Lecture slides for

Automated Planning: Theory and Practice

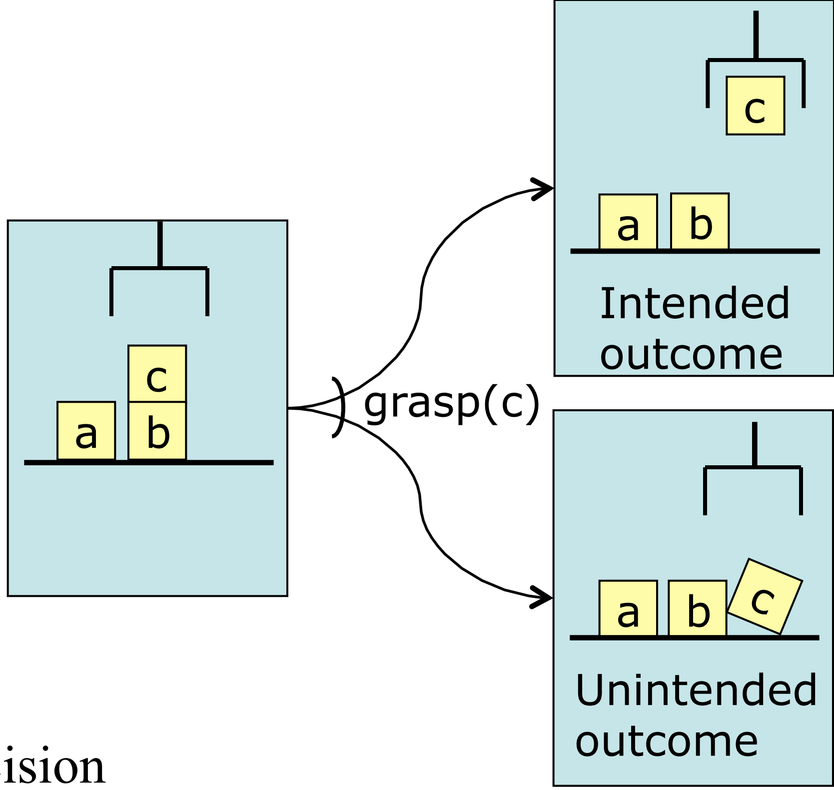
Chapter 17
Planning Based on Model Checking

