ \dots\ P_{2k})=(Q_1\ Q_2\ \dots\ Q_k)$.

下面用反证法证明:$Para(Pd_1)=\sigma Para(Pd_2)$,即$(P_{11}\ P_{12}\ \dots\ P_{1k})=(Q_1\ Q_2\ \dots\ Q_k)$.

假设:在参数表 $Para(Pd_1)$和$\sigma Para(Pd_2)$中,第 1 组不同参数为 P_{1i} 和 Q_i,即 $P_{1i}\neq Q_i$.

(3.1) $P_{2i}=Q_i$

由假设条件"$P_{1i}\neq Q_i$"可知:$P_{1i}\neq P_{2i}$.

由置换σ的构造过程可知:$P_{1i}/P_{2i}\in\sigma$.

所以,当用置换σ对参数表$(P_{21}\ P_{22}\ \dots\ P_{2k})$中进行变换时,参数 P_{2i} 被变换成 P_{1i},即 $Q_i=P_{1i}$.

这就与假设"$P_{1i}\neq Q_i$"相矛盾.

(3.2) $P_{2i}\neq Q_i$

由定义 2.6 可知:Q_i 是由变量参数 P_{2i} 通过置换σ变换所得,即 $Q_i/P_{2i}\in\sigma$.

由置换σ的构造过程可知:$Q_i=P_{1i}$.

这也与假设"$P_{1i}\neq Q_i$"相矛盾.

综合情况(3.1)和情况(3.2)可得:当 $P_{1i}\neq Q_i$ 时,不论何种情况,都能得出矛盾之处.

所以,$(P_{11}\ P_{12}\ \dots\ P_{1k})$和$(Q_1\ Q_2\ \dots\ Q_k)$中的每对参数都是相同的,即 $Para(Pd_1)=\sigma Para(Pd_2)$.

由引理 2.1 可知:$\sigma Para(Pd_2)=Para(\sigma Pd_2)$.

所以,$Para(Pd_1)=Para(\sigma Pd_2)$.

再由 $Type(Pd_1)=Type(\sigma Pd_2)$,$Name(Pd_1)=Name(\sigma Pd_2)$和定义 2.4 可知:$Pd_1=\sigma Pd_2$.

所以,若 $Pd_1<\sim Pd_2$,则一定存在置换σ,使得:$Pd_1=\sigma Pd_2$,$U(\sigma)\subseteq V(Para(Pd_2))$. □

定理 2.1 说明:任意两个相似的谓词都可以通过置换变成一种谓词形式.这种相似谓词之间可一致化的性质为提取领域知识提供了必要的理论基础.

3 动作的置换原理

领域动作是智能规划领域中的重要组成部分,是规划研究的主体之一.求出满足要求的最短动作序列是智能规划研究所追求的目标.所以,规划领域中的动作在智能规划研究中具有极其重要的位置.

下面用 BlocksWorld 规划领域中的 *Unstack* 动作描述加以说明,具体动作描述见第 1 节或附录 A.

由动作 *Unstack* 的描述可知:动作中每个谓词的每个变量参数都一定在动作参数表(?*x* ?*y*)之中,即:$\forall Pd\in PC(Unstack)\cup E(Unstack)$,都有:$V(Para(Pd))\subseteq\{?x\ ?y\}$.

由此不难看出,规划领域中的动作描述具有下列特征:

(1) 动作的描述是动作模式的一种说明

不同的动作参数对应不同的具体动作,但它们都是同一种动作.如:(*Unstack A B*)和(*Unstack B C*)就是动作 *Unstack* 的两个不同的具体动作.

(2) 动作内部所包含的信息具有封闭性(常量除外)

动作的描述是一个独立的、完整的个体,它把动作的变化封装在动作参数的变化之中.动作描述中的信息是随动作参数的变化而变化的.当动作参数发生变化时,其所需要的前提条件和所得到的效果才会发生变化.这也就是说,对任意一个动作,我们不能直接改变其前提条件中的谓词参数,也不能直接改变其动作效果中的谓词参数,只能通过改变动作参数表来达到改变这些内部信息的目的.因此,动作及其内部信息的改变,其实质都是它们参数表的改变.

假设有待实现的谓词 Pd^{***},存在一个动作 $Act,Pd_i\in E(Act)$,且 $Pd<\sim Pd_i$.

由定理 2.1 可知:一定存在置换σ,使得:$Pd=\sigma Pd_i,U(\sigma)\subseteq V(Para(Pd_i))$.

由于动作是独立的个体,所以,我们不能单独用置换σ对其效果谓词 Pd_i 的参数表进行置换,而需要把置换σ扩展到动作 *Act* 的参数表上.只有通过动作 *Act* 的参数置换,才能统一地改变整个动作的内部信息.

下面研究把谓词参数上的置换扩展到动作参数上的置换问题.

定义 **3.1**. 假设有参数表 $Para_1$ 和 $Para_2$,置换$\sigma_2=\{V_1/U_1,V_2/U_2,\ldots,V_k/U_k\}$,若 $U(\sigma_2)\subseteq V(Para_2)\subseteq V(Para_1)$,则称按下面方法所得到的置换$\sigma_1$为置换$\sigma_2$在参数表 $Para_1$ 上的扩展置换:

(1) $\sigma_1\leftarrow\sigma_2$;

(2) 对$\forall P_j\in V(Para_1)-V(Para_2)$,若 $P_j\in U(\sigma_2)$,则$\sigma_1\leftarrow\{P'_j/P_j\}\cup\sigma_1$;

其中:P'_j是新构造出来的、具有唯一性的变量名,变换 P'_j/P_j 是对参数表 $Para_1$ 中的变量 P_j 进行换名操作,使之保持变换之前的互异特征.

例如:$Para_1=(?x_1\ ?x_2),Para_2=(?x_2),\sigma_2=\{?x_1/?x_2\},U(\sigma_2)=\{?x_2\}$.

于是有:$\sigma_2Para_1=(?x_1\ \ ?x_1)$.显然,置换$\sigma_2$破坏了参数表 $Para_1$ 中各参数之间的互异特征,所以,需要把置换σ_2扩展到参数表 $Para_1$.

