规划的形式表示技术研究*)

陈蔼祥[1,2] 姜云飞[2] 柴啸龙[1,2]
(广东商学院数学系 广州 510320)[1] (中山大学软件研究所 广州 510275)[2]

摘 要 智能规划是人工智能研究领域中的一个富有挑战性的课题。同时,智能规划问题具有广泛的实际应用背景。而自动规划问题面临的首要问题是规划问题的形式表示问题。本文系统地分析了情景演算,STRIPS,PDDL,HTN等各种规划形式表示技术,比较了各种形式表示技术的优缺点,并指出了进一步研究设计高效智能规划系统的方向。

关键词 智能规划,形式表示,情景演算,STRIPS,PDDL,HTN

Research on the Formal Representation of Planning Problem

CHEN Ai-xiang[1,2] JIANG Yun-fei[2] CAI Xiao-long[1,2]
(Department of Mathematics, Guangdong University of Business Studies, Guangzhou 510320, China)[1]
(Institute of Software, Sun Yat-Sen University, Guangzhou 510275, China)[2]

Abstract Intelligent planning is a challenge topic on the field of artificial intelligence, and it has broad application background. The first problem in automated planning is how to formally represent the realistic planning problem. In this paper, we systematically analysis several kind of formally representation technologies such as situation calculus, STRIPS, PDDL, HTN technology, systematically discuss how to characterize the realistic planning problem formally and point out further direction to improve the current automated planning systems.

Keywords Intelligent planning, Formal representation, Situation calculus, STRIPS, PDDL, HTN

1 引言

智能规划研究的主要目的是建立高效实用的智能规划系统。该系统的主要功能可以描述为:给定问题的状态描述、对状态描述进行变换的一组操作、初始状态和目标状态,要求智能规划系统能够自动给出从初始状态到达目标状态的一个动作序列。因此,对于智能规划的研究,首先需要解决一个所谓的知识表达的问题,即需要有一种精确严密的形式语言,以方便我们对实际的规划问题进行有效建模。

一阶逻辑[1]是最早用于规划描述的语言,并且应用推理方法求解规划问题。即使到现在,它仍是很重要的规划描述语言。情景演算[2]是一种表示动态世界变化的二阶逻辑语言,是早期的规划形式表示方法。此后,Nillson等人提出了STRIPS[3]的规划描述方法,改变了原来纯逻辑框架下的规划研究手段。在STRIPS方法的影响下,规划研究领域逐步形成了一种统一的规划描述语言PDDL[4,5],PDDL语言是目前应用得最广泛的语言,并且该语言仍然在进一步扩充之中[6]。另外,为了有效地提高规划求解的效率,有些研究者从任务分解的角度对规划问题进行研究,并形成了一种称为层次任务分解(HTN)[7]的规划描述语言。

本文第2部分阐述规划问题的基本组成要素,第3部分介绍规划的各种形式化描述方法,包括情景演算方法,STRIPS方法,PDDL描述方法和HTN方法。从中我们可以看到,各种不同的描述方法的差别主要体现在它们的规划问题的理解角度不同而导致它们对规划的基本组成要素的不同定义。最后,我们对各种规划技术的优缺点进行了总结对比,并指出了进一步研究设计高效智能规划系统的方向。

2 规划问题的基本组成要素

通常,规划是指给定问题的初始状态和目标状态,以及规划动作的描述,要求能自动找到一动作序列,使系统能从初始状态转换到目标状态。因此,对于一个规划问题,其基本要素有三个:状态集合,动作集合,初始状态和目标状态。状态的描述通常涉及类似机器人所处的位置,集装箱的堆放情况以及各种物理世界的描述。所有各种不同的状态就构成状态空间,状态空间既可以是离散的(有限或可数无限),也可以是连续的(不可数无限)。在大多数应用场合,状态空间的规模通常都是非常巨大而难以进行显式描述。对状态的定义是规划形式描述的一个重要的组成部分,并且直接影响到相关的求解算法的设计和分析。通常,动作是要作用于状态的,动作描述的内容表明当动作作用于某一状态时,状态将会发生怎样的变化。因此,可将动作看作是一个状态转换函数。一般情况下,要求一个规划问题中所涉及的动作应是简单的,容易对其进行描述的,但又难以且没必要对所有动作采用穷举的办法一一列举出来。例如,对于一个积木世界,我们只需要unstack,stack,putdown,pickup这四个基本动作就足够了,显然这四个基本动作是简单而且易描述的,对于不同的积木世界问题,我们只需要对这四个基本动作进行实例化,即可得到我们所需要的实例化动作。最后,对于一个规划问题,还必须给出一个初始状态以及规划最终必须达到的目标状态。

一般地,对于一个规划问题,我们可用一个三元组$P=(\Sigma,s_0,S_g)$的形式进行表示。其中$\Sigma=(S,A,\gamma)$为规划领域,S为状态集合,A为动作集合,γ为确定型状态转换函数,s_0为系统初始状态,S_g为规划所要达到的目标状态。