Dana S. Nau

University of Maryland

1:19 PM February 29, 2012

Motivation

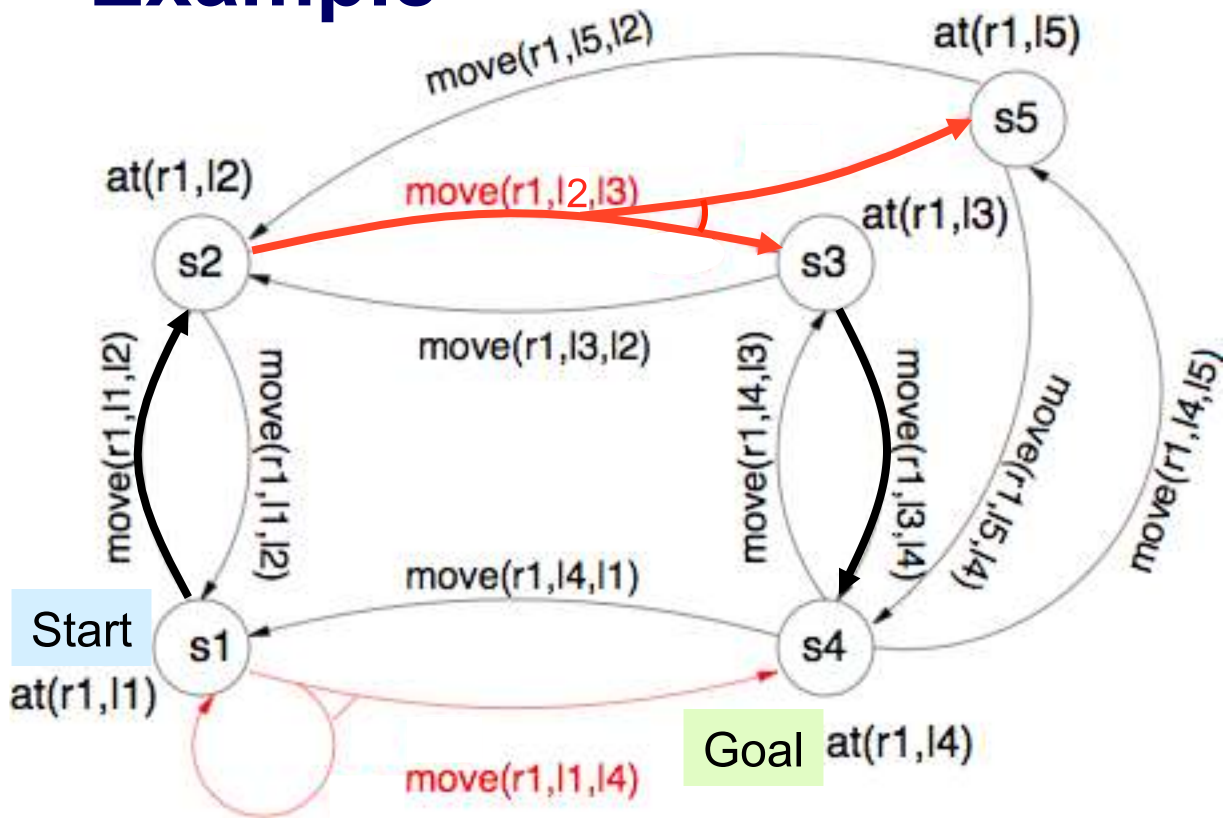


- Actions with multiple possible outcomes
  - Action failures
    - e.g., gripper drops its load
  - Exogenous events
    - e.g., road closed
- *Nondeterministic systems* are like Markov Decision Processes (MDPs), but without probabilities attached to the outcomes
  - Useful if accurate probabilities aren't available, or if probability calculations would introduce inaccuracies

Nondeterministic Systems

- *Nondeterministic system*: a triple $\Sigma = (S, A, \gamma)$
  - S = finite set of states
  - A = finite set of actions
  - $\gamma: S \times A \rightarrow 2^s$
- Like in the previous chapter, the book doesn't commit to any particular representation
  - It only deals with the underlying semantics
  - Draw the state-transition graph explicitly
- Like in the previous chapter, a policy is a function from states into actions
  - $\pi: S \rightarrow A$
- Notation: $S_\pi = \{s \mid (s,a) \in \pi\}$
  - In some algorithms, we'll temporarily have *nondeterministic policies*
    - » Ambiguous: multiple actions for some states
  - $\pi: S \rightarrow 2^A$, or equivalently, $\pi \subseteq S \times A$
  - We'll always make these policies deterministic before the algorithm terminates

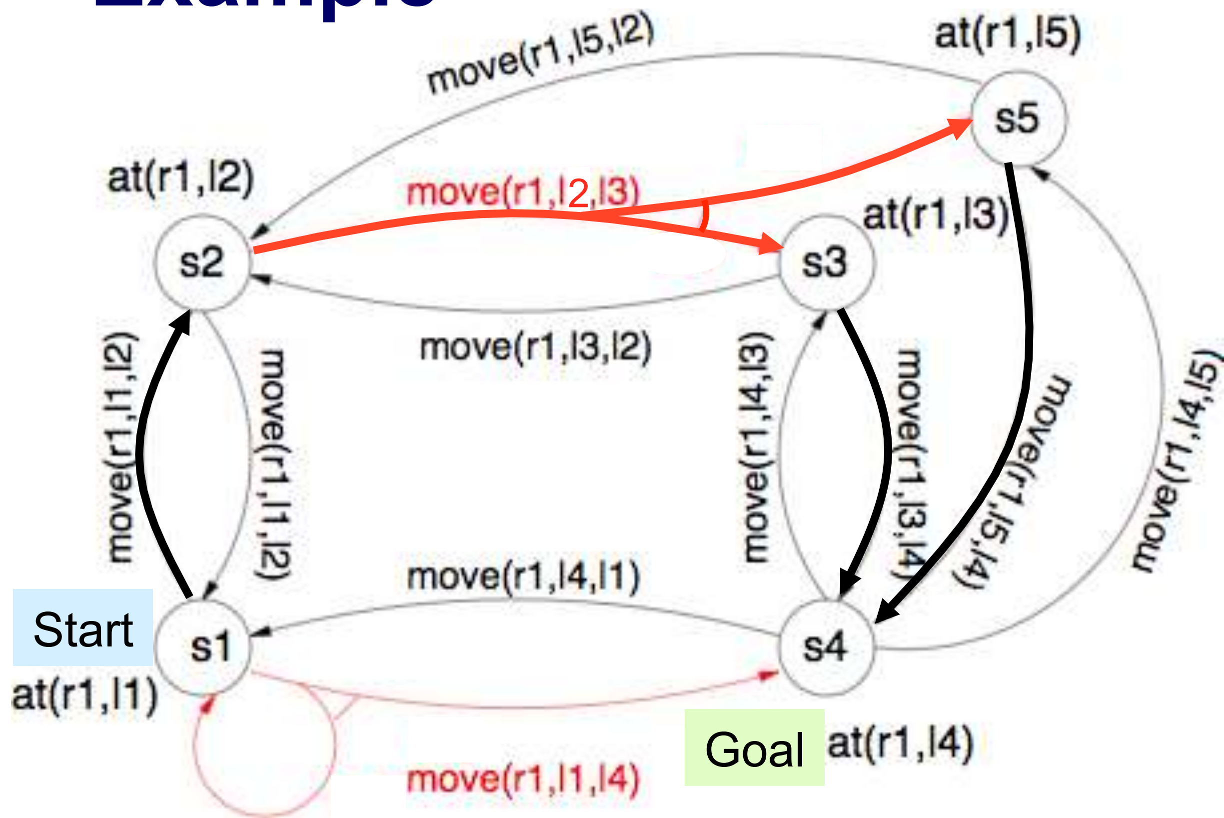
Example



- Robot r1 starts at location l1
- Objective is to get r1 to location l4

- π_1 = **{(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}**
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- π_3 = {(s1, move(r1,l1,l4))}

