1986年第4期　中山大学学报 ACTA SCIENTIARUM NATURALIUM UNIVERSITATIS SUNYATSENI　№. 4. 1986

HPS中冲突信息的利用及对临界值的学习校正指派

曹　勇
（计算机科学系）

摘　要

本文讨论了ABSTRIPS[1]系统的控制策略，提出利用冲突信息进行回溯以改进系统执行效率的MABSTRIPS系统，它大于ABSTRIPS的求解范围．还提出一学习算法，根据解树对临界值进行学习校正指派，以求解MABSTRIPS不能求解的问题．

ABSTRIPS分层系统是在STRIPS系统基础上引进抽象空间（Abstract Space）进行分层求解，同STRIPS一样，是使用STRIPS—型规则的不可交换系统。

定义　称STRIPS型规则（STRIPS—like operator）为下面形式的三元组对：

$$\phi = \langle C_\phi, D_\phi, A_\phi \rangle$$

C_ϕ、D_ϕ和A_ϕ分别代表规则ϕ的前提条件集、删除条件集和添加条件集，它们是文字的合取或析取形式，下面也称F规则。

在ABSTRIPS中，每个C_ϕ中的每一文字都附有一个临界值，相同的文字的临界值相同。在第i层抽象空间，所有C_ϕ中临界值不低于i的文字组成第i层抽象空间的规则的前提。

所要求解的问题是闭公式对$\langle x, y \rangle$，分别为初始条件公式和目标公式。在ABSTRIPS中，把目标公式作为一个哑操作的前提条件

$$DUMMY = \langle Y, \square, \square \rangle$$

其中□表示空公式，它永远处在解树根部。

定理1　所有按ABSTRIPS策略可解的问题，按STRIPS策略也可解。

证明　显然。

从控制策略可以看出，ABSTRIPS遵循如下的公设，原作者称其为长度优先策略。

公设　一个问题$\langle x, y \rangle$按ABSTRIPS控制策略是可解的，当且仅当在其每一抽象空

本文1985年1月收到

间，按ABSTRIPS控制策略求解都存在解。

推论 所有按ABSTRIPS策略可解的问题，在每一个抽象空间不考虑规则的低层前提时，按STRIPS策略都能求解。

STRIPS和ABSTRIPS系统的控制策略中虽然都有回溯点，但由于没有回溯信息提供，使在一些情况下求解效率不高，甚至在STRIPS系统可解的问题在ABSTRIPS系统是不可解的。参见附录。

若直接采用树形结构来保存解，是可提供回溯信息并根据它有目标地试探新的解。

我们采用一个栈来记录回溯信息，它的任一时刻的栈顶元素（若不为空）都代表下一次的回溯入口点。

注意到发生冲突时，或者当前的条件在以前实现过但后来被其它规则删除了；或者当前的条件初值时满足但被前面的某一规则删除；或者当前的条件只能依赖于初值而初值没有给定。这后一情形将直接返回到其父亲结点，寻求新规则，实现拥有当前冲突条件作为前提的规则所要实现的条件。在前二种情形中，解树中至少有一结点B 删除当前的条件，我们把结点B 入栈，并设置标志来标识二种冲突。

进栈遵循如下原则：

记栈顶元素为A，准备进栈的元素为B。

若B是A的祖先结点，则B不进栈；

若B的扩展层次比A的扩展层次高，则B不进栈；

若B的扩展层次与A的扩展层次相同，但在解树中B不在A的右边，则B不进栈；

在其它情形（包括原栈为空），B都进栈，成为新的栈顶元素。

初始运行时栈为空。

利用这个回溯信息栈，采用如下的回溯策略：

1。 在当前条件P无法实现时，往前寻找删除它的结点B，若找到，则将B入栈，否则将当前条件P的父亲结点入栈；P的父亲结点指扩展P作为前提条件的结点。

2。 取出栈顶元素A，若当前的P的父亲结点是A的祖先（可以相同），则寻求另一规则替代A，并回溯到A的扩展层。若找不到替换者，则当A与条件P结点为兄弟，交换A与P的实现次序（即将P的父亲结点的前提重新排序），重新实现排序后的前提，扩展层次不变。

3。 若找不到替换A的规则，且A与条件P不为兄弟，但P的父亲是A的祖先之一。此时，在A处开始回溯，扩展层次不改变。若如此一直回溯到与P 同代的结点仍发生冲突时，调换P的父亲结点前提次序重新实现。

4。 若P的父亲结点不是结点A的祖先，则先将A再次进栈，然后试图将P的父亲进栈。取出栈顶元素B（可能为A，也可能为P的父亲结点），重新寻找规则替代B，扩展层次置为B的扩展层。若找到规则，则在B处往下扩展，否则在B处回溯。

