# 第四部分　基于逻辑的规划方法

## 第21章　状 态 演 算

### 21.1　状态和动作推理

第7章介绍了状态空间的概念及如何搜索来计算动作计划以到达目标。当时还将讨论了基于图标或基于特征的状态空间搜索。现在，我们有更丰富的语言来描述特征和特征之间的约束，能更全面地研究基于特征的规划方法。

我们能删掉世界状态的一些属性，它们与当前的问题不相关或是未知的，这个事实是使用基于特征方法的强有力的方面。这在描述我们想让 agent通过其动作达到的目标条件方面尤其重要。从图21-1中的配置开始，我们想让agent生成一个计划以让某个积木在*B*上面（不必关心哪个积木在*B*上或者*B*在那里）。这个目标能用公式（ x）On(x, B)简单地描述。一般地讲，在谓词演算中，一个目标条件能用任何合式公式描述，我们能通过试图从这些公式证明目标合式公式来确定一个目标能否在一个由这些公式描述的状态空间中被满足。

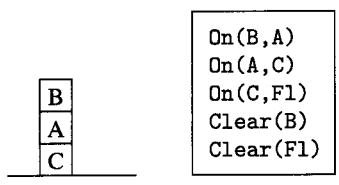


图21-1　积木的一个配置

在本章和下一章，将介绍一些用来寻找一组动作以获得由合式公式描述的目标状态的技术。这里用谓词演算机制直接推理状态和动作。就像在所有的谓词演算推理中一样，搜索仍然是必要的，但现在搜索是在一个逻辑表达式空间上进行，而不是在一个世界状态的模型空间上进行。在下一章，描述了另一个方法，在那里算子被用来改变状态描述，搜索将在一个状态描述空间上进行。

状态演算(*situation calculus*)[McCarthy & Hayes 1969, Green 1969a]是一种状态、动作和动作作用于状态的结果的谓词演算的形式化。在一阶谓词演算中我们把知识表达为关于状态和动作的公式，然后用一个演绎系统来问这样的问题：“存在满足一定（目标）属性的状态吗？如果存在，现在的状态如何通过动作才能被转换为那种状态呢？”。对这样一个询问的回答是构造一个到达希望状态的计划。虽然状态演算在一些早期的 AI规划系统中有显著的地位，但现在很大程度上已被下一章讲到的方法所替代。然而，这种方法对研究和帮助澄清有关动作结果的概念问题仍有重要作用。

通过一个积木世界的例子引入状态演算。假如把图 21-1中的状态标识为*S0*。用一阶谓词演算，我们可以用下面的公式描述*S0*：

$$On(B, A) \wedge On(A, C) \wedge On(C, Fl) \wedge Clear(B) \wedge \cdots$$

为了描述状态演算中的这个状态和其他状态，我们把状态具体化[⊖]，即将它们作为现存实

[⊖] 动词reify 意思是说把某个抽象的东西看作是一个实在的或具体的东西。

体包括进我们的世界概念中。可能有很多这种状态，它们能用常量符号（如S0，S1，S1，…）、变量或者函数表达式表示。改变原子合式公式以包括一个指称包含预期关系的状态项。要对这些原子合式公式的预期解释进行一个相应的变化，这些原子合式公式现在指称状态间的关系，它们被称为流（*fluent*）。用公式构造一个在状态*S0*中为真的句子：

$$On(B, A, S0) \wedge On(A, C, S0) \wedge On(C, Fl, S0) \wedge Clear(B, S0)$$

我们也可以有所有状态的命题真值。例如：

$$(\forall x, y, s)[On(x, y, s) \wedge \neg(y = Fl) \supset \neg Clear(y, s)]$$

和

$$(\forall s)Clear(Fl, s)$$

(Clear（x，s）的意思是x上可以放某个东西。)用这些一般公理，我们能证明*S0*的各种命题。例如，我们能证明¬clear(A, S0)和Clear(Fl，S0)。