*)基金项目:国家自然科学基金(项目编号:60173039)。陈蔼祥 博士,主要研究方向为智能规划与诊断;姜云飞 教授,博士生导师,主要研究方向为自动推理、智能规划和基于模型的诊断等;柴啸龙 博士研究生,讲师,主要研究方向为智能规划与诊断。

3 规划的形式表示方法

规划的形式表示方法主要有情景演算方法[2]，STRIPS[3]方法，PDDL[4-6]方法以及 HTN[7]方法。各种表示方法的差别主要在于对规划的状态，规划的动作以及规划的初始状态和目标状态这三个基本要素的不同表示。这些不同的表示方法的提出，要么是出于提高规划效率的目的，要么是为增强规划的表示能力。例如，对非经典的智能规划的表示，就需要在经典规划的基础上增加对不确定因素、时序逻辑的目标优化要求等信息的表示。

3.1 情景演算表示方法

情景演算是一种表示动态世界变化的二阶逻辑语言。在情景演算中，世界的状态用情景表示，所谓情景即是一阶逻辑项的集合，初始情景用 s_0 表示。所有情景的变化都是由于使用某一动作的结果，$do(\alpha,s)$ 表示将动作 α 作用于情景 s 的后续情景。而动作则是带有参数的，比如 $put(x,y)$ 表示将物体 x 拿起来放在物体 y 上，$do(put(A,B),s)$ 表示在当前情景 s 下，将 A 放到 B 上后导致的后续情景，$do(putdown(A),do(walk(L),do(pickup(A),s_0)))$ 表示从初始情景 s_0 出发，执行动作序列[pick(A)，walk(L)，putdown(A)]后的结果情景。

前提条件公理

显然，对于某一动作 α 和某一情景 s 而言，动作 α 必须满足一定的条件才能将其作用于情景 s，也就是说，动作必须满足一定的前提条件。在情景演算中，动作的前提条件被表示成前提条件公理。为表示动作的前提条件公理，需要引入谓词符号 Poss，$Poss(\alpha,s)$ 表示在情景 s 下，动作 α 是可执行的。例如 $Poss(pickup(r,x),s)\supset[\forall z\neg holding(r,z,s)]\wedge\neg heavy(x)\wedge nextTo(r,x,s)$ 表示，在情景 s 下，动作 $pickup(r,x)$ 可执行的前提条件是，在情景 s 下机器人 r 手中没有拿任何其他物体，同时，机器人要在物体 x 旁边，并且物体 x 是不重的。

效果公理

将动作作用于某一情景后将导致情景的转换。在情景演算中，动作的这一效果是通过所谓的效果公理来表示的。例如，$fragile(x,s)\subset broken(x,do(drop(r,x),s))$ 表示一个易碎的物体从机器手上掉下来后，该物体将会被打碎。

这样，我们就可以实现对规划问题中的初始情景，目标情景以及动作进行公理化描述。相应的规划问题就转化为在给定初始情景和目标情景的公理化描述的基础上，要求找到某一动作序列，使得系统能从初始情景最终到达目标情景。由于情景演算表示方法是一种在纯逻辑框架下的表示方法，因此，在情景演算表示方法下，规划的产生过程可被看作是一个定理证明的过程。因此，对于一个规划问题，情景演算方法除了要对通常的规划三要素进行公理化表示外，还必须对所谓的资历问题、派生问题以及框架问题进行公理化表示。对这一部分内容本文不打算详细介绍，有兴趣的读者可参考相关文献。下面仅给出一种用情景演算表示规划问题的例子。

例 1：假定有一单臂机器人 R，每次机器手最多只能抓一个物体，该机械手可以抓起(pickup)或放下(drop)某一物体，机器人可以走动(walk)。现有两个物体 A,B 分别在不同的位置，并且机器人 R 在机器手为空的情况下位于靠近物体 A 的位置上。现要求机器人 R 将物体 A 放到物体 B 所处位置的地面上。

对于上述规划问题，我们可以对其用情景演算方法进行表示。首先，我们给出规划问题的初始情景和目标情景。

初始情景：

$robot(R),nextTo(R,A,S0),\forall z\neg holding(R,z,S0)$

目标情景：

$nextTo(A,B,Sg)\wedge onfloor(A,Sg)$

动作的前提条件公理可表示成下列形式：

$Poss(pickup(r,x),s)\equiv robot(r)\wedge[(\forall z)\neg holding(R,z,s)]\wedge nextTo(r,x,s)$，

$Poss(wald(r,y),s)\equiv robot(r)$，

$Poss(drop(r,x),s)\equiv robot(r)\wedge holding(r,x,s)$。

动作的效果公理可表示成下列形式：

$holding(r,x,do(pickup(r,x),s))$，$\neg holding(r,x,do(drop(r,x),s))$，$nextTo(r,y,do(walk(r,y),s))$，$nextTo(r,y,s)\supset nextTo(x,y,do(drop(r,x),s),y\neq x\supset\neg nextTo(r,x,do(walk(r,y),s))$，$onfloor(x,do(drop(r,x),s))$，$\neg onfloor(x,do(pickup(r,x),s))$。