5。 在前面涉及到的前提排序中，若P的父亲结点的所有前提排序都已考虑，则寻找替换P的父亲结点的规则，找不到时往上回溯。

由于使用STRIPS型规则的求解系统是不可交换系统，因此解树的扩展过程只能是按层次从高到低、从左到右地进行。很自然，回溯时应选择最接近目前的无法实现条件

的结点，即由右到左，由低到高所检查出的第一个回溯入口点。本文的结点入栈策略和回溯策略正是依赖于这个想法。

我们把增加了回溯信息的分层求解系统记为MABSTRIPS。它的扩展过程是在解树上按后序周游次序扩展。在控制策略上与原系统ABSTRIPS的不同之处只在回溯的处理上，因此它也遵循“长度优先”策略。但有：

定理 2　一个问题$\langle x,y\rangle$按MABSTRIPS控制策略是可解的，当且仅当在其每一抽象空间，按STRIPS策略都能求解。

证明　显然，只需证明充分性。即要证明：一个问题$\langle x,y\rangle$，若在每一抽象空间，按STRIPS策略都存在解，则按MABSTRIPS也存在解。

设最高的抽象空间为第n层，显然MABSTRIPS在第n层存在解。

现设从第n层到第$i+1$层，MABSTRIPS都能找出解，考虑第i层。

在第i层抽象空间按STRIPS求解是指在规则集ϕ的所有规则中（包括哑规则），去掉前提中所有临界值低于i的文字，寻找一个以哑规则为根，叶结点满足初值的解树。如果存在解，即说明在所有可供选择的规则中的所有可能的前提排序下，至少存在一组合适的组合构成解树。

MABSTRIPS求解是在$i+1$层的解树基础上，考虑实现所有规则的临界值为i的前提条件。

由于STRIPS在第i层可以找到解，因此如果按MABSTRIPS求解时，若所有的当前条件都是已经实现的或有规则进行实现，那末当检查到哑规则时，解也就找到了。

假若在扩展过程中，某一当前条件无法实现，即已经没有新规则来实现它，那末它或者是由于以它为前提的规则不合适，或者是前面最后一个删除它的规则不合适，或者是它的某个祖先结点的前提条件排序不合适。此时，我们按MABSTRIPS控制策略中的回溯策略，从右到左，从低到高有目标地处理冲突，就可以处理到目前为止发现的所有非平凡冲突。这里平凡冲突是指当前的冲突是由于以前检测出的某冲突，因没来得及改而引起的新冲突。

解决冲突的过程就是调换不合适的规则、或者调换规则的不合适排序的过程。因此，我们根据回溯信息总能有效地调换出在第i层合适的规则组合和前提排序来。

从定理2，我们看出MABSTRIPS确实比ABSTRIPS求解能力要强。但是，由于分层系统采用所谓的“长度优先”策略，以致于存在一些情形使得STRIPS可解的问题在所有依赖于“长度优先”策略的分层系统无法求解，除非对不同的问题采用不同的分层法。产生这种现象的原因是由于STRIPS的策略使得不能保证：一问题$\langle x,y\rangle$在STRIPS可解，当对前提分层时，在每一抽象空间也可解。

在STRIPS策略中，一个规则的前提只能在以这个规则为根的子树中实现，除非它作为另一个规则的副作用而实现可以出现在左边的某个子树内。这样一来，会发生如下情况：

假若某规则的前提本应是由另一规则副作用实现，但由于那一个产生副作用的规则所要实现的条件的临界值低于副作用实现的这个规则，那末在实现当前前提时，由于低的条件不考虑，因而无法在当前规则的子树以外的某地方引进一结点来副作用实现它，

造成在最低层按STRIPS可解，而在某高层不可解。

一旦出现这种情况，即说明我们在临界值的指派过程中把一些文字的临界值指派得过低，或者反过来说，是把另一些文字的临界值指派得过高。对这种过激的指派，有必要进行一定程度上的修正，使其既能达到好的执行度量，同时又能求解尽可能多的问题。

我们注意到，假若把扩展层置为最低层，赋于这一层的初始解序列只有DUMMY(Y,□,□)一个，此时按MABSTRIPS求解即等同于按STRIPS求解。如果此时找不到解，即$\langle x,y\rangle$在STRIPS下无解。我们只对能找到解的情形感兴趣。

定义 广义逆后序周游：

设多叉树T中结点A有m个儿子结点，从左到右依次排列为$B_1,B_2,\cdots,B_m$，那末周游次序为$B_m,B_{m-1},\cdots,B_1$，最后访问A。周游每一结点时又遵循广义逆后序周游。

指定初始入口的广义逆后序周游：

初始假设同前，若我们指定B_k为初始入口，那末周游次序为B_{k-1}，$B_{k-2},\cdots,B_1$，最后访问A。而周游每一结点时遵循广义逆后序周游。

