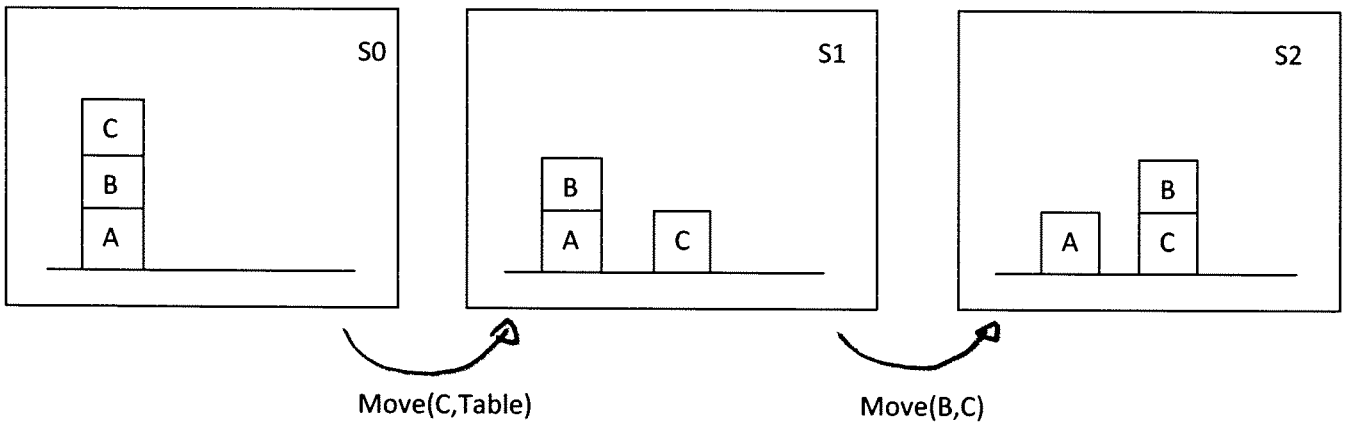
There are two ways to live your life – one as though nothing is a miracle.
The other as though everything is.

Albert Einstein


S0
C
B
A
S1
B
A
C
S2
B
A
C
Move(C,Table)
Move(B,C)

$ON(B,C) \wedge ON(A,B)$.

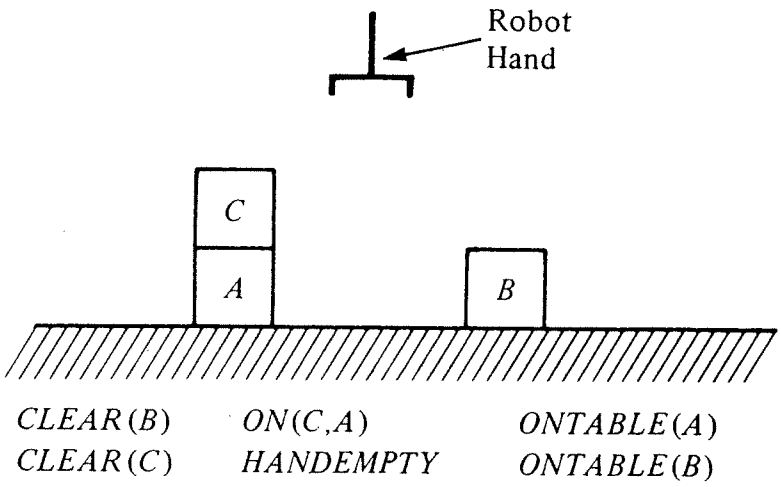



*Fig. 7.1 A configuration of blocks.*

**pickup**($x$)

Precondition: $ONTABLE(x) \wedge HANDEMPTY \wedge CLEAR(x)$

Delete list: $ONTABLE(x)$, $HANDEMPTY$, $CLEAR(x)$

Add formula: $HOLDING(x)$

1) **pickup**($x$)

P & D: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$

A: $HOLDING(x)$

2) **putdown**($x$)

P & D: $HOLDING(x)$

A: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$

3) **stack**($x,y$)