另外，我们还需要一些解决有关框架问题的后续情景公理(successor state axioms)以及动作名称唯一性公理(unique names axioms for actions)。

后续情景公理：

$$Holding(r,x,do(\alpha,s))\equiv\alpha=pickup(r,x)\vee holding(r,x,s)\wedge\alpha\neq drop(r,x) \quad (1)$$

$$nextTo(x,y,do(\alpha,s))\equiv\alpha=walk(x,y)\vee(\exists r)[nextTo(r,y,s)\wedge\alpha=drop(r,x)]\vee nextTo(x,y,s)\wedge(\neg\exists z)[\alpha=walk(x,z)\wedge z\neq y] \quad (2)$$

$$onfloor(x,do(\alpha,s))\equiv(\exists r)\alpha=drop(r,x)\vee onfloor(x,s)\wedge\neg(\exists r)\alpha=pickup(r,x) \quad (3)$$

动作名称唯一性公理：

$Pickup(r,x)\neq drop(r',y)$，$pickup(r,x)\neq walk(r',y)$，$Walk(r,y)=walk(r',y')\supset r=r'\wedge y=y'$等等。

因此，根据上述公理，我们可以通过逻辑推理得到一些相关事实：

根据公式(1)，我们有：

$$Holding(R,A,do(pickup(R,A),S0)) \quad (4)$$

根据公式(4)，(1)以及动作名称唯一性公理，我们有：

$$Holding(R,A,do(walk(R,y),do(pickup(R,A),S0))) \quad (5)$$

根据公式(2)，我们有：

$$nextTo(R,y,do(walk(R,y)do(pickup(R,A),S0))) \quad (6)$$

根据公式(6)，(2)，我们有：

$$nextTo(A,y,do(drop(R,A),do(walk(R,y),do(pickup(R,A),S0)))) \quad (7)$$

根据公式(3)，我们有：

$$Onfloor(A,do(drop(R,A),do(walk(R,y),do(pickup(R,A),S0)))) \quad (8)$$

显然，根据规划问题的目标情景，我们希望能通过推理获得以下事实：

$$(\exists s)nextTo(A,B,s)\wedge onfloor(A,s) \quad (9)$$

即希望能够找到一个情景 s 能使目标情景中的命题都成立。换句话说，就是希望能找到一个动作序列，使得物体 A 在物体 B 旁边的地板上。显然，公式(4)-(8)为公式(9)中句子所代表的目标情景提供了一个构造性的证明，也就是说，我们可以找到这样的情景 s 满足公式(9)：

$s=do(drop(R,A),do(walk(R,B),do(pickup(R,A),S0)))$

我们将上述情景项 s 解释为将物体 A 移到物体 B 旁边的地板上的规划：首先，机器人 R 抓起物体 A，然后，机器人走向 B，最后，机器人将物体 A 放下。

这样，经过上述情景演算方法将规划问题进行公理化描述后，规划的产生过程就转化为定理的证明过程，即根据一些基本公理，证明存在某一情景能满足目标情景中的命题：

Axions $\vdash(\exists s)G(s)$

最后我们给出有关情景演算表示方法中对规划的严格定义。

定义 1 令 D 为规划领域的公理化描述，σ 为不含变量的情景项，$G(s)$ 为一除 s 为自由变量外，不含其他自由变量的情景公式。那么 σ 为 G 的规划(关于 D)当且仅当：

$D\vdash \text{executable}(\sigma)\wedge G(\sigma)$

一种求解 σ 的方法是调用定理证明器证明 $(\exists s)\text{executable}(s)\wedge G(s)$。定理证明过程中，任意对 s 进行绑定都将保证能产生可执行的规划，并最终能产生满足目标 G 的情景。

情景演算方法是在纯逻辑的框架下对规划问题进行研究。其好处在于，用逻辑语言对规划问题进行描述，具有很强的表达能力。但情景演算的不足之处在于，除需要对实际规划问题进行公理化描述，同时，还要解决有关资历问题、派生问题以及框架问题，这给实际规划问题的形式化描述带来了困难。并且，在情景演算方法下，规划的求解过程效率不高，因为规划的产生过程实质上是一个定理证明的过程。因此，规划研究者们又尝试采用其他规划描述方法。