由于扩展层为最低层，设为1，因此每一扩展结点的标层都为1，必须对其重新标层。按照我们的控制策略，解树中除根结点外，每一结点的引进均为实现某一条件，取该条件的临界值作为这个结点的层次值。

解树中除叶结点外，每一内部结点A都与一规则A'相关联，以A作为初始入口，广义逆后序周游结点A的父亲结点，直到找到一个结点B，它的添加集中含有规则A'的一个前提P，而P在A的下一代结点中没有实现它的结点。若找不到，则以A的父亲结点为入口继续，若一直都未找到，则必出现在初始集，不考虑。在结点B和条件P之间建立一种联系以利于后面的修正指派。对规则A'的所有没有在A的下一代结点实现的条件都找出相应的B结点来。

对解树根结点，定义其层次值为0，采用如下的修正算法进行学习修正指派：

1. FOR层次从n到1 STEP－1 DO
2. Begin
3. Repeat F:=0; A:=DUMMY;
4. Repeat if A的层次值与当前层次值相等，then对所有与结点A相关联的文字，如果文字的临界值高于当前层次值，则将文字的临界值置为当前层次值，且将标志F置为1；
5. A:=按前序周游解树时A的下一个；
6. Until A为空；
7. if F=1 then 根据新的文字临界值对解树重新标层；
8. Until F:=0；
9. END；

算法结束时的临界值指派即作为修正结果。所有未出现在解中的文字的临界值将维持不变。

由修正算法可得如下结论：

定理3 设d_1，d_2为出现在解树中的任意两个文字，d_1作为某规则的前提条件被实

现d_2的规则所实现，那末通过修正后，d_1,d_2的临界值d_1'，d_2'满足 $d_1' < d_2'$。

附　录

一个ABSTRIPS系统不能求解的实例

仔细分析ABSTRIPS的控制策略，可以发现如下现象：

同一规则的几个前提条件的实现中，临界值高者先实现（临界值高于扩展层次时除外），反映到解树上就是临界值高的前提条件在靠左边的子树实现；

对一规则，只关心本层没有实现的条件，而不注意已经满足的前提条件是如何实现的。

考虑积木世界如下一组带推理型规则的规则集：

1. PICKUP(x)
   P&D: ONTAB(x), CLEAR(x), HANDEMPTY
   A: HOLDING(x)
2. PUTDOWN(x)
   P&D: HOLDING(x)
   A: ONTAB(x), CLEAR(x), HANDEMPTY.
3. STACK(x,y),
   P: HOLDING(x), LARGE(y,x), SQUARE(Y), CLEAR(Y)
   D: HOLDING(x), CLEAR(Y)
   A: ON(x,y), CLEAR(x), HANDEMPTY
4. UNSTACK(x,y)
   P&D: ON(x,y), CLEAR(x), HANDEMPTY
   A: CLEAR(Y), HOLDING(x).
5. REASON1 (x,y,z)
   P: LARGE(x,y), LARGE(y,z)
   D:
   A: LARGE(x,z)
6. REASON2(x,y)
   P: ON(x,y)
   D:
   A: SQUARE(y), LARGE(y,x).

取前提条件中各谓词的临界值指派为：

ON	6
CLEAR, HOLDING	5
ONTAB	4
LARGE	3
SQUARE	2
HANOEMPTY	1.

所要求解的问题是：

目标：ON(A,C)，ON(C,D)，ON(D,B)

初始条件为：CLEAR(A)，CLEAR(C)，ON(C,B)，CLEAR(D)

SQUARE(C), LARGE(B,D), ONTAB(A), ONTAB(B),
ONTAB(D), SQUARE(D), LARGE(C,A), LARGE(D,C).

使用STRIPS求解，可得如下解序列：

REASON2(C,B), UNSTACK(C,B), PUTDOWN(C), PICKUP(D),
STACK(D,B), PICKUP(C), STACK(C,D), PICKUP(A) STACK(A,C)

而用ABSTRIPS求解时，无论按其策略怎样回溯，在第三层都无法找到解，因而ABSTRIPS不能求解此问题。

参 考 文 献

[1] Sacerdoti, E.D., Planning in a Hierarchy of Abstraction Spaces, *Artificial Intelligence*, 5 (1974), 115—135.

[2] Nilsson, N.J., *Principles of Artificial Intelligence*, Palo Alto, CA, Tioga Publishing co., 1980.

Using Conflict Information and Adjusting Learning Indicated Criticality Values Based on Solution Tree in Hierarchical Problem Solver (HPS)

Cao Yong

Abstract

We discuss the control strategy of ABSTRIPS[1], and present a modified system——MABSTRIPS, which can use conflict information to guide backtracking. Its solvable range is larger than that of ABSTRIPS. We also present a new learning algorithm, which adjusts criticality values based on solution tree.