由 $U(\sigma_2)\subseteq V(Para_2)\subseteq V(Para_1)$和定义 3.1 可得:置换$\sigma_2$的扩展置换$\sigma_1=\{?x_3/?x_1,?x_1/?x_2\}$.

所以有:$\sigma_1Para_1=(?x_3\ ?x_1)$.

引理 **3.1**. 假设有参数表 $Para_1$ 和 $Para_2$,置换σ_2,且 $U(\sigma_2)\subseteq V(Para_2)\subseteq V(Para_1)$,若置换$\sigma_1$是$\sigma_2$在 $Para_1$ 上的扩展置换,则$\sigma_1Para_2=\sigma_2Para_2$.

证明:

由定义 3.1 可知:$\sigma_1=\{P'_{j_1}/P_{j_1},P'_{j_2}/P_{j_2},\ldots,P'_{j_s}/P_{j_s}\}\cup\sigma_2,P_{j_i}\in V(Para_1)-V(Para_2),i=1..s$.

于是有:$\{P_{j_1},P_{j_2},\ldots,P_{j_s}\}\cap V(Para_2)=\varnothing$.

*** 在本文讨论谓词时,基本是以“正谓词”的形式来论述的,对“负谓词”的形式也同样有效,不再一一叙述.

任取一个变量参数 $Q_i \in V(Para_2)$.

显然,$Q_i \notin \{P_{j_1}, P_{j_2}, \ldots, P_{j_s}\}$,即:$Q_i$ 不会被 $\{P'_{j_1}/P_{j_1}, P'_{j_2}/P_{j_2}, \ldots, P'_{j_s}/P_{j_s}\}$ 中的变换所改变.

由定义 2.6 可知:$\sigma_1 Para_2 = (\{P'_{j_1}/P_{j_1}, P'_{j_2}/P_{j_2}, \ldots, P'_{j_s}/P_{j_s}\} \cup \sigma_2) Para_2 = \sigma_2 Para_2$.

所以,置换 σ_1 是 σ_2 在 $Para_1$ 上的扩展置换,则 $\sigma_1 Para_2 = \sigma_2 Para_2$. □

引理 3.1 说明了扩展所得到的置换与原置换对参数表 $Para_2$ 的作用是不变的.这种置换效果的不变性为效果谓词参数上的置换扩展到动作参数表上的置换提供了必要的理论准备.在此基础上,讨论从效果谓词参数表上的置换到动作参数表上置换的扩展问题以及这种扩展置换对动作内部信息的影响.

假设:待实现的谓词 Pd,存在动作 Act,$Pd_i \in E(Act)$,且 $Pd <\sim Pd_i$.

由定理 2.1 可知:一定存在置换 σ_2,使得:

$$Pd = \sigma_2 Pd_i,\ U(\sigma_2) \subseteq V(Para(Pd_i)) \tag{3.1}$$

由动作的描述可知:$V(Para(Pd_i)) \subseteq V(Para(Act))$.

所以,$U(\sigma_2) \subseteq V(Para(Pd_i)) \subseteq V(Para(Act))$.

由于效果谓词 Pd_i 的参数是通过动作参数的变换而变换的,它不能直接用置换 σ_2 来进行置换,所以,需要把置换 σ_2 扩展到参数表 $Para(Act)$上.

由于 $U(\sigma_2) \subseteq V(Para(Pd_1)) \subseteq V(Para(Act))$,可按定义 3.1 的方法把置换 σ_2 扩展到 $Para(Act)$上,得到扩展置换 σ_1. 动作 Act 中的置换如图 1 所示.

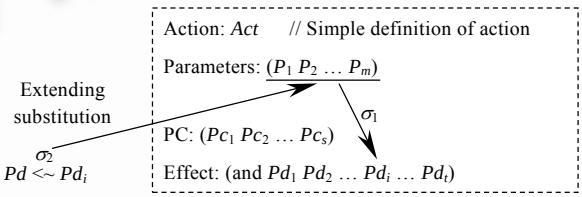

Fig.1 The meaning of Lemma 3.1 for action

图 1 引理 3.1 在动作中的含义

由引理 3.1、引理 2.1、式(3.1)和定义 2.4 可知: $\sigma_1 Para(Pd_i) = \sigma_2 Para(Pd_i) = Para(\sigma_2 Pd_i) = Para(Pd)$.

由动作描述的整体性可知:其效果谓词 Pd_i 应由置换 σ_1 来进行置换,而不是用置换 σ_2 来进行,即使两个置换对谓词 Pd_i 的置换效果一样.

定义 **3.2**. 假设有动作 Act 和置换 σ,σAct 表示用置换 σ 对动作 Act 进行置换后所得的动作形式.置换动作的具体规则如下:

(1) $Para(\sigma Act) = \sigma Para(Act)$;

(2) $\forall Pc \in PC(Act)$,有:$\sigma Pc \in PC(\sigma Act)$;

(3) $\forall Pd \in E(Act)$,有:$\sigma Pd \in E(\sigma Act)$.

假设,在 BlocksWorld 领域中有待实现的谓词 Pd=(*clear* ?*x*),在动作(*Unstack* ?*x* ?*y*)的效果中存在谓词 Pd_i=(*clear* ?*y*),且(*clear* ?*x*)<~(*clear* ?*y*),其中?*x* 和?*y* 都是同类的变量,代表任意一个积木.

令 $\sigma_2 = \{?x/?y\}$.

显然有:Pd=(*clear* ?*x*)=σ_2(*clear* ?*y*),且 $U(\sigma_2) = \{?y\} \subseteq V(Para(Pd_i))$.

按照定义 3.1 的方法,把置换 σ_2 扩展到动作参数(?*x* ?*y*)上,可得置换 $\sigma_1 = \{?x_1/?x, ?x/?y\}$.