3.2 STRIPS 方法

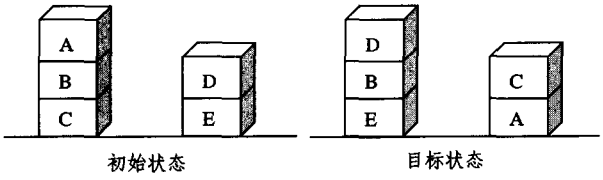

图 1

STRIPS 方法是以“Stanford Research Institute Planning System”这个学术机构的名字来命名的。在 STRIPS 方法中，状态用一阶语言表示的逻辑原子集合来表示，因此，规划的初始状态和目标状态都被表示成合式公式的形式，这些与情景演算方法类似。不同的是，STRIPS 方法中，规划动作被表示成一个三元组的形式，即 $o=\langle \text{pre}(o),\text{add}(o),\text{del}(o)\rangle$，其中，$\text{Pre}(o)$ 表示动作 o 的前提条件集合，$\text{Add}(o)$ 表示动作 o 的添加效果集合，$\text{Del}(o)$ 表示动作 o 的删除效果集合。这样，对于某一动作和某一状态而言，当动作的所有前提条件在该状态中全部为真时，称这个动作相对于这个状态是可用的。当一个可用动作作用于某一状态后，动作的效果可被分为两类：添加效果和删除效果。添加效果会在原来的状态基础上增加一些新的事实命题，而删除效果则在原来状态的基础上删除某些事实命题。此时，一个规划问题的求解被定义为在某一状态空间中搜索特定节点的问题，即要求搜索某一动作序列，使得系统从初始状态转换到目标状态。下面以积木块世界为例，给出规划问题的 STRIPS 描述形式。

例 2：假定积木块问题的初始状态和目标状态分别如图 1 中所示，该规划问题中有三个动作，分别为 move(X,Y,Z)，moveFromTable(X,Y) 以及 moveToTable(X,Y)。现用 STRIPS 方法对该规划问题进行形式化描述。

首先，初始状态可用下列命题公式集合进行表示：

{onTable(c), on(B,C), on(A,B), onTable(E), on(D,E), clear(A), clear(D)}

目标状态可用下列命题公式集合进行表示：

{ontable(E), on(B,E), on(D,B), onTable(A), on(C,A), clear(D), clear(C)}

对规划动作分别描述如下：

move(X,Y,Z)

Preconds: clear(X), clear(Z), on(X,Y), $X\neq Z$

Adds: clear(Y), on(X,Z)

Dels: clear(Z), on(X,Y)

moveFromTable(X,Y)

Preconds: clear(X), clear(Y), onTable(X), $X\neq Y$

Adds: on(X,Y)

Dels: clear(Y), onTable(X)

moveToTable(X,Y)

Preconds: clear(X), on(X,Y)

Adds: clear(Y), onTable(X)

Dels: on(X,Y)

与情景演算方法相比，STRIPS 方法的不同之处在于：

• STRIPS 方法不需要对动作的前提条件和动作的效果进行公理化描述；

• STRIPS 方法中，对动作的描述不再是逻辑子句的形式，而是以前提条件表，添加效果表和删除效果表的形式对动作进行描述；

• 由于动作描述中使用了添加效果和删除效果这种机制，因此，STRIPS 方法可以避免情景演算中大量的框架公理问题；

• STRIPS 方法下，规划的求解问题本质上是在一个状态空间中搜索目标状态的搜索问题，而情景演算方法下，规划求解问题是一个定理证明的过程。

Fikes 和 Nilsson 设计的 STRIPS 系统[9]在智能规划的研究中具有重大的意义和价值，他的突出贡献是引入了 STRIPS 操作符的概念，改变了原来局限于纯逻辑框架下对规划问题进行研究的情况，使得原来很神秘的规划问题求解变得明朗清晰起来。为过程化的规划方法奠定了坚实的基础。此后，在 STRIPS 方法的影响下，规划研究取得了很大进展，并且最终逐渐形成一种统一的规划表示语言，即 PDDL 语言。

3.3 PDDL 表示方法

PDDL 是规划领域描述语言(Planning Domain Description Language)的简称，是作为智能规划问题设计的标准语言。最早的版本是由 McDermott 定义，用在 1998 年规划大赛上，目前最新版本是 PDDL3.0。其中，PDDL2.1 是对原由 STRIPS 描述的一个较大突破，PDDL2.2 和 PDDL3.0 分别是在 PDDL2.1 框架下的扩展。

由于受文章篇幅所限，本文不可能也没必要对 PDDL 系列语言进行详细介绍。下面，为让读者能在不必拘泥于 PDDL 语言严格的语法和语义细节的情况下，就能对 PDDL 语言有一个直观的了解，本文先给出一个简单的规划问题，看我们如何对这种规划问题用 PDDL 语言进行表示，然后我们简要介绍 PDDL 语言的基本特点，以及其扩展语言 PDDL2.1，PDDL2.2 和 PDDL3.0 的相关特点。对规划表示语言感兴趣的读者，可进一步参考相关文献。