P & D: $HOLDING(x)$, $CLEAR(y)$

A: $HANDEMPTY$, $ON(x,y)$, $CLEAR(x)$

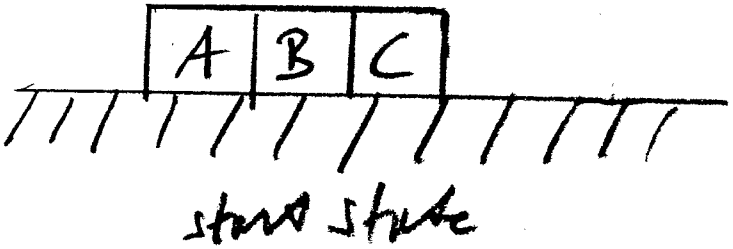
4) **unstack**($x,y$)

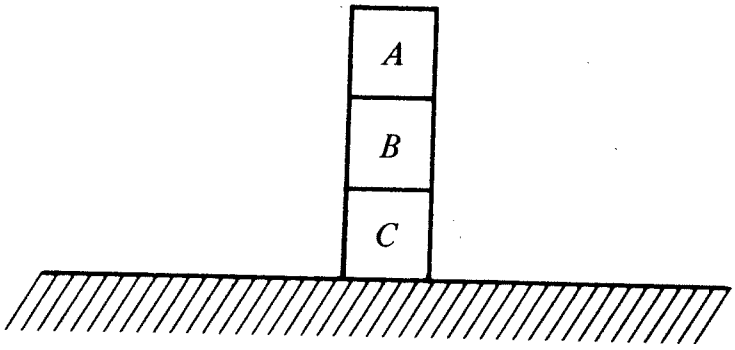
P & D: $HANDEMPTY$, $CLEAR(x)$, $ON(x,y)$

A: $HOLDING(x)$, $CLEAR(y)$

F-rules (F for formal planning)