于是,根据定义 3.2 可得:

σ_1(*Unstack* ?*x* ?*y*): (*Unstack* ?x_1 ?*x*)

σ_1PC: (*clear* ?x_1)(*on* ?x_1 ?*x*)(*arm-empty*)

σ_1Effect: (*holding* ?x_1)(*clear* ?*x*)(*not* (*on* ?x_1 ?*x*))(*not* (*clear* ?x_1))(*not* (*arm-empty*))

所以,当动作(*Unstack* ?x_1 ?*x*)的所有前提条件都满足时,执行该动作就可以实现谓词(*clear* ?*x*).

定理 **3.1**. 假设有谓词 Pd 和动作 Act,若$\exists Pd_i\in E(Act)$,且 $Pd<\sim Pd_i$,则存在置换σ,使得:$Pd\in E(\sigma Act)$.

证明:

由"$Pd<\sim Pd_i$"和定理 2.1 可知:存在置换σ_2,使得:$Pd=\sigma_2 Pd_i$,$U(\sigma_2)\subseteq V(Para(Pd_i))$.

由定义 2.4 和引理 2.1 可知:

$$Para(Pd)=Para(\sigma_2 Pd_i)=\sigma_2 Para(Pd_i) \tag{3.2}$$

由领域动作的描述规则可知:$V(Para(Pd_i))\subseteq V(Para(Act))$.

由 $U(\sigma_2)\subseteq V(Para(Pd_i))\subseteq V(Para(Act))$和定义 3.1 可知:把置换$\sigma_2$ 扩展到动作参数表 $Para(Act)$上,得到新的置换σ_1.

由引理 2.1 和引理 3.1 可知:

$$Para(\sigma_1 Pd_i)=\sigma_1 Para(Pd_i)=\sigma_2 Para(Pd_i) \tag{3.3}$$

由式(3.2)和式(3.3)可得:

$$Para(Pd)=Para(\sigma_1 Pd_i) \tag{3.4}$$

由 $Pd<\sim Pd_i$、定义 2.3 和定义 2.7 可得:$Type(Pd)=Type(\sigma_1 Pd_i)$,$Name(Pd)=Name(\sigma_1 Pd_i)$.

由式(3.4)和定义 2.4 可得:$Pd=\sigma_1 Pd_i$.

由"$Pd_i\in E(Act)$"和定义 2.2 可知:$\sigma_1 Pd_i\in E(\sigma_1 Act)$.

所以,$Pd\in E(\sigma_1 Act)$,即:置换σ_1 即为所求的置换σ,且 $Pd\in E(\sigma Act)$. □

对任意一个领域谓词 Pd,定理 3.1 说明了如何判断领域动作 Act 能否实现谓词 Pd.具体判断操作如下:

(1) 若存在 $Pd_i\in E(Act)$,有:$Pd<\sim Pd_i$,则一定存在置换σ,使得 $Pd\in E(\sigma Act)$,所以,在动作σAct 的前提条件 $PC(\sigma Act)$都得到满足时,执行动作σAct 就一定能实现谓词 Pd;

(2) 否则,一定不可能用动作 Act 及其变型来实现谓词 Pd.

下面分析判断谓词 Pd 能否用动作 Act 来实现所需要的时间复杂度.

任取 $Pd_i\in E(Act)$.

由定义 2.3 可知:判断"$Pd<\sim Pd_i$"所用的时间主要取决于判断"$Para(Pd)<\sim Para(Pd_i)$"所用的时间.

由定义 2.2 可知:判断"$Para(Pd)<\sim Para(Pd_i)$"最多所用的时间为 $O(|Para(Pd)|)$[****].

显然,在最坏情况下,谓词 Pd 需与 $E(Act)$中的每个谓词都要进行相似性判断.所以,在最坏情况下,共需用的时间为 $O(|E(Act)|\times|Para(Pd)|)$.

由于一个规划动作的效果谓词个数是较少的,一个谓词参数表的长度也是较短的,而且它们也都是固定的,所以,我们能够快速地(即在常数时间内)判断出一个规划动作能否实现一个指定的谓词.

4 规划领域知识的提取方法

在第 2 节,我们定义了谓词的相似性,并证明了两个相似谓词可以通过置换使它们变成同一个谓词形式.在第 3 节,根据动作定义的整体性特点,把动作内部谓词的置换扩展成动作参数上的置换,然后利用动作参数表上的置换来统一改变动作内部的所有信息,并证明了扩展的置换不会改变特定谓词的置换效果.

在规划领域中,动作的定义主要包括:动作参数、前提条件和效果.当动作前提条件满足且执行该动作时,就用动作的效果谓词来修改当前状态.由此可见,动作的前提条件和效果之间存在着一定的逻辑关系.

本节在规划领域的动作定义基础上,利用谓词的相似性来研究动作的前提条件和效果之间的逻辑关系,并从中提取出能够反映该领域内在本质的领域知识(或领域规则).

4.1 规划领域的基本规则

假设在一个规划领域的定义中,有 n 个动作 $Act_0,Act_1,\ldots,Act_{n-1}$.

对任意一个领域谓词 Pd,下面利用谓词的相似性来研究所有可实现谓词 Pd 的动作集.

[****] $|Para(Pd)|$表示谓词 Pd 参数表的长度.

定义 **4.1**. 对任意一个领域谓词 Pd,用 $Act(Pd)$来表示该规划领域中所有可实现谓词 Pd 的动作集:

$$Act(Pd)=\{Act_j \mid Act_j\in\{Act_0,Act_1,\ldots,Act_{n-1}\},且\exists Pd_i\in E(Act_j),使得:Pd<\sim Pd_i\}.$$

例如:对 BlocksWorld 领域中的谓词(*holding* ?*x*)和(*not* (*holding* ?*x*)),用定义 4.1 得到下面动作集*****:

$$Act((holding\ ?x))=\{Pickup,Unstack\},\ Act((not\ (holding\ ?x)))=\{Putdown,Stack\}.$$

定理 **4.1**. 对任意一个谓词 Pd,若 $Pd<\sim Pd_1$,则动作集 $Act(Pd_1)$中的动作经过置换后都可实现谓词 Pd.

证明:

在动作集 $Act(Pd_1)$中任取一个动作 Act_t.