PDDL 语言

为使描述尽量简洁，本文就以图 1 积木块世界的规划问题为例，看我们如何用 PDDL 语言对积木块世界进行描述。一般情况下，用 PDDL 语言对规划问题进行描述分为两部分，一部分为领域描述部分 domain. pddl，一部分是问题描述部分 fct. pddl，比如，对积木块世界的领域描述记为 bw-domain. pddl，则 bw-domain. pddl 内容如下所示：

```
bw-domain. pddl
(define (domain bw)                                          // 领域名称
(:requirements:strips)                                       // 所能处理的问题类型
(:predicates (on ? x ? y)(on-table ? x)                      //领域中所涉及的谓词
             (clear ? x)(arm-empty)(holding ? x)
)
(:action pick-up
    :parameters (? ob1)
    :precondition (and (clear ? ob1)
                       (on-table ? ob1)
                       (arm-empty)
                  )
    :effect (and (not (on-table ? ob1))
                 (not (clear ? ob1))
                 (not (arm-empty))
                 (holding ? ob1)
                 )
)
(:action stack
    :parameters (? sob ? sunderob)
:precondition (and (holding ? sob)
                   (clear ? sunderob)
                 )

:effect (and (not (holding ? sob))
                   (not (clear ? sunderob))
                   (clear ? sob)
                   (arm-empty)
                   (on ? sob ? sunderob)
                 )
)
)
```

对图 1 中规划问题描述文件 bw-fct. pddl 可表示成下列形式：

```
Bw-fct. pddl
(define (problem bw-fact)                    // 规划问题名称
(:domain bw)                                 // 规划领域名称
(:objects A B C D E)                         // 规划问题中所有对象变量
(:init (clear A) (arm-empty)                 // 规划问题初始状态
    (on A B) (on B C)(on-table C)
    (clear D)(on D E)(on-table E)
)
(:goal (and (clear D) (on D B)               / 规划问题目标状态
         (on B E)(on-table E)
        (clear C)(on C A)(on-table A)
      )
)
)
```

对于支持 PDDL 语言的规划器，只要将规划领域描述文件，比如 bw-domain. pddl，以及问题描述文件，比如 bw-fct. pddl，输入规划器，则规划器就会对相应规划问题进行求解，最终返回规划解。

PDDL 是作为智能规划问题设计的标准语言。最早的版本是由 McDermott 定义用在 1998 年规划大赛上，它极大地丰富了原来的 STRIPS 语言。PDDL 语言的主要特点有以下几个方面：

- 描述基本 STRIPS 风格的动作
- 描述条件效果
- 支持动态世界的全称量词
- 描述分层理论中的领域公理
- 描述安全约束
- 描述层次动作和层次目标
- 支持多领域多问题的多种不同特点的语言子集

有关 PDDL 语言的这些特点的详细描述，这里不打算进行深入介绍，感兴趣的读者，可参考有关的 PDDL 语言手册。

PDDL2. 1

PDDL2. 1 主要是在原来 PDDL1. 0 版本的基础上，对其进行扩充增加了时序逻辑的内容，即考虑了动作延续时间的描述和时间流逝对动作的影响。PDDL2. 1 包括五个层次：

第一层：对应于 PDDL 的命题和 ADL 层，为了与以前的语言标准兼容，没有做任何的增改。

第二层：在第一层的基础上增加了数值变量，并且可以在瞬间访问和修改它们的值。

第三层、第四层：在第二层基础上增加了持续性动作，即动作的效果发生在动作开始执行后的一段时间，不同的是，第三层的持续性动作在时间上是离散的，而第四层的持续性动作可以有连续的效果。因为持续性动作只限制对它们所影响的变量的状态的访问，所以第四层实际上是简化了实时领域的模型。

第五层：是第四层的一种自然扩充，表示了一种富有表达力的规划领域描述语言的设计方式，可以表示任意的、离散时间和连续时间相混合的领域。

PDDL2. 2

PDDL2. 2 是用在 2004 年规划比赛上的描述语言，它增加了派生谓词(Derived Predicates)，和在初始状态里带时间的谓词描述。派生谓词可以在原有规划搜索上，在状态判断是否实现了目标，再多做一层推理。带时间的谓词描述是指在某个时间点上，谓词才会得到满足，这相当于增强了时间的约束。

PDDL3. 0

PDDL3. 0 是在 2006 年规划比赛的描述语言，其主要的新特点是在 PDDL2. 2 的基础上，增加了对所谓的“软目标”(soft goals)和“状态轨迹约束”(state trajectory constraints)的描述。软目标是指对于合法规划解不一定必然会达到的期望目标。状态轨迹约束主要是指规划结构中的约束，用来表示规划领域和/或规划问题的合法规划解必须满足的某些控制知识或限制。状态轨迹约束同样有“硬约束”和“软约束”之分。软目标和状态轨迹约束都是用来表示某种偏好，而这种偏好通常都会反映到对规划解的质量的评价。

有关 PDDL 系列语言的详细描述，以及各种版本的 PDDL 语言的详细差异，感兴趣的读者请自行参考相关文献，本文只在此作一简单介绍，而不打算做深入的分析和探讨。因为本文在此讨论 PDDL 语言的目的仅是从规划问题的知识表示的角度进行介绍，希望能帮助读者了解如何对规划问题建模，以及目前都有哪些规划描述方法。