simple problem
for forward planning




GOAL: $[ON(B,C) \wedge ON(A,B)]$

***Fig. 7.2 Goal for a robot problem.***

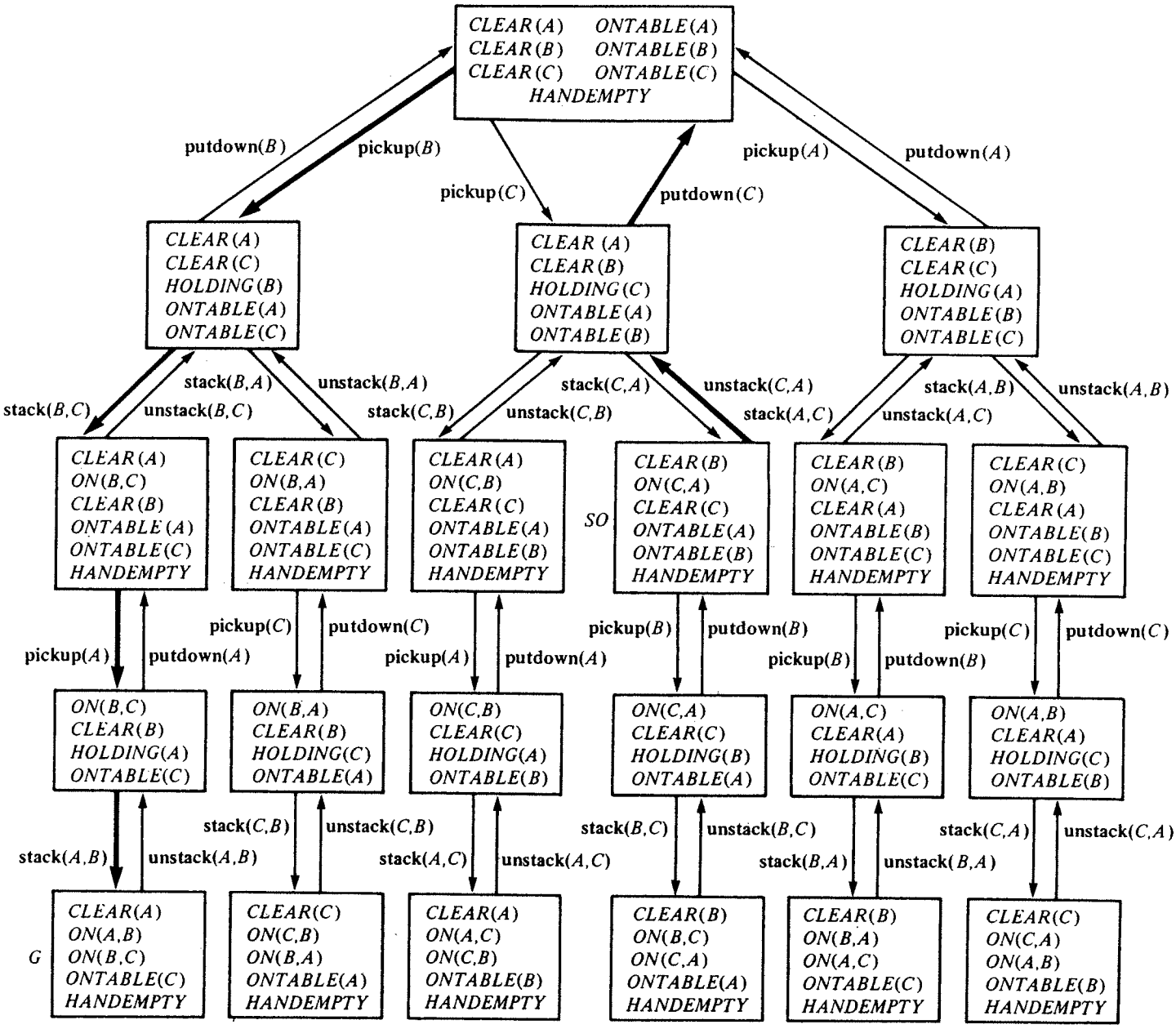



***Fig. 7.3 The state-space for a robot problem.***

State space for prior goals (example forward planning)

*Recursive Procedure* **STRIPS**( $G$ )

1 **until** $S$ matches $G$, **do:;** the main loop of **STRIPS** is iterative

2 **begin**

3 $g \leftarrow$ a component of $G$ that does not match $S$; a nondeterministic selection and therefore a backtracking point

4 $f \leftarrow$ an F-rule whose add list contains a literal that matches $g$; another backtracking point

5 $p \leftarrow$ precondition formula of appropriate instance of $f$

6 **STRIPS**($p$); a recursive call to solve the subproblem

7 $S \leftarrow$ result of applying appropriate instance of $f$ to $S$

8 **end**

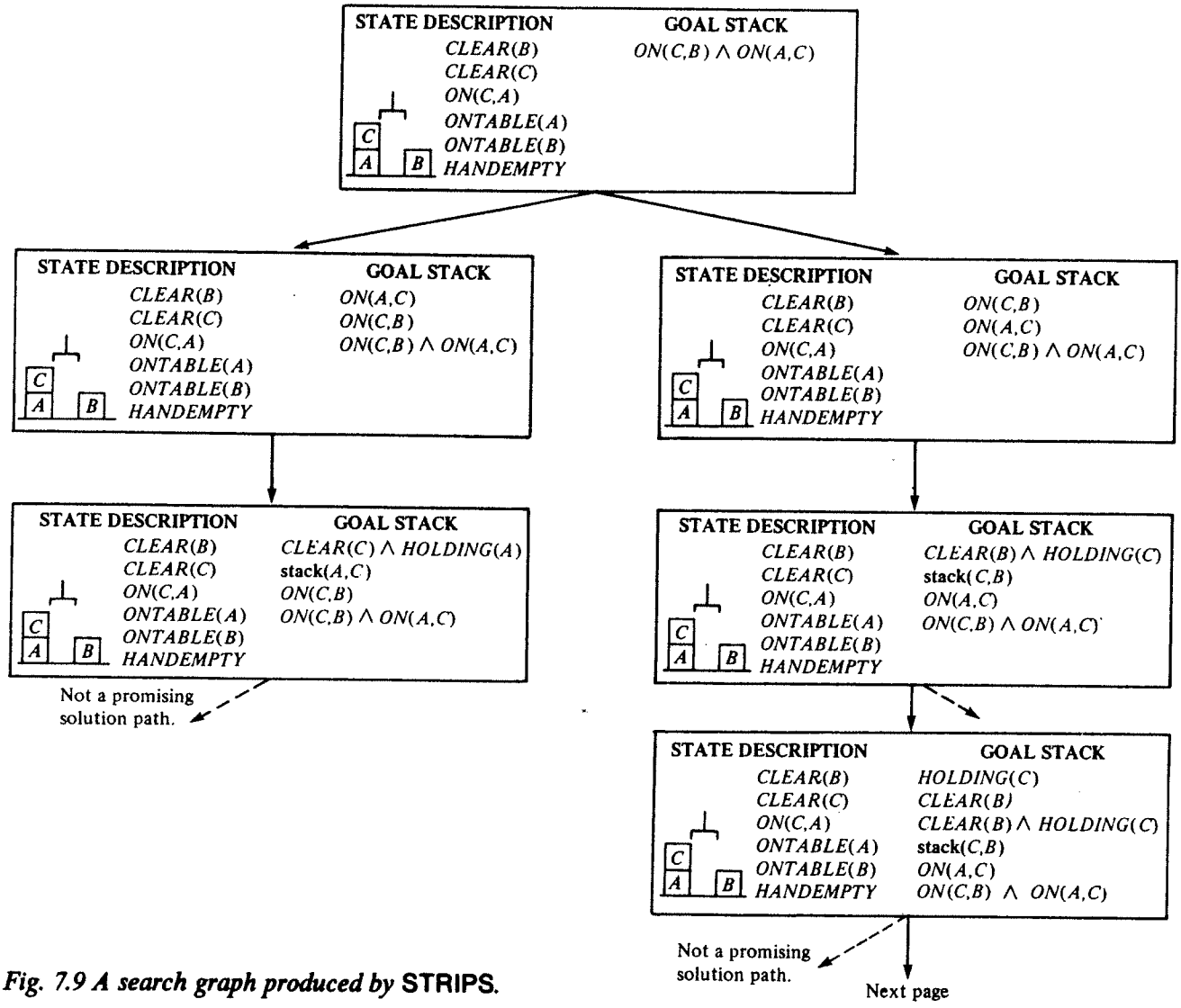
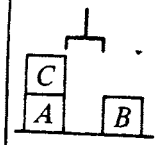