由定义 4.1 可知:$\exists Pd_j\in E(Act_t)$,使得 $Pd_1<\sim Pd_j$.

由定理 3.1 可知:一定存在置换σ_t,使得 $Pd_1\in E(\sigma_t Act_t)$.

由"$Pd<\sim Pd_1$"和定理 3.1 可知:一定存在置换σ_s,使得 $Pd\in E(\sigma_s\sigma_t Act_t)$.

所以,执行动作$\sigma_s\sigma_t Act_t$就一定可以实现谓词 Pd. □

由定义 4.1 和定理 4.1 可知:在规划领域定义中,所有可实现谓词 Pd 的动作都在 $Act(Pd_1)$之中.

例如:在 BlocksWorld 领域中,假设当前待实现的谓词为(*clear A*).

$$Act((clear\ ?x))=\{Putdown,Unstack,Stack\}.$$

由"(*clear A*)<~(*clear ?x*)"和定理 4.1 可知:实现谓词(*clear A*)的 3 个动作是(*Putdown A*),(*Unstack* $?x_1$ *A*)和(*Stack A* $?y_1$).

定义 **4.2**. 假设 $Act(Pd)=\{Act_{j_1}, Act_{j_2},\ldots, Act_{j_k}\}$,$k$ 个置换 $\sigma_{j_1},\sigma_{j_2},\ldots,\sigma_{j_k}$,使得:$Pd\in E(\sigma_{j_i}Act_{j_i})$,$i=1..k$,那么,

(1) 若谓词 $Pd_i\in\cap_{i=1..k}E(\sigma_{j_i}Act_{j_i})-\{Pd\}$,则不论采用哪个动作,在实现谓词 Pd 的同时,Pd_i 也都同时被实现,称谓词Pd_i是实现谓词Pd的伴随结果,记为 $Pd_{(not\ Pd)}\rightarrow Pd_i$,其中,谓词(*not Pd*)为"真"是该伴随关系成立的条件;

(2) 若谓词 $Pc\in\cap_{i=1..k}PC(\sigma_{j_i}Act_{j_i})-\{(not\ Pd)\}$,则不论采用哪个动作实现谓词 Pd,都需要前提条件 Pc,称谓词 Pc 是实现谓词 Pd 的前提条件,或称谓词(*not Pc*)******为"真"时,阻碍谓词 Pd 的实现,记为:

$$(not\ Pc)_{(not\ Pd)}-\backslash->Pd,$$

其中,(*not Pd*)为"真"是该阻碍关系成立的条件.

关于定义 4.2 说明如下:

(1) $Pd_{(not\ Pd)}\rightarrow Pd_i$ 的含义有两种

- 当(*not Pd*)为"假"
  即 Pd 为"真",显然无须再用动作实现谓词 Pd,当然也就不存在伴随实现谓词 Pd_i 的问题,所以,在谓词(*not Pd*)为"假"时,伴随关系 $Pd_{(not\ Pd)}\rightarrow Pd_i$ 不起作用.
- 当(*not Pd*)为"真"
  即 Pd 为"假".当用动作实现谓词 Pd(使 Pd 为"真")时,谓词 Pd_i 才同时被实现.

(2) $(not\ Pc)_{(not\ Pd)}-\backslash->Pd$ 的含义有两种

- 当(*not Pd*)为"假"
  即 Pd 为"真",显然无须再用动作使谓词 Pd 为"真",当然也就不存在其他谓词阻碍谓词 Pd 的实现.
- 当(*not Pd*)为"真"
  即 Pd 为"假".当用动作实现谓词 Pd 时,就需要满足动作的前提条件 Pc,所以,谓词(*not Pc*)一定阻碍谓词 Pd 的实现.

在不引起混淆的情况下,谓词之间的伴随关系和阻碍关系可简记如下:

***** BlocksWorld 领域中实现谓词的动作集见附录 B.

****** 当 $Pc=(not\ (Name_1\ P_1\ P_2\ \ldots\ P_k))$时,$(not\ Pc)=(Name_1\ P_1\ P_2\ \ldots\ P_k)$.

$Pd_{(not\ Pd)}\rightarrow Pd_i$ 简记为 $Pd\rightarrow Pd_i$　　$(not\ Pc)_{(not\ Pd)}$–\–>Pd 简记为$(not\ Pc)$–\–>Pd

阻碍关系“$(not\ Pc)$–\–>Pd”说明:当用动作实现谓词 Pd 时,若谓词$(not\ Pc)$为“真”,则必须先通过动作(或动作序列),使谓词$(not\ Pc)$为“假”,即谓词 Pc 为“真”.

谓词之间的阻碍关系反映了用动作实现的谓词与动作前提条件之间的逻辑关系.在求解规划时,可利用谓词之间的阻碍关系来确定实现谓词的先后次序.

例如,在 BlocksWorld 领域中取下列 2 个动作集:

$$Act((on\ ?x\ ?y))=\{Stack\},\ Act((holding\ ?x))=\{Pickup,Unstack\}.$$

利用定义 4.2 分析上面两个动作集,可以得到相应的伴随关系和阻碍关系:

(1) $Act((on\ ?x\ ?y))$

由定义 4.1 和定理 3.1 可知:存在置换σ,使得$(on\ ?x\ ?y)\in E(\sigma Stack)$.