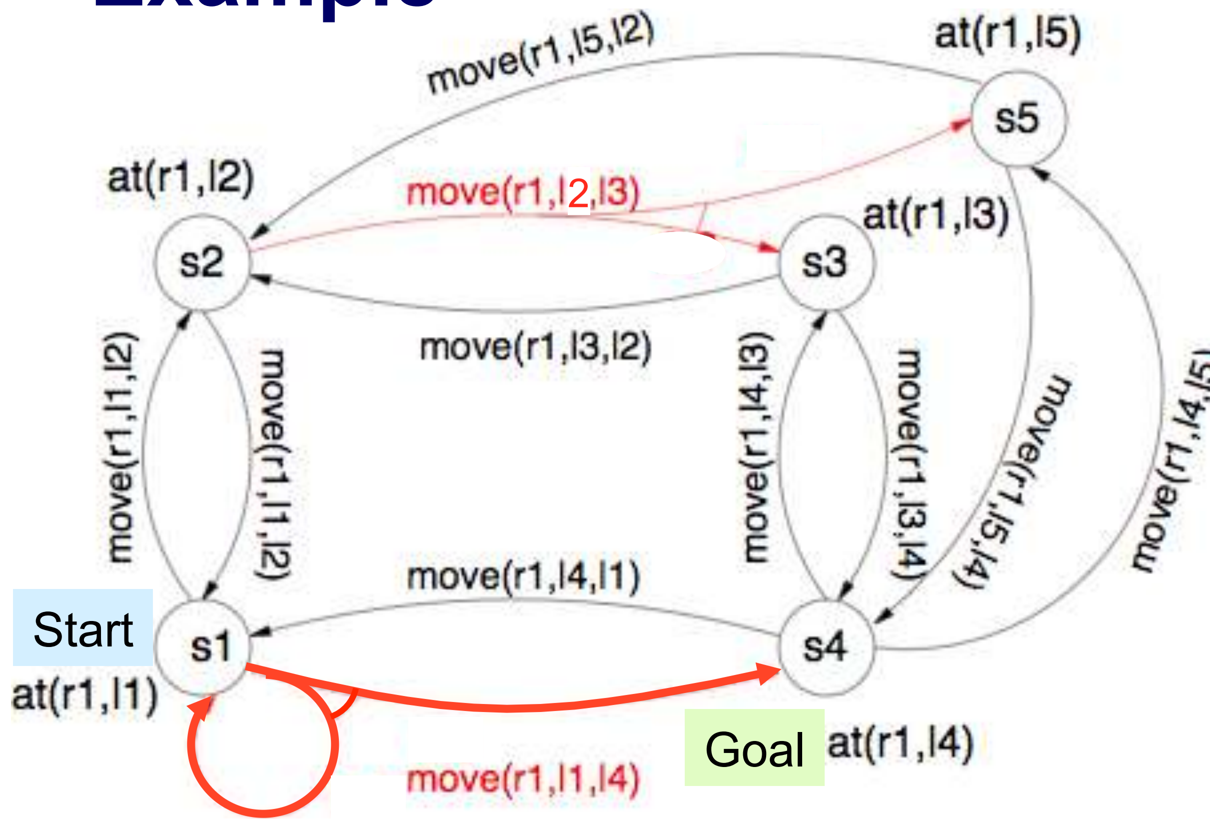
Example



- Robot r1 starts at location l1
- Objective is to get r1 to location l4

- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- **π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}**
- π_3 = {(s1, move(r1,l1,l4))}