为了表示动作及其结果，我们采取下面的步骤：

1) 把动作具体化（即我们想像有这样一个东西来做为动作）。这样，动作能用常量、变量或函数表达式表示。在状态演算形式中，一个动作被看作为动作涉及的实体的函数。在例子中，动作是积木的函数。例如，考虑把一个积木从一个地方移到另一个地方的动作。用表达式move（B，A，Fl）表示把*B*从*A*移到地板上的动作⊖（把一个动作看作为由那个函数值给定的一个“对象”）。

一般地讲，我们能用模式(*schema*)move (x, y, z)表达一个move动作系列，其中x、y和z是模式变量。利用常量，这些变量的实例化可以产生指称真正动作实例的表达式。

2) 接下来，想像一个函数常量 do，它指称一个可以把动作和状态映射到状态的函数。如果α代表一个动作，σ代表一个状态，那么do（α，σ）指称把状态—动作对映射到通过在σ指称的状态中执行由α指称的动作获得的状态的一个函数。

3) 用合式公式表达动作的结果。在某些形式中，对每个动作-流对有两个这种合式公式（目前，我们忽略动作对流没有影响的对）。对{On, move}对，合式公式是

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \supset On(x, z, do(move(x, y, z), s))]$$

和

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \supset \neg On(x, y, do(move(x, y, z), s))]$$

这里我们假定公式中提到的所有变量都被全称量化。这些公式中的第一个被称为正效应公理(*positive effect axiom*)，它描述了一个动作怎么使一个流为真。第二个叫负效应公理(*negative effect axiom*)，它描述一个动作如何使一个流为假。在这个例子中，一个效应公理的先行条件描述了为了应用那个动作必须满足的前提条件 (*precondition*)，结果描述了在应用动作后流是如何被改变的。

即使已根据On定义了Clear，我们也能为{Clear, move}对写出效应公理。当然，它们必须和定义以及{On ,move}的效应公理相一致。对{Clear, move}的效应公理是：

---

⊖ 注意，和前面用到的move算子相比，这里移动一个积木块的表达式包括那个积木被移走的地方。

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \wedge (y \neq z) \supset Clear(y, do(move(x, y, z), s))]$$

和

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z) \wedge (z \neq Fl) \supset \neg Clear(z, do(move(x, y, z), s))]$$

在这个例子中，前提由两部分组成。一部分表达了执行动作的前提条件；另一部分表达了条件，在这个条件下，如果动作被执行，动作将产生表达在公理结果中的效果。在正效应公理中，第二部分是（y≠z）（即使y=z，动作也能被执行——把一个积木从一个位置向后移到相同的位置，但它没有在结果中声明的效果）。在负效应公理中，第二部分是（z≠Fl）。（由于我们假定地板总是空的，没有任何动作可使它不空的）。

为了说明如何使用效应公理，考虑图 21-2顶部的积木世界状态。这个状态满足使用置换{B/x，A/y，S0/s，Fl/z}的效应公理的前提条件。因此，我们能应用由move(B, A, Fl)指称的动作推出如下结果：

$$On(B, Fl, do(move(B, A, Fl), S0)),$$

$$\neg On(B, A, do(move(B, A, Fl), S0)),$$ 和

$$Clear(A, do(move(B, A, Fl), S0))$$

这些表达式中的每一个都有do(move(B,A,Fl),S0)）作为它的动作后的状态项（为简短起见，用S1 指称这个动作后状态）。在图21-2中，显示了应用动作后的结果状态和描述那个状态的公式（除了从效应公理推出的公式，还有一些其他的公式也是 S1 的真值公式；下面将简要描述这些其他的公式是如何被推导的。）

即使在一个动作后，在执行动作允许的推导之前我们拥有的所有公式仍是真的！意识到这一点是很重要的（在把 *B*移到地面后，在状态 *S0*中，*B*仍在*A*的上面。同样，在把 *B*移到地板之前,在由do(move(B, A, Fl), S0)指称的状态中，B在地板上面。）状态演算中的公式在感觉上是“无状态的”，它们总是真值（它们谈论的状态）。