由定义 4.2 可得 4 个伴随关系、2 个阻碍关系:

$(on\ ?x\ ?y)\rightarrow(not\ (clear\ ?y))$　　$(on\ ?x\ ?y)\rightarrow(not\ (holding\ ?x))$

$(on\ ?x\ ?y)\rightarrow(arm\text{-}empty)$　　$(on\ ?x\ ?y)\rightarrow(clear\ ?x)$

$(not\ (clear\ ?y))$–\–>$(on\ ?x\ ?y)$　　$(not\ (holding\ ?x))$–\–>$(on\ ?x\ ?y)$

(2) $Act((holding\ ?x))$

由定义 4.1 和定理 3.1 可知:存在 2 个置换σ_1和σ_2,使得:

$$(holding\ ?x)\in E(\sigma_1 Pickup),(holding\ ?x)\in E(\sigma_2 Unstack).$$

由领域的动作定义可知:

$$E(\sigma_1 Pickup)\cap E(\sigma_2 Unstack)-\{(holding\ ?x)\}=\{(not\ (clear\ ?x)),(not\ (arm\text{-}empty))\},$$

$$PC(\sigma_1 Pickup)\cap PC(\sigma_2 Unstack)=\{(clear\ ?x),(arm\text{-}empty)\}.$$

由定义 3.2 可得 2 个伴随关系、2 个阻碍关系:

$(holding\ ?x)\rightarrow(not\ (clear\ ?x))$　　$(holding\ ?x)\rightarrow(not\ (arm\text{-}empty))$

$(not\ (clear\ ?x))$–\–>$(holding\ ?x)$　　$(not\ (arm\text{-}empty))$–\–>$(holding\ ?x)$

在后面的叙述中,我们称谓词之间的伴随关系为伴随规则,谓词之间的阻碍关系为阻碍规则;并统称伴随规则和阻碍规则为领域规则(或领域知识).BlocksWorld 领域的领域规则见附录 C.

4.2 规划领域基本规则的性质

在第 4.1 节中,利用实现谓词的动作集获取了规划领域的基本规则,这些基本规则反映了该规划领域中的领域知识,但这些领域规则都是以一种规则模式的形式出现的,它不代表某两个具体谓词之间的关系.比如,$(on\ ?x\ ?y)\rightarrow(not\ (clear\ ?y))$就是一个领域规则模式.对两个具体的谓词,如$(on\ A\ B)$和$(not\ (clear\ B))$,它们之间是否符合某规则的模式就需要进行判定.因此,该判定问题就是两个具体谓词与领域规则之间的模式匹配问题.

下面研究规划领域中具体谓词和领域规则之间的模式匹配问题以及领域规则所反映出来的谓词之间的特殊关系.

定义 **4.3**. 假设有谓词 Pd_i 和 Pd_j,领域规则 R 中的两个谓词为 Pd_1 和 Pd_2,若能通过“代入规则”,使得 $Pd_i=Pd_1$,$Pd_j=Pd_2$,那么,

(1) 若 $R=Pd_1\rightarrow Pd_2$,则称伴随规则“$Pd_i\rightarrow Pd_j$”是成立的;

(2) 若 $R=Pd_1$–/–>Pd_2,则称阻碍规则“Pd_i–/–>Pd_j”是成立的.

领域规则是谓词之间的一种规则模式.对于具体的谓词,若它们符合领域规则中规则 R 的模式,则这些具体谓词之间就具有规则 R 的关系.

例如:在 BlocksWorld 领域中,判断谓词$(on\ A\ B)$和$(not\ (clear\ B))$之间的关系.

在第 4.1 节中,通过对动作集 $Act((on\ ?x\ ?y))$的分析可得领域规则 R:$(on\ ?x\ ?y)\rightarrow(not\ (clear\ ?y))$.

对规则 R,令:$?x=A$,$?y=B$,那么,规则中的谓词就与待判断的两个谓词分别相等.

由定义 4.3 可知:伴随规则"(*on A B*)→(*not* (*clear B*))"是成立的.

同理可知:阻碍规则"(*not* (*clear B*))–\–>(*on A B*)"也是成立的.

定义 **4.4**. 假设规划领域中有谓词 Pd_i 和 Pd_j,

(1) 若 Pd_i→(*not* Pd_j),Pd_j→(*not* Pd_i),则称谓词 Pd_i 和 Pd_j 是互斥的;

(2) 若 Pd_i→(*not* Pd_j),(*not* Pd_i)→Pd_j,Pd_j→(*not* Pd_i),(*not* Pd_j)→Pd_i,则称谓词 Pd_i 和 Pd_j 是互补的.

假设在规划求解过程中,需要添加或删除谓词 Pd_i 和 Pd_j,那么,

(1) 谓词 Pd_i 和 Pd_j 的互斥性表示谓词 Pd_i 和 Pd_j 在初始状态中可以同时存在,不可以都不存在.但一旦删除其中之一后,在随后的规划状态中最多只有一个存在,可以两个都不存在;

(2) 谓词 Pd_i 和 Pd_j 的互补性表示谓词 Pd_i 和 Pd_j 在任何规划状态下都有且仅有一个存在.

由定义 4.4 可知:若谓词 Pd_i 和 Pd_j 是互补的,则谓词 Pd_i 和 Pd_j 是互斥的;但反之不然.

定理 **4.2**. 假设谓词 Pd_i 和 Pd_j 是互补的,若在规划求解过程中需增减谓词 Pd_i 或 Pd_j,则在初始状态中一定含有且仅含有两个互补谓词中的一个.

证明:

假设:在某个规划问题求解过程中需增减谓词 Pd_i 或 Pd_j,且状态变化序列为 S_0(Init),S_1,S_2,…,S_t(Goal).

由"谓词 Pd_i 和 Pd_j 是互补的"和定义 4.4 可知:存在下列 4 个伴随规则:

$$Pd_i\rightarrow(not\ Pd_j),\ (not\ Pd_i)\rightarrow Pd_j,\ Pd_j\rightarrow(not\ Pd_i),\ (not\ Pd_j)\rightarrow Pd_i.$$

(1) $Pd_i\in S_0$,$Pd_j\in S_0$

由已知条件可知:一定存在状态 S_u,使得 $Pd_i\in S_u$,$Pd_j\in S_u$,$Pd_i\notin S_{u+1}$ 或 $Pd_j\notin S_{u+1}$,$0\le u<t$.

再设:当从状态 S_u 变成状态 S_{u+1} 时,所执行的动作为 Act_v,且 $Pd_i\notin S_{u+1}$.

由"$Pd_i\in S_u$"和"(*not* Pd_i)→Pd_j"可知:在执行动作 Act_v 删除谓词 Pd_i 时,需要在当前状态 S_u 中添加谓词 Pd_j,使之变成下一状态 S_{u+1}.