Example

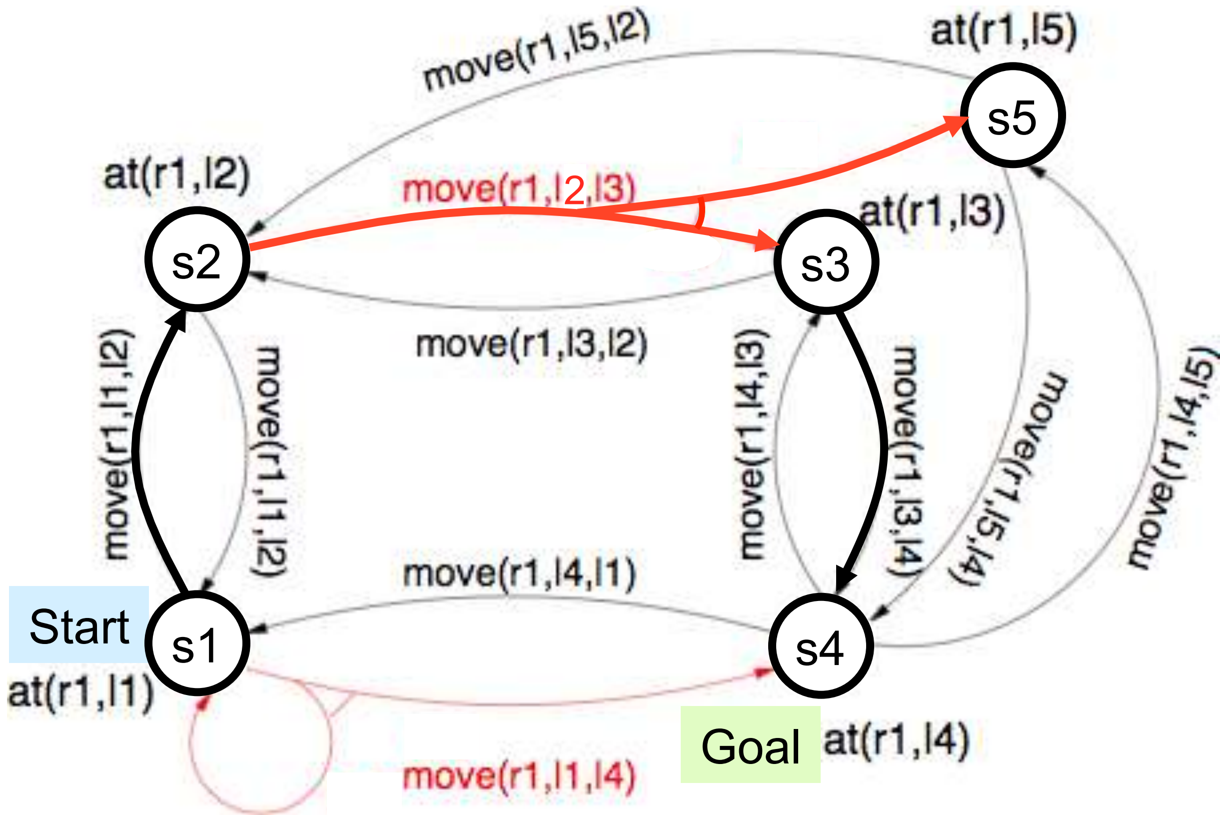


- Robot r1 starts at location l1
- Objective is to get r1 to location l4

- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- **π_3 = {(s1, move(r1,l1,l4))}**

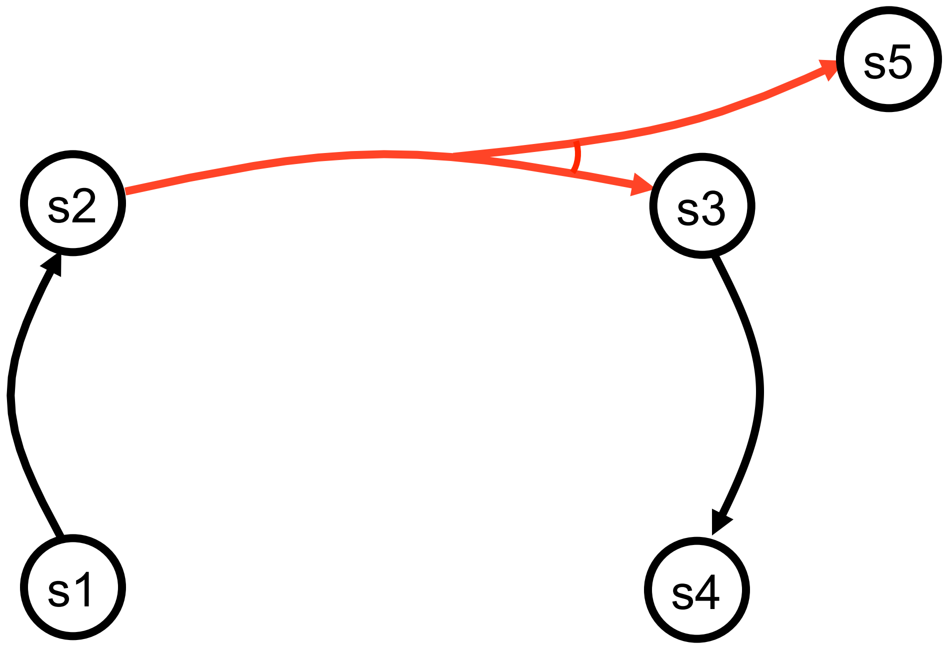
Execution Structures

- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$



- **π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}**
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- π_3 = {(s1, move(r1,l1,l4))}