3.4 HTN 语言

所谓 HTN 规划表示方法是指分层任务网络规划(Hier-

archical Task Network Planning)方法的简称。智能规划的任务是要自动找到能解决给定规划任务的动作序列。已有证明,领域无关的规划问题是 PSPACE 完全或更难[13,14]。因此,为了提高规划求解的效率,研究者们提出了各种规划方法。HTN 规划就是一种将任务分解的思想应用于规划求解,以期提高规划求解效率的方法。HTN 规划方法由于使用了任务分解的思想,因此该方法下关于规划问题的形式表示与其他规划方法有所不同,因此,为叙述方便起见,我们约定将 HTN 规划方法下的规划表示语言称为 HTN 语言。

例 3:如图 2 所示的规划问题,要求将一堆集装箱在不改变其堆放顺序的前提下,从位置 $p1$ 移到位置 $p3$。也就是说,对于每一集装箱,在移动前和移动后,均处在同一个集装箱上面。

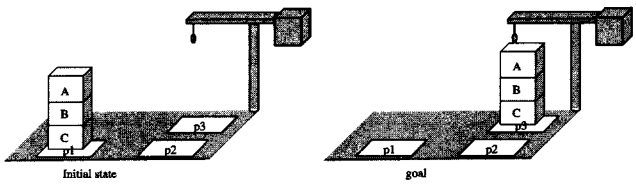

图 2 将集装箱从 $p1$ 移到 $p3$ 的规划问题,要求移动前和移动后集装箱的堆放顺序不变

HTN 规划方法采用任务分解的思想来解决类似例 3 形式的规划问题。下面给出关于解决此问题的 HTN 方法的非形式描述。

• 为了完成在保持集装箱堆放顺序的前提下移动集装箱的任务,对这些集装箱,可以下列方式进行处理:首先将其移动到中间位置(此时集装箱以逆序的顺序堆放),然后再将其移动到目标位置(此时集装箱又将恢复到初始状态时的堆放顺序)。

• 为了完成将一堆集装箱从位置 p 移动到位置 q 的任务,可按下列方式进行处理:重复移动位置 p 最上面的集装箱到位置 q,直到位置 p 上不再有集装箱为止。

• 为了完成将位置 p 上最上面的集装箱移动到另一位置 q,可直接执行规划领域中的基本动作 take 将集装箱从位置 p 搬走,并执行 put 动作将集装箱放到位置 q 的上面。

HTN 规划的问题描述包含规划基本动作、初始状态、保持集装箱堆放顺序不变的前提下集装箱的任务以及上述方法的形式化描述。这些方法使得这些规划问题的求解显得更加容易,因为这些方法只产生属于问题合法解的规划。

在 HTN 规划器中,规划的目的不是要达到某一目标状态集合,而是要完成某一任务集合。也就是说,与 STRIPS 方法和 PDDL 方法不同的是,HTN 语言不是直接告诉规划器规划目标,而是将规划目标转换成任务集合的形式。同时,规划系统的输入不仅包含与经典规划器类似的动作集合,还包含一方法集合。所谓方法就是以处方(prescription)的形式告诉系统如何将某一类任务分解成更小的子任务集合。规划的过程就是递归地将那些非原子任务分解成越来越小的子任务,直到出现那些可以直接执行规划动作就能完成的原子任务为止。同时,HTN 规划器也是以方法的形式执行动作。

下面来看如何用 HTN 语言对类似图 2 形式的规划问题进行表示。首先,与 STRIPS 方法和 PDDL 方法类似,对于图 2 中的初始状态,我们容易用一阶逻辑语言进行表示,假定初始状态用 s_0 表示。其次,在 STRIPS 方法和 PDDL 方法中,对规划目标的表示与初始状态的表示形式是一样的,即都是用一阶语言对图 2 中的 goal 状态进行描述。但在 HTN 语言描述中,规划目标是以任务的形式给出的。为了将系统从初始状态转换到目标状态,必须完成 $t1$=move-one-stack()的任务。当规划目标以规划任务的形式给出后,我们需要用一序列的方法实现对规划任务的分解,直到将抽象规划任务细分到我们可以直接用动作来完成某一任务为止。例如要完成 move-one-stack()任务,需要完成 $t2$=move-stack($p1$,$p2$)和 $t3$=move-stack($p2$,$p3$)两个子任务。也就是说,我们可用方法 $m1$=move-twice(),将任务 $t1$ 分解为两个子任务 $t2$ 和 $t3$。而对于子任务 $t1$,我们可以不断地将 $p1$ 最顶部的集装箱移动到 $p2$,直到 $p1$ 不存在其他集装箱为止。子任务 $t3$ 的分解方法与 $t1$ 相同。因此,对于子任务 $t1$ 和 $t2$,我们可用方法 $m2$=recursive-move(p,q,c,x)不断地移动最顶部的集装箱,即 $t4$=move-topmost-container(p,q)。要完成任务 $t4$,我们可以首先抓起要移动的集装箱,移到目的地,然后将集装箱放下,即完成任务 $t4$ 可用方法 $m3$=take-and-put(c,k,$p1$,$p2$,$x1$,$x2$)实现。而方法 $m3$ 又将任务分解成两个子任务,将集装箱抓起,即 $t5$=take(k,c,$x1$,$p1$),然后将集装箱放下,即 $t6$=put(k,c,$x2$,$p2$)。显然,任务 $t5$ 和 $t6$ 直接使用相对应的基本动作 take 和 put 即可完成,此时就不需要对任务 $t5$ 和 $t6$ 再进行分解了。