而 $Pd_j\in S_u$,即谓词 Pd_j 已在状态 S_u 之中,这显然与规划领域动作的"添加效果"原义相矛盾.

(2) $Pd_i\notin S_0$,$Pd_j\notin S_0$

由已知条件可知:一定存在状态 S_u,使得 $Pd_i\notin S_u$,$Pd_j\notin S_u$,$Pd_i\in S_{u+1}$ 或 $Pd_j\in S_{u+1}$,$0\le u<t$.

再设:当从状态 S_u 变成状态 S_{u+1} 时,所执行的动作为 Act_v,且 $Pd_i\in S_{u+1}$.

由"$Pd_i\notin S_u$"和"Pd_i→(*not* Pd_j)"可知:在执行动作 Act_v 增加谓词 Pd_i 时,需要在当前状态 S_u 中删除谓词 Pd_j,使之变成下一状态 S_{u+1}.

而 $Pd_j\notin S_u$,即谓词 Pd_j 不在状态 S_u 之中,这显然与规划领域动作"删除效果"的原义相矛盾.

综合情况(1)和情况(2)可知:若在规划求解过程中需要改变谓词 Pd_i 或 Pd_j,则在初始状态中同时含有谓词 Pd_i 和 Pd_j,或不含谓词 Pd_i 和 Pd_j,都会得出矛盾之处.

所以,在初始状态中只能含有且仅含有谓词 Pd_i 和 Pd_j 其中的一个. □

假设:在初始状态中,存在两个互补的谓词 Pd_i 和 Pd_j.下面用图 2 来说明定理 4.2 的含义.

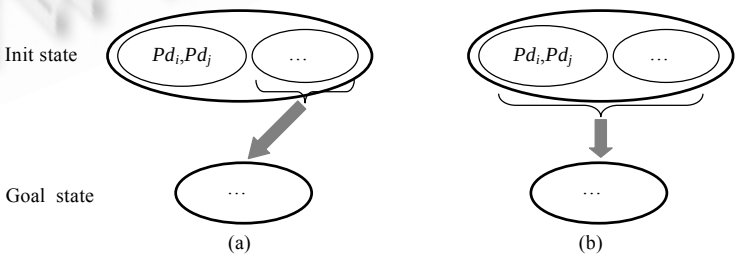

Fig.2 Two cases of complementary predications Pd_i and Pd_j in initial state

图 2 初始状态中存在互补谓词 Pd_i 和 Pd_j 的两种情况

(1) 若在规划求解时无须改变谓词 Pd_i 和 Pd_j,如图 2(a)所示,那么,目标状态不会因矛盾的初始状态而改变

其可解性.这表明对一个可实现的目标状态,不论在原初始状态中添加多少无关的矛盾谓词,该目标状态还是可实现的.

(2) 若在规划求解时需增减谓词 Pd_i 或 Pd_j,如图 2(b)所示,那么,初始状态中的互补谓词一定会影响目标状态的实现.这种情况正是定理 4.2 所描述的情况.

定理 **4.3**. 假设在目标状态中存在互斥谓词 Pd_i 和 Pd_j,若谓词 Pd_i 或 Pd_j 需用动作实现,则该目标状态一定是不可实现的.

证明:

假设:该目标状态是可实现的,且这些状态序列为 S_0(Init),S_1,S_2,…,S_t(Goal),$Pd_i \in S_t$,$Pd_j \in S_t$.

对状态序列 S_0,S_1,S_2,…,S_t 反向顺序查找状态 S_u,使得 $Pd_i \notin S_u$ 或 $Pd_j \notin S_u$,$Pd_i \in S_{u+1}$,$Pd_j \in S_{u+1}$.

(1) 在状态序列 S_0,S_1,S_2,…,S_t 中,找不到这样的状态 S_u

也就是说:$Pd_i \in S_k$,$Pd_j \in S_k$,k=0..t,即在整个规划求解的过程中,互斥谓词 Pd_i 或 Pd_j 从初始状态到目标状态都不会被改变,如图 3(a)所示,所以,它们都无须用动作来实现.这显然与已知条件"谓词 Pd_i 或 Pd_j 需用动作实现"相矛盾.

(2) 存在状态 S_u,满足:$Pd_i \notin S_u$ 或 $Pd_j \notin S_u$,$Pd_i \in S_{u+1}$,$Pd_j \in S_{u+1}$

不妨再设:$Pd_i \notin S_u$,且执行动作 Act_v,使得:当前状态 S_u 变成下一个状态 S_{u+1},如图 3(b)所示.

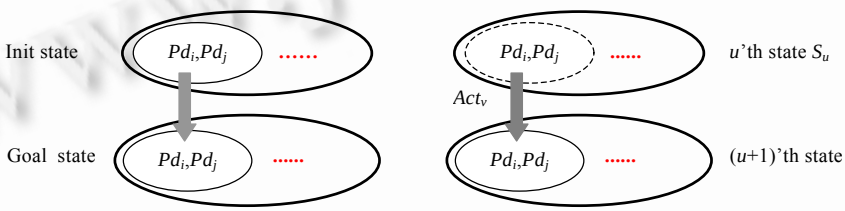

Fig.3 Two cases of mutually exclusive predications Pd_i and Pd_j in goal state

图 3 目标状态中存在互斥谓词 Pd_i 和 Pd_j 的两种情况

由"谓词 Pd_i 或 Pd_j 是互斥的"和定义 4.4 可知:$Pd_i \rightarrow (not\ Pd_j)$,$Pd_j \rightarrow (not\ Pd_i)$.

由"$\{Pd_i,Pd_j\} \subseteq S_{u+1}$"可知:动作 Act_v 把谓词 Pd_i 增加到当前状态 S_u 之中,使之变成状态 S_{u+1}.

由伴随规则"$Pd_i \rightarrow (not\ Pd_j)$"可知:用动作实现谓词 Pd_i 时,谓词 Pd_j 一定不在状态 S_{u+1} 之中.

这就与"$Pd_j \in S_{u+1}$"相矛盾.

综合情况(1)和情况(2)可得:不论在状态序列 S_0,S_1,S_2,…,S_t 中能否找到符合要求的状态,都可得出矛盾.