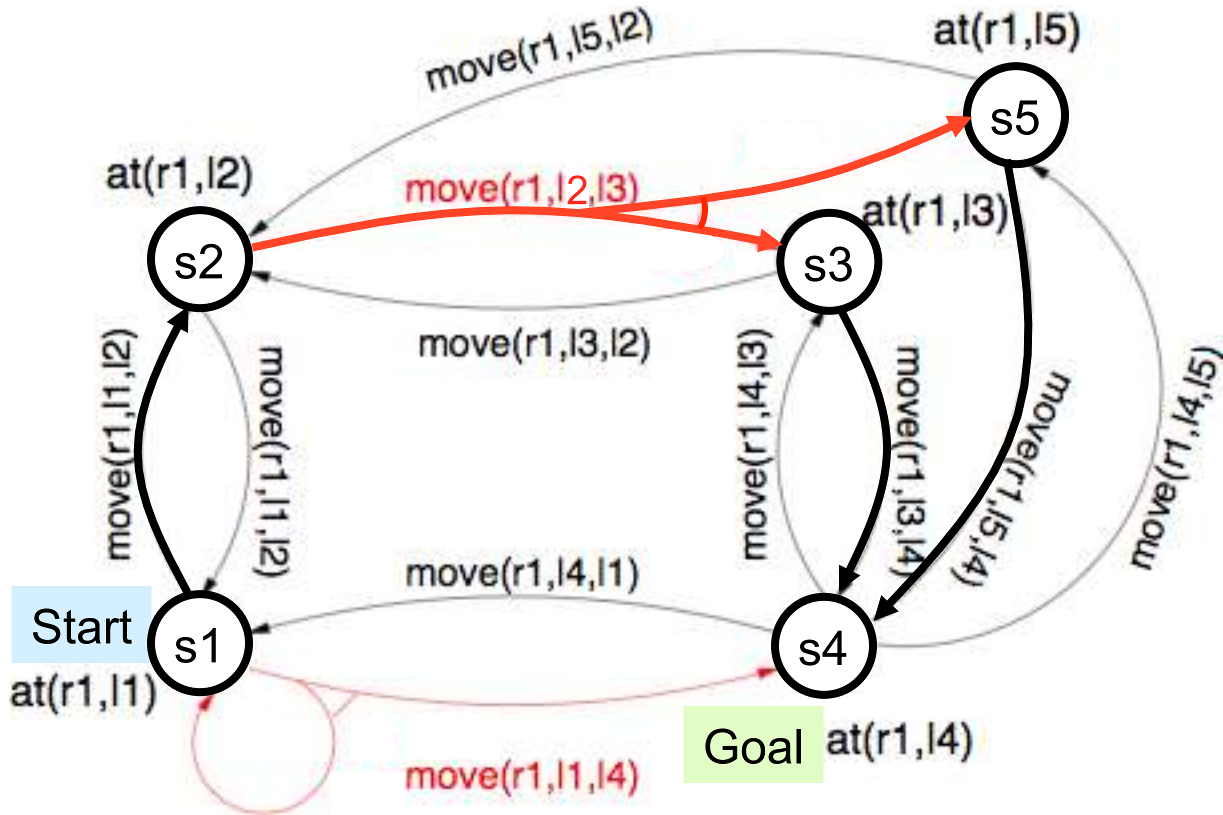
Execution Structures



- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$

- **π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}**
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- π_3 = {(s1, move(r1,l1,l4))}

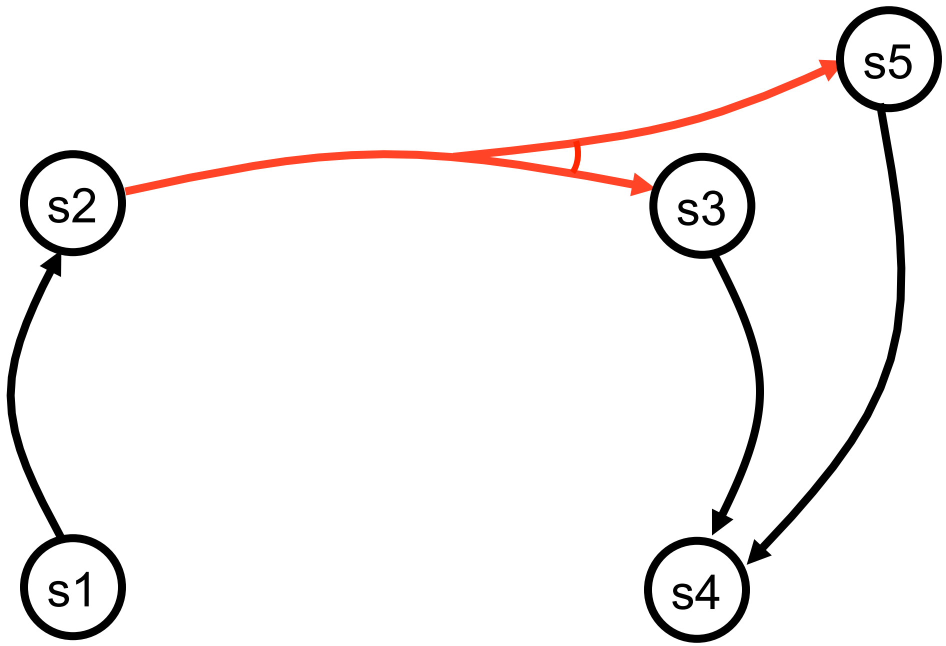
Execution Structures



- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$

- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- **π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}**
- π_3 = {(s1, move(r1,l1,l4))}