基于上述分析,我们可以将图 2 所示规划问题表示成任务的形式,并给出如下所示的这些任务相应的分解方法:

```
move-twice():
    task:       move-one-stack()
    precond:      ; no preconditions
    subtasks:     ;move the stack twice:
                  t2=move-stack(p1,p2)
                  t3=move-stack(p2,p3)
                  {(t2,t3)}

recursive-move(p,q,c,x):
    task:       move-stack(p,q)
    precond:    top(c,p),on(c,x) ; true if Pis not empty
    subtasks:   〈move-topmost-container(p,q),move-stack(p,q)〉
                ;; the second subtask recursively moves the rest of
                stack
do-nothing(p,q)
    task:       move-stack(p,q)
    precond:    top(pallet,p) ;true if Pis empty
    subtasks:   〈〉 ; no subtasks,because we are done.

take-and-put(c,k,l1,l2,p1,p2,x1,x2):
    task:       move-topmost-container(p1,p2)
    precond:    top(c,p1),on(c,x1), ; true if p1 is not empty
                attached(p1,l1),belong(k,l1),; bind l1 and k
                attached(p1,l2),top(x2,p2) ;bind l2 and x2
    subtasks:   〈take(k,l1,c,x1,p1),put(k,l2,c,x2,p2)〉
```

如果我们用 M 表示上述所有方法的集合,O 表示 take,put 等构成的基本动作集合,则图 2 中的规划问题可表示成下列三元组 P=〈s_0,$t1$,D〉的形式,其中,s_0 表示初始状态,$t1$ 表示规划任务,D=(M,O)表示规划领域。

结束语 智能规划问题本质上是如何用计算机进行问题求解的问题。因此,解决智能规划问题首先必须对实际规划问题进行形式表示。一般来说,智能规划问题由三个基本要素:初始状态、目标状态和规划动作。各种不同的规划形式表示方法主要体现在对规划问题三要素的不同表示,这种形式表示上的差别背后隐藏的是对智能规划问题的不同理解:情景演算方法用极具表达力的二阶逻辑语言对动态世界进行刻画,在情景演算框架下,规划求解过程就是一个定理证明的过程;STRIPS 方法部分否定了情景演算的方法,改用前提条件,添加效果,删除效果的方法直接对动作的适用条件和效果进行显式表示,这种做法为过程化规划求解奠定了基础,从而

带来规划求解效率的极大提高，但代价是牺牲了一定的表达能力；PDDL是在STRIPS方法基础上发展起来的，其主要贡献在于给出了一种统一的领域描述语言，并扩充其表达能力，以弥补STRIPS方法在表达能力上的不足；HTN语言则是从任务分解的角度对规划问题进行表示，如果待求解的规划问题很容易进行任务分解，则用HTN语言进行表示，则在规划求解过程中，用HTN规划方法就能快速地产生规划解。但HTN方法的难点在于不容易找到合适的任务分解的方法，通常情况下，对于很多实际规划问题，我们不太容易甚至根本不可能找到合适的任务分解的方法进行求解。

因此，从总体上来看，现代智能规划技术研究的发展趋势主要存在以下两个方向：一是从知识表示的角度出发，寻求新的有效的规划形式表示方法，扩充规划形式表示能力，以表示和刻画复杂的现实规划问题；二是在已有的规划形式表示框架下，寻求高效的智能规划求解算法，并最终设计出性能良好、处理能力强的规划器，以满足实际应用的需要。

参考文献

[1] Green C C. Theorem proving by resolution as a basis for question-answering systems//Meltzer B, Michie D, eds. Machine Intelligence 4, American Elsevier, New York, 1969: 183-205

[2] McCarthy J. Situations, actions and causal laws. Technical report. Stanford University, 1963

[3] Fikes R, Nilsson N. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, 2(3): 189-203

[4] McDermott D. The 1998 AI planning systems competition. AI Magazine, 2002, 21(2)

[5] McDermott D. The AIPS'98 Planning Competition Committee. PDDL-the planning domain definition language. Technical report. Available at: www.cs.yale.edu/homes/dvm, 1998

[6] Fox M, Long D. PDDL2.1: An extension to pddl for expressing temporal planning domains. Journal of Artificial Intelligence Research, 2003(20): 61-124

[7] Biundo S, Schattenberg B. From abstract crisis to concrete relief——a preliminary report on combining state abstraction and HTN planning // Proceedings of the European Conference on Planning(ECP). 2001: 157-168