所以,若互斥谓词 Pd_i 或 Pd_j 需用动作实现,则包含它们的目标状态就一定是不可实现的. □

例如:在 BlocksWorld 领域中,有下面的伴随规则

(*arm-empty*)→(*not* (*holding* ?*x*)) (*not* (*arm-empty*))→(*holding* ?*x*)

(*holding* ?*x*)→(*not* (*arm-empty*)) (*not* (*holding* ?*x*))→(*arm-empty*)

由定义 4.4 可知:谓词(*arm-empty*)和(*holding A*)是互补的,其中,*A* 代表一个具体的积木块.

由定理 4.2 可知:谓词(*arm-empty*)和(*holding A*)只能有一个且仅有一个在初始状态之中,即初始状态中,要么"手空",要么"手中抓住某积木块",二者必具其一.

由定理 4.3 可知:谓词(*arm-empty*)和(*holding A*)一定不能同时在目标状态之中.也就是说,不可能同时实现"手空"和"手中抓住某具体的积木块".

由此可见,在领域基本规则的支持下,可用定理 4.2 来判断初始状态中的矛盾之处,也可用定理 4.3 来判断目标状态中的矛盾之处.在规划领域基本规则的基础上,我们还可以利用逻辑推理进一步得到更深层次的领域知识:间接阻碍关系和绝对阻碍关系等,并可用这些深层次的领域知识来指导相应的规划求解.

5 结 论

本文从领域的动作定义出发,利用谓词之间的相似关系以及规划动作的前提条件和效果之间的逻辑关系,自动提取出领域动作所隐含的领域知识.该自动提取策略对所有用 PDDL 语言描述的 STRIPS 领域都是有效的.该领域知识的自动提取策略在 Linux 环境下给予了实现,并对国际规划竞赛 IPC 所公布的所有基于 STRIPS 的基准规划领域(benchmark domain)都进行了测试,所提取出的领域规则都非常直观地表达了这些领域定义中所隐含的领域知识.

本文中的领域自动提取策略已应用于我们开发的"与领域无关的智能规划器——StepByStep"之中,所提取出来的领域规则可直接判断一些具体规划问题的可解性.在规划求解时,可用这些领域规则来决定谓词的选择策略,从而提高规划效率.有关规划器 StepByStep 的效率对比实验数据将另文介绍.

References:

[1] Fox M, Long D. The automatic inference of state invariants in TIM. Journal of AI Research, 1998,9:367−421.

[2] Cresswell S, Fox M, Long D. Extending TIM domain analysis to handle ADL constructs. In: McCluskey L, ed. Proc. of the AIPS 2002 Workshop on Knowledge Engineering Tools and Techniques for A.I. Planning. 2002.

[3] Blum A, Furst M. Fast planning through planning graph analysis. Artificial Intelligence, 1997,90:281−300.

[4] Thiébaux S, Hoffmann J, Nebel B. In defense of PDDL axioms. Artificial Intelligence, 2005,168:38−69.

[5] Bacchus F, Kabanza F. Using temporal logics to express search control knowledge for planning. Artificial Intelligence, 2000,116: 123−191.

[6] Bacchus F, Kabanza F. Planning for temporally extended goals. Annals of Mathematics and Artificial Intelligence, 1998,22:5−27.

[7] TLPlan.2006. http://www.cs.toronto.edu/~fbacchus/tlplan.html

[8] Kvarnström J, Doherty P. TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence (AMAI), 2001,30:119−169.

[9] Doherty P, Kvarnström J. TALplanner: A temporal logic based planner. AI Magazine, Fall Issue, 2001.

[10] TALPlan. 2006. http://www.ida.liu.se/~patdo/aiicssite1/kplab/projects/talplanner/index.html

[11] Hoffmann J. FF: The fast-forward planning system. AI Magazine, 2001,22(3):57−62.

[12] Hoffmann J, Nebel B. The FF planning system: Fast plan generation through heuristic search. Artificial Intelligence Research, 2001, 14:253−302.

[13] FF. 2006. http://www.mpi-sb.mpg.de/~hoffmann/ff.html

[14] Nguyen X, Kambhampati S, Nigenda RS. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. Artificial Intelligence Journal, 2002,135(1-2):73−123.

[15] Nigenda RS, Nguyen X, Kambhampati S. AltAlt: Combining the advantages of graphplan and heuristic state search. ASU Technical Report, 2000.

[16] AltAlt. 2006. http://rakaposhi.eas.asu.edu/altweb/altalt.html

[17] Kautz H, Selman B. Unifying SAT-based and graph-based planning. In: Dean T, ed. Proc. of the IJCAI '99. Morgan Kaufmann Publishers, 1999. 318−325.

[18] SATPLAN. 2006. http://www.cs.washington.edu/homes/kautz/satplan/

[19] Nilsson NJ. Artificial Intelligence: A New Synthesis. Morgan Kaufmann Publisher, 1999. 363−404.

[20] Kautz H, Selman B. BLACKBOX: A new approach to the application of theorem proving to problem solving. In: Proc. of the Aips'98 Workshop on Planning as Combinatorial Search. 1998. 58−60.

[21] Kautz H, Selman B. The role of domain-specific knowledge in the planning as satisfiability framework. In: Proc. of the 4th IJCAI. Menlo Park: AAAI Press, 1998. 181−189.

[22] Blackbox. 2006. http://www.cs.washington.edu/homes/kautz/satplan/blackbox/index.html

[23] Refanidis I, Vlahavas I. The GRT planning system: Backward heuristic construction in forward state-space planning. Artificial Intelligence Research, 2001,15:115−161.

[24] Refanidis I, Vlahavas I. GRT: A domain independent heuristic for strips worlds based on greedy regression tables. In: Proc. of the 5th European Conf. on Planning (ECP '99). Durham: Springer-Verlag, 1999. 346−358.

[25] GRT. 2006. http://macedonia.uom.gr/~yrefanid/GRT/index.html

[26] Edelkamp S, Helmert M. On the implementation of MIPS. In: Proc. of the Artificial Intelligence Planning and Scheduling (AIPS), Workshop on Decision-Theoretic Planning. 2000. 18−25.