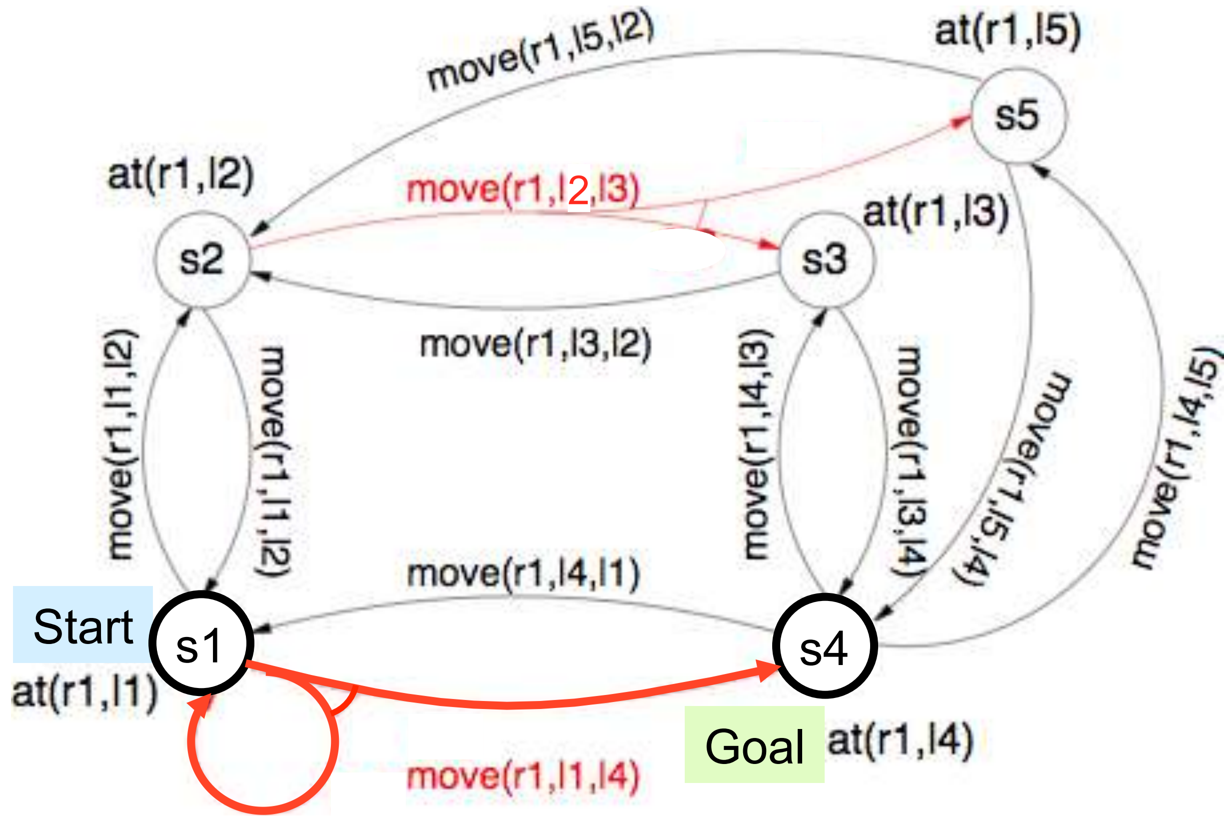
Execution Structures



- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$

- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- **π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}**
- π_3 = {(s1, move(r1,l1,l4))}

Execution Structures

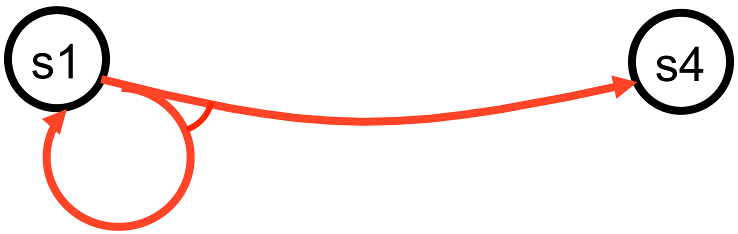


- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$

- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- **π_3 = {(s1, move(r1,l1,l4))}**

Execution Structures

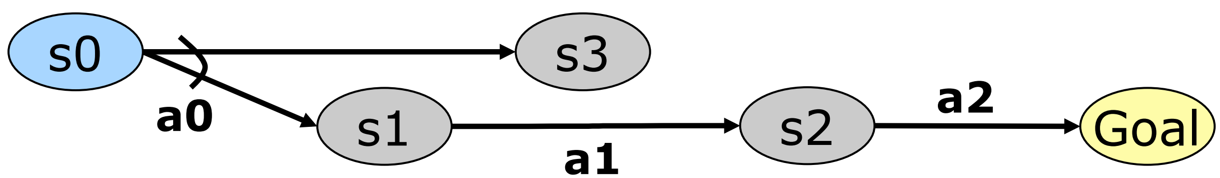
- *Execution structure* for a policy π:
  - The graph of all of π's execution paths
- Notation: $\Sigma_\pi = (Q,T)$
  - $Q \subseteq S$
  - $T \subseteq S \times S$