(上接第102页)

$2)^{n/2}$，其中，n为HASH函数的输出长度，当n足够大时是可忽略的。除了上面的三种攻击外还有一种是拒绝服务攻击，但由于拒绝服务攻击本身不泄露秘密信息，不影响我们的讨论，因此可以不考虑。

以上的分析与推理说明采用基于消息链的状态检测方法能够准确判定主动攻击行为的发生。因此，只要将上述检测机制附加到交互型密码应用系统中，根据失败-停止协议的原理，就能实现对主动攻击的容侵功能，即使发生攻击也不会泄露秘密信息。

5 基于等待时间的入侵检测方法

在失败-停止协议中，终止协议执行的条件除了发生消息篡改或消息重放行为外，还有一个条件是期望的消息没有在规定时间内到达，因为任何类型的主动攻击都需要一定的时间，纯粹的中间人攻击在很多情况下构不成威胁。因此，可以设计一个消息响应的时间模型，确定协议执行中每一条消息发出后，容许等待回应消息的最长时间值。这个值应根据生成回应消息的计算量、客户或服务器的计算能力、网络的性能与延时等因素来确定。特别是当密码系统用于一个特定的群组时，时间模型能够被较为准确地设计出来。此外，为了防止敌人根据消息的计算时间来发现密钥，例如Kocher在文献[6]中提出了一种通过分析计算指数$m^x \bmod p$所花的时间来发现x的攻击方法，可以对指数运算量进行适当随机化。具体方法如下：

(1)秘密设置初始随机数种子：随机选择$r_1^0 \in Z_p^*$，计算$r_2^0 = (r_1^0)^{-x}$

(2)对$j=0,1,\cdots,N$，第j次计算$m^x \bmod p$的算法为：

① 计算$u=(m \cdot r_1^j) \bmod p, t=u^r \bmod p, t \cdot r_2^j \bmod p (=m^x \bmod p)$。其中，$u$是对消息$m$的盲化，$t \cdot r_2^j \bmod p = m^x \bmod p$是去盲。

② 设置新的种子：$r_1^{j+1}=(r_1^j)^2, r_2^{j+1}=(r_2^j)^2$

该算法的基本原理是，因为r_1^0是秘密选择的随机数种子，对于攻击者来说，在不知道r_1^0的情况下，r_1^1是随机的，同理，如果不知道r_1^0, r_1^1，则r_1^2也是随机的，依次类推，序列$r_1^0, r_1^1, \cdots, r_1^j$具有随机性。当然，在计算中的暂时结果$u, t$和需要保存在内存的种子$r_1^j, r_2^j$不能泄露。

时间模型确定后，通信双方可以内部建立各自的消息容许等待时间表，其中包括对预期消息的开始等待时间，容许等待的最大时间等，用以判断预期接收的消息在到达时间上的合法性，如果在容许范围内没有到达，则自动结束协议，并做相应的处理。这种容侵方案也是对前面基于消息链验证的方案的一种补充，虽然简单，但对那些安全需求较高的应用系统是很有用的，因为这是一种以效率换取安全的使用解决方案。

结束语 密码系统的安全实现是一项复杂的系统工程。本文将容侵思想应用于密码系统的实现，并利用失败-停止协议的基本概念提出了基于消息链鉴别和基于等待时间的容侵实现方法，这种方法的特点是简单实用，计算量的增加也不大，而且具有一般性。不仅可以用于密码协议来保证单个数据包的可靠性，也可用于TCP协议，对整个数据流的可靠性进行检验。需要注意的是应针对具体应用选择安全的HASH函数，并对内存秘密数据进行保护。

参考文献

[1] Bleichenbacher D. Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1 // Hrawczyk. Advances in Cryptology-CRYPTO'98. vol. 1462 of Lecture Notes in Computer Science. Berlin: Springer-Verlag, 1998: 629-660

[2] Manger J. A Chosen ciphertext attack on RSA optimal asymmetric encryption padding(OAEP) as Standardized in PKCS # v2.0 // Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London, UK, 2001

[3] Lucks S, Schmoigl N, That E I. The Idea and Architecture of a Cyptographic Compiler // WEWoRC 2005 Coference Records. Leuven, Belgium, 2005

[4] Millen J, Denker G. CAPSL and MuCAPSL. Journal of Telecommunications and Information Technology, 2002, 4: 16-25

[5] Li Gong, Syverson P. Fail-sto Pprotocols: An approach to designing secure protocols // R. K. Iyer, M. Morganti, Fuchs W. K, and V. Gligor, eds. Dependable Computing for Critical Applications. 5. IEEE Computer Society, 1998: 79-100

[6] Kocher P. Cryptanalysis of Diffie-Hellman, RSA, DSS, and other Cryptosystems using timing attacks. In Advances in Cryptology, CRYPTO'95 // 16TH Annual International Cryptology Conference. California, USA, 1995