*Fig. 7.9 A search graph produced by* **STRIPS.**

From previous page

**STATE DESCRIPTION**

*CLEAR(B)*
*CLEAR(C)*
*ON(C,A)*
*ONTABLE(A)*
*ONTABLE(B)*
*HANDEMPTY*



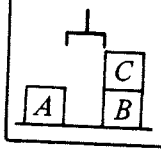

**GOAL STACK**

*HANDEMPTY* ∧ *CLEAR(C)* ∧ *ON(C,y)*
**unstack**(*C,y*)
*CLEAR(B)*
*CLEAR(B)* ∧ *HOLDING(C)*
**stack**(*C,B*)
*ON(A,C)*
*ON(C,B)* ∧ *ON(A,C)*

With $\{A/y\}$ the top subgoal matches the current state description. We can then apply **unstack**(*C,A*). Now the next two goals match also, so we can apply **stack**(*C,B*).

**STATE DESCRIPTION**

*CLEAR(C)*
*CLEAR(A)*
*ON(C,B)*
*HANDEMPTY*
*ONTABLE(A)*
*ONTABLE(B)*

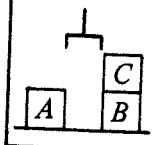



**GOAL STACK**

*ON(A,C)*
*ON(C,B)* ∧ *ON(A,C)*

**STATE DESCRIPTION**

*CLEAR(A)*
*CLEAR(C)*
*ON(C,B)*
*ONTABLE(A)*
*ONTABLE(B)*
*HANDEMPTY*

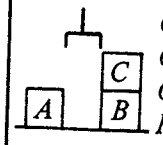



**GOAL STACK**

*CLEAR(C)* ∧ *HOLDING(A)*
**stack**(*A,C*)
*ON(C,B)* ∧ *ON(A,C)*

**STATE DESCRIPTION**

*CLEAR(A)*
*CLEAR(C)*
*ON(C,B)*
*ONTABLE(A)*
*ONTABLE(B)*
*HANDEMPTY*



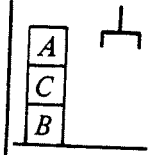

**GOAL STACK**

*ONTABLE(A)* ∧ *CLEAR(A)* ∧ *HANDEMPTY*
**pickup**(*A*)
*CLEAR(C)* ∧ *HOLDING(A)*
**stack**(*A,C*)
*ON(C,B)* ∧ *ON(A,C)*

Now we can apply **pickup**(*A*), and then the next goal will be matched, so we can apply **stack**(*A,C*). Now the last remaining goal on the stack is matched.

**STATE DESCRIPTION**

*ON(A,C)*
*ON(C,B)*
*HANDEMPTY*
*CLEAR(A)*
*ONTABLE(B)*




**GOAL STACK**

*NIL*