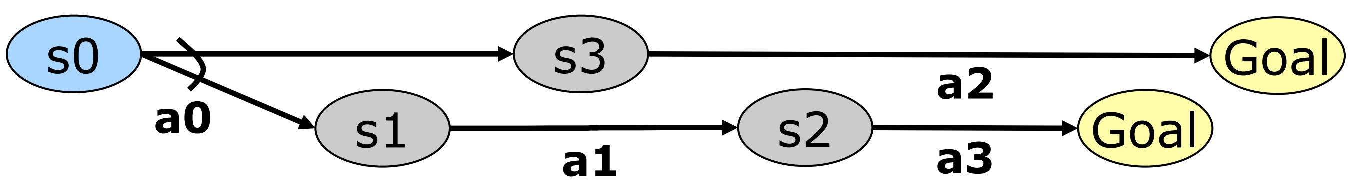
- π_1 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4))}
- π_2 = {(s1, move(r1,l1,l2)), (s2, move(r1,l2,l3)), (s3, move(r1,l3,l4)), (s5, move(r1,l3,l4))}
- **π_3 = {(s1, move(r1,l1,l4))}**

Types of Solutions

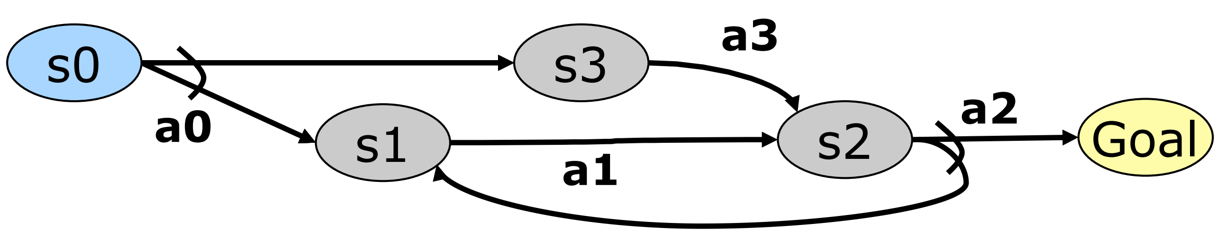
- **Weak solution**: at least one execution path reaches a goal



- **Strong solution**: every execution path reaches a goal



- **Strong-cyclic solution**: every *fair* execution path reaches a goal
  - Don't stay in a cycle forever if there's a state-transition out of it



Finding Strong Solutions

- Backward breadth-first search

- StrongPreImg(S)
  $= \{(s,a) : \gamma(s,a) \neq \varnothing, \gamma(s,a) \subseteq S\}$
  - all state-action pairs for which all of the successors are in S

- PruneStates(π,S)
  $= \{(s,a) \in \pi : s \notin S\}$
  - S is the set of states we've already solved
  - keep only the state-action pairs for other states

Strong-Plan(P)
    $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
    While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
        $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
        $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
        $\pi \leftarrow \pi'$
        $\pi' \leftarrow \pi' \cup \pi''$
    if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
        else return(failure)
end

- MkDet(π')
  - π' is a policy that may be nondeterministic
  - remove some state-action pairs if necessary, to get a deterministic policy