## 21.2 存在的一些困难

### 21.2.1 框架公理

图21-2明显地表明，并不是所有关于状态 do(move (B, A, Fl), S0)为真的语句都能从效应公理中推导出。例如，在移动之后，很明显在移动之前的状态为真的事实，如 *C*在地板上和*B*是空的，在移动之后状态仍为真值。典型地讲，动作只有“局部”影响，故留下很多流没有改变。为了对这些恒久不变的状态做推理，我们为每个动作和每个流提供一对所谓的框架公理(*frame axiom*)，它不会因动作的结果而改变。例如，{move, On}的框架公理是：

$$[On(x, y, s) \wedge (x \neq u)] \supset On(x, y, do(move(u, v, z), s))$$

和

$$(\neg On(x, y, s) \wedge [(x \neq u) \vee (y \neq z)]) \supset \neg On(x, y, do(move(u, v, z), s))$$

(如果在一个动作前一个积木在第二个积木上，那么在动作后它仍在第二个的上面，条件是那个动作没有把它从第二个积木上移走。如果动作前一个积木不在第二个积木上面，那么动作后它仍不在第二个上面，条件是该动作没有把它放在第二个上面)。

和效应公理类似，框架公理的第一个叫正框架公理 (*positive frame axiom*)，第二个叫负框架公理(*negative frame axiom*)。

{*move, Clear*}的框架公理是

$$Clear(u, s) \wedge (u \neq z) \supset Clear(u, do(move(x, y, z), s))$$

$$\neg Clear(u, s) \wedge (u \neq y) \supset \neg Clear(u, do(move(x, y, z), s))$$

(假如动作前一个积木是 clear，如果那个动作没有把另一个积木放在它上面，那么该动作后那个积木仍是clear。如果动作前一个积木不是clear, 而该动作没有从它上面移去另一个积木，那么该动作后它仍不是clear)。

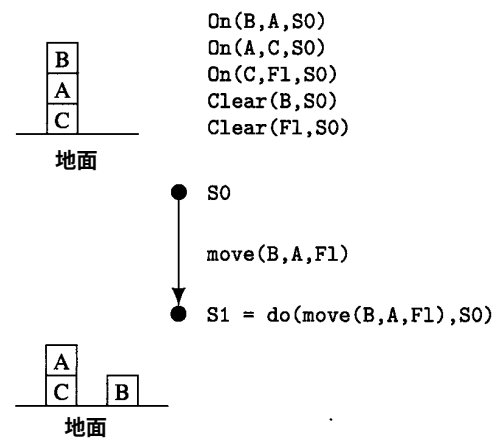


图21-2 将状态-动作对映射为一个状态

框架公理被用来证明如果状态由一个不影响某个属性的动作改变，那个属性保持为真。例如，{move, Clear}框架公理之一能用来推理图21-2中的clear (B, do(move (B, A, Fl)), S0))。典型地讲，由于对每一个流和动作的组合我们将有一对框架公理，在现实问题中用状态演算方式表达动作如何改变世界变得相当的难管理。

有几个作者已经探索了如何减少大量的框架公理，以及如何从效应公理自动导出框架公理。这里不想讨论这些技术，但是它们涉及到了那个假设 —— 能施加于一个流的变化仅仅是那些被效应公理明确指定的变化（你可以参考 [Pednault 1986, Schubert 1990, Reiter 1991, Elkan 1992])。即使能够减少大量的框架公理，用它们对关于在几个动作序列上什么流不会改变做推理的计算仍是笨重的。各种与解决不受动作影响的流有关的困难被称为框架困难。下一章将讨论解决框架问题某些方面的一个方法。