[27] Edelkamp S, Helmert M. The model checking integrated planning system. AI-Magazine (AIMAG), 2001, 67−71.

[28] Lifschitz V. On the semantics of STRIPS. Reasoning about Actions and Plans, 1987,1−9.

[29] Mcdermott D. PDDL—The planning domain definition language. Technical Report, 1998.

[30] Fox M, Long D. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. Technical Report. University of Durham, 2002.

附录 A BlocksWorld 领域的描述

(define (domain BlocksWorld)
(:requirements :strips)
(:predicates (*clear* ?*x*)(*on-table* ?*x*)(*arm-empty*)(*holding* ?*x*)(*on* ?*x* ?*y*))
(:action *Pickup*
:parameters (?*x*)
:precondition (and (*clear* ?*x*)(*on-table* ?*x*)(*arm-empty*))
:effect (and (*not* (*clear* ?*x*))(*not* (*on-table* ?*x*))(*not* (*arm-empty*))(*holding* ?*x*)))
(:action *Putdown*
:parameters (?*x*)
:precondition (*holding* ?*x*)
:effect (and (*not* (*holding* ?*x*))(*clear* ?*x*)(*on-table* ?*x*)(*arm-empty*)))
(:action *Unstack*
:parameters (?*x* ?*y*)
:precondition (and (*clear* ?*x*)(*on* ?*x* ?*y*) (*arm-empty*))
:effect (and (*not* (*clear* ?*x*))(*not* (*on* ?*x* ?*y*))(*not* (*arm-empty*))(*holding* ?*x*)(*clear* ?*y*)))
(:action *Stack*
:parameters (?*x* ?*y*)
:precondition (and (*clear* ?*y*)(*holding* ?*x*))
:effect (and (*not* (*clear* ?*y*))(*not* (*holding* ?*x*))(*clear* ?*x*)(*on* ?*x* ?*y*)(*arm-empty*)))
)

附录 B BlocksWorld 领域中实现谓词的动作集

对规划领域 BlocksWorld,由定义 4.1 可得下列 10 个动作集:

Act((*on* ?*x* ?*y*))={*Stack*}
Act((*ontable* ?*x*))={*Putdown*}
Act((*clear* ?*x*))={*Putdown*, *Unstack*, *Stack*}
Act((*arm-empty*))={*Pickup*, *Stack*}
Act((*holding* ?*x*))={*Pickup*, *Unstack*}
Act((*not* (*on* ?*x* ?*y*)))={*Unstack*}
Act((*not* (*ontable* ?*x*)))={*Pickup*}
Act((*not* (*clear* ?*x*)))={*Pickup*, *Unstack*, *Stack*}
Act((*not* (*arm-empty*)))={*Pickup*, *Unstack*}
Act((*not* (*holding* ?*x*)))={*Putdown*, *Stack*}

附录 C BlocksWorld 领域的基本规则

利用定理 4.1 和定义 4.2,分析所有效果谓词动作集,共得到下列 22 个伴随规则和 11 个阻碍规则:

1. 规划领域的伴随规则(22 个伴随规则)

(*on* ?*x* ?*y*)→(*not* (*clear* ?*y*))
(*on* ?*x* ?*y*)→(*arm-empty*)
(*not* (*on* ?*x* ?*y*))→(*not* (*arm-empty*))
(*not* (*on* ?*x* ?*y*))→(*clear* ?*y*)
(*ontable* ?*x*)→(*not* (*holding* ?*x*))
(*ontable* ?*x*)→(*clear* ?*x*)
(*not* (*ontable* ?*x*))→(*not* (*arm-empty*))
(*arm-empty*)→(*not* (*holding* ?*x*))
(*not* (*arm-empty*))→(*not* (*clear* ?*x*))
(*holding* ?*x*)→(*not* (*arm-empty*))
(*not* (*holding* ?*x*))→(*arm-empty*)
(*on* ?*x* ?*y*)→(*not* (*holding* ?*x*))
(*on* ?*x* ?*y*)→(*clear* ?*x*)
(*not* (*on* ?*x* ?*y*))→(*not* (*clear* ?*x*))
(*not* (*on* ?*x* ?*y*))→(*holding* ?*x*)
(*ontable* ?*x*)→(*arm-empty*)
(*not* (*ontable* ?*x*))→(*holding* ?*x*)
(*not* (*ontable* ?*x*))→(*not* (*clear* ?*x*))
(*arm-empty*)→(*clear* ?*x*)
(*not* (*arm-empty*))→(*holding* ?*x*)
(*holding* ?*x*)→(*not* (*clear* ?*x*))
(*not* (*holding* ?*x*))→(*clear* ?*x*)

2. 规划领域的阻碍规则(11 个阻碍规则)

(*not* (*clear* ?*y*))–\–>(*on* ?*x* ?*y*)
(*not* (*arm-empty*))–\–>(*not* (*on* ?*x* ?*y*))
(*not* (*holding* ?*x*))–\–>(*ontable* ?*x*)
(*not* (*arm-empty*))–\–>(*not* (*ontable* ?*x*))
(*not* (*holding* ?*x*))–\–>(*arm-empty*)
(*not* (*arm-empty*))–\–>(*holding* ?*x*)
(*not* (*holding* ?*x*))–\–>(*on* ?*x* ?*y*)
(*not* (*clear* ?*x*))–\–>(*not* (*on* ?*x* ?*y*))
(*not* (*clear* ?*x*))–\–>(*not* (*ontable* ?*x*))
(*not* (*clear* ?*x*))–\–>(*not* (*arm-empty*))
(*not* (*clear* ?*x*))–\–>(*holding* ?*x*)

吴向军(1965－),男,副教授,主要研究领域为人工智能,算法设计,网络应用.

姜云飞(1945－),男,教授,博士生导师,CCF 高级会员,主要研究领域为自动推理,智能规划和基于模型诊断.

凌应标(1965－),男,博士,主要研究领域为智能规划,演化计算,智能优化.