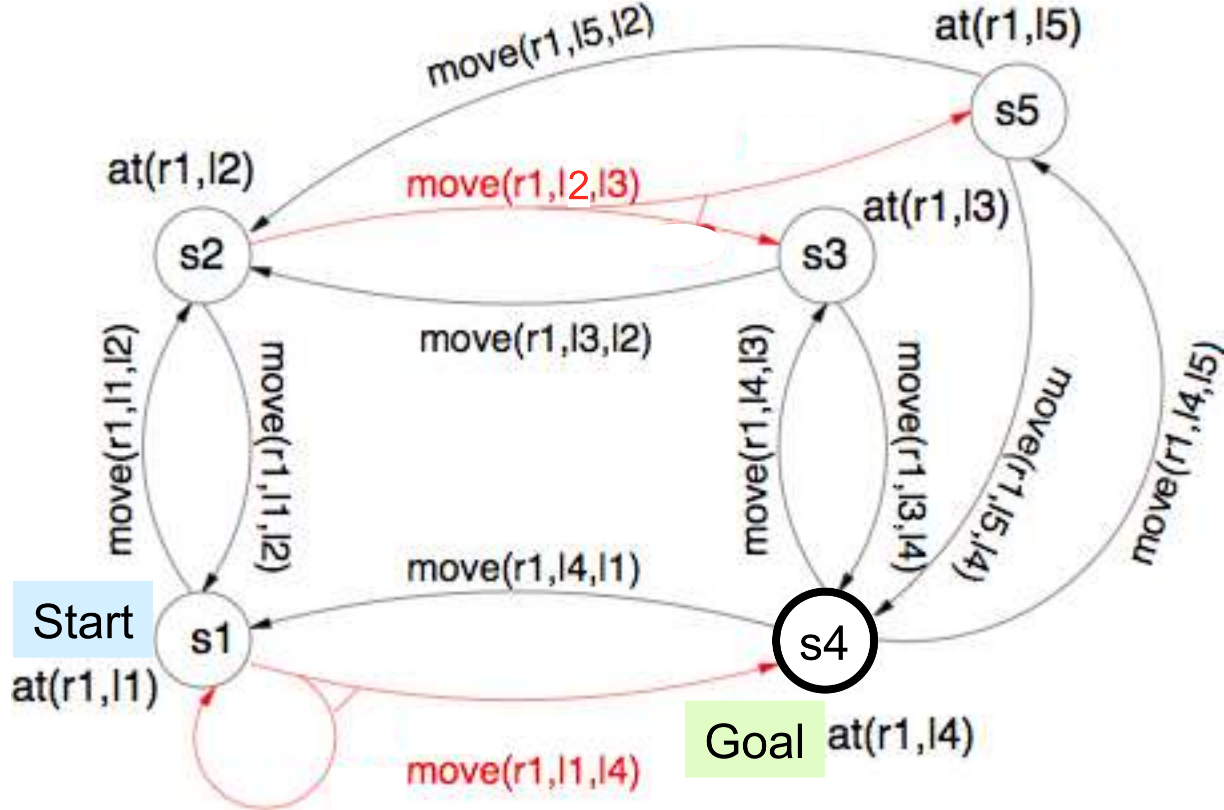
Example

π = failure

$\pi' = \varnothing$

$S_{\pi'} = \varnothing$

$S_g \cup S_{\pi'} = \{s4\}$



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg$(S_g \cup S_{\pi'})$
    $\pi'' \leftarrow$ PruneStates$(PreImage, S_g \cup S_{\pi'})$
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

Example

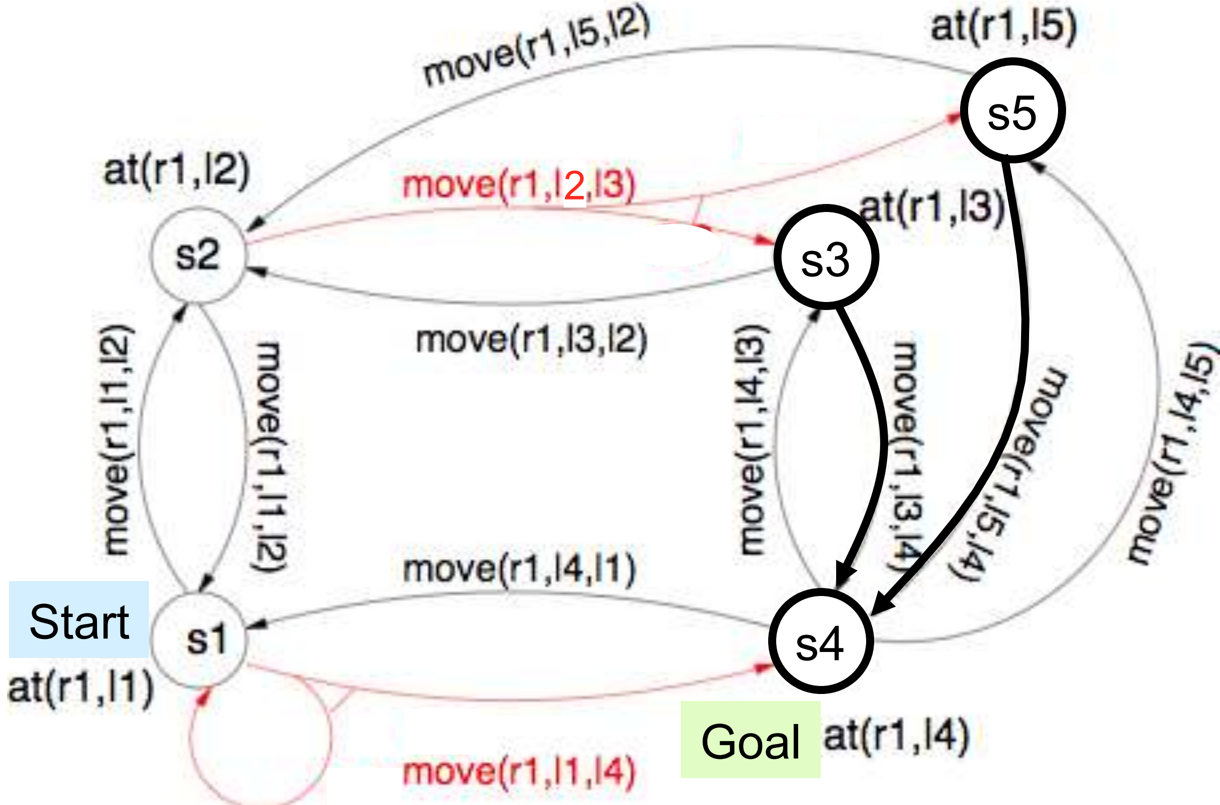
$\pi = \text{failure}$

$\pi' = \varnothing$

$S_{\pi'} = \varnothing$

$S_g \cup S_{\pi'} = \{\text{s4}\}$

$\pi'' \leftarrow PreImage =$
$\{(\text{s3,move(r1,l3,l4)}), (\text{s5,move(r1,l5,l4)})\}$



Strong-Plan(P)
    $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
    While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
        $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
        $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
        $\pi \leftarrow \pi'$
        $\pi' \leftarrow \pi' \cup \pi''$
    if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
        else return(failure)
end

Example

~~π = failure~~

~~$\pi' = \varnothing$~~