### 21.2.2 条件

描述一个形如move的动作的转换公式的前提为一个相当理想的事件给出了一个前提条件。假如我们想通过确认被移动的对象不是太重而更准确一些，那么我们必须给前提条件加上另一个合取式¬ Too_heavy (x, s)。这个特别关注能被无穷无尽地进行，导致要加上其他的条件像 ¬ Glued_down (x, s)、¬ Armbroken (s), …，和那个明确的预期解释。指定所有的重要条件的困难称为条件问题。已进行使用非单调推理的很多尝试以解决条件问题。这个思想是效应公理允许缺省的结论，如果要加入进一步的条件，这个结论能被撤消（例如，参见 [Dean & Wellman 1991, pp, 63以后])，这些尝试还没有完全成功。

### 21.2.3 分枝

还有其他的问题。在复杂的领域，我们常常用基于领域的一般知识演绎有关对象的语句。

例如，一个机器人可能推导出，如果它自己在一个房间里，那么它正在推的一个包也是在那个房间里。不用效应和框架公理推断包的位置，我们可能宁愿推理机器人的位置，然后用我们的普通知识去推理包的位置。例如，假如在状态S0我们有事实In(PA，R1)，其中PA指称某个包，R1指称某个房间。用它的效应公理，在移进某个不同的房间后（用R2表示），机器人能推断它（机器人）确实是在一个新的状态下。它还能推断出那个包也在一个新的状态下。但是现在我们如何才能阻止在新的状态下，框架公理断定那个包不是仍在由 R1指示的房间里呢？跟踪导出的公式哪一个在随后的状态转换中幸存被称为分枝问题。已经提出各种机制（与真值维护过程有关）以解决分枝问题。

## 21.3 生成计划

让我们暂时不考虑框架、条件和分枝问题，从原则上说明状态演算如何用来规划一个使用推理方法的动作序列。为了生成到达某个目标γ(s)的一个计划，我们设法证明( s) γ(s)，并用应答谓词提取状态作为嵌套动作的一个函数，那个动作产生了状态。

例如，假定我们想生成一个计划，它把积木 *B*从图21-1中的初始状态*S0*放到地面上。为了计算这个计划，我们必须证明（ s）On（B, Fl, s)。我们将用归结反驳法证明（ s）On(B, Fl, s)的否定句（ S）¬ On (B, Fl, s)与描述*S0*和移动结果的公式不一致。用一个应答谓词来捕获证明期间所做的置换。对这个问题的公式是：

$$\neg On(B, Fl, s) \vee Ans(s)$$

$$On(B, A, S0)$$

$$On(A, C, S0)$$

$$On(C, Fl, S0)$$

$$Clear(B, S0)$$

$$Clear(Fl, S0)$$

$$[On(x, y, s) \wedge Clear(x, s) \wedge Clear(z, s) \wedge (x \neq z)$$

$$\supset On(x, z, do(move(x, y, z), s))]$$

最后一个公式是{On,move}的正效应公理。在执行一个归结反驳法之前，它必须被转换成子句形式。像我们在归结反驳法中做的那样，单独留下等式谓词 (x≠ z ) （这是表达¬(x=z)的一种方式)，直到它的所有变量被约束。然后，像前面讨论的那样，假定我们有无穷多的像（A=A)、¬(A=B）等等这样的公理可用在逆着它们的归结法中。将归结反驳法会产生 Ans (do (move (B, A, Fl), S0)留给你去验证。

如果到达一个目标需要几个动作，动作函数将是嵌套的。例如，如果问题是从图 21-1中的初始状态把*A*放在*B*上，我们将得到Ans (do (move（A, C, B）,do (move (B, A, Fl), S0)))。但是你将注意到，如果试图用效应和框架公理解决这个问题，证明工作对这个简单的计划是太多了。用状态演算产生计划的大部分努力都受到这类框架问题的困挠。

## 21.4 补充读物和讨论

条件问题首先由[McCarthy 1977]提出。使用非单调推理方法的各种努力试图解决框架和条件问题。[Hanks & McDermott 1986]发现，非单调结论关于哪些流改变了和哪些流保持没变是模糊的。以后由[Baker 1991]和[Shoham 1988]所做的工作讨论了这个问题（参见[Kartha 1994]

关于Baker方法的评论）。框架问题的另一个非单调方法使用一个“后继状态公理”限制和描述了能被任何动作施加于任何流的所有变化[Reiter 1991]。[Shanahan 1997]是一本关于框架问题的书。

虽然大多数当前有关agent计划的研究都使用了下一章描述的另一类方法，但状态演算和相关语言GOLOG（alGOl in LOGic）[Levesque, et al. 1997]在Toronto大学的认知机器人小组仍是机器人研究的基础[Scherl & Levesque 1993]。

## 习题

21.1 想像一个机器人$R$，在一个有两个盒子$B1$和$B2$的房子中。在初始状态，机器人在位置$PR$，$B1$在$P1$，$B2$在$P2$。在这个问题中，所有的三个实体（$R$、$B1$和$B2$）在同一位置是可能的。实际上，我们想让$B1$和$B2$在位置$P3$。机器人有两个动作，用`goto(x)`和`push(y, w, z)`表示。`Goto(x)`表示把机器人从任何地方移到`x`指示的位置；它没有任何前提条件。它的惟一结果是在执行那个动作后，机器人将处在`x`位置处。动作`push(y, w, z)`把机器人和由`y`表示的盒子移到由`z`指示的位置。只有当机器人和盒子都在相同的位置`w`时，该动作才能执行。它的结果是机器人和对象`y`都在`z`位置。

1) 用状态演算中的合式公式表达初始状态和目标条件。把初始状态公式和目标公式的否定形式写成子句形式。
2) 把两个动作的前提条件和结果表达为状态演算中的合式公式。把这些合式公式转换为子句形式。
3) 写出所有的框架公理，把它们转换成子句形式。
4) 勾画出一个步骤策略，导出一个从初始状态到达目标的计划。你实际上不必进行证明；仅仅用语言解释一下如何进行就行了。

21.2 想像一个屋中有一个机器人、一本书、一个桌子和三个不同的地方，分别叫做中间、壁橱和门口。机器人在门口，书在在桌子上，桌子在中间。机器人能在屋子的任何地方之间移动（即使那里已有东西）。机器人能把桌子从一个地方推到另一个地方，只要机器人和桌子在同一地方。

1) 用状态演算描述这个初始状态。
2) 用状态演算写出描述机器人移动结果的公式。
3) 用状态演算写出描述机器人推桌子的结果公式。
4) 写出需要的框架公理显示当机器人移动时书和桌子的位置不会改变。
5) 写出需要的框架公理显示当机器人推桌子时，有关书的位置的结果。
6) 按照初始状态和动作算子表达状态：书在桌子上，桌子在壁橱里。

21.3 建立Tower-of-Hanoi问题的一个状态演算公式（见习题7.4）。

1) 指定描述初始状态$S0$的公式。
2) 提出一个law-of-motion公式，描述一个移动结果，而且至少写一个与移动效果有关的框架公理。
3) 写出指定目标的公式。

21.4 考虑下面的积木世界问题：有两个积木（$A$和$B$）和三个位置（$L1$、$L2$和$L3$）。公式`At(x, y, s)`的预期含意是指在状态`s`，积木$x$在位置$y$。公式`Empty(z, s)`的预期含意是指

在状态s，位置z是空的。我们有一个简单的动作：move（x, y, z），它指的是把积木*x*从*y*移到*z*。我们假定move有一个前提条件是一个积木正被移过去的位置必须是空的。做下面的工作：

1) 写出move的算子描述和框架公理。
2) 描述一个初始状态：*A*在*L*1，*B*在*L*3。
3) 描述目标：*A*移到*L*2，*B*在*L*3不动。
4) 用归结和Ans谓词生成一个动作序列，当在初始状态执行时，会满足刚刚描述的目标。对不相等的事情可以做任何假设，如（B ≠A），你可能在证明中需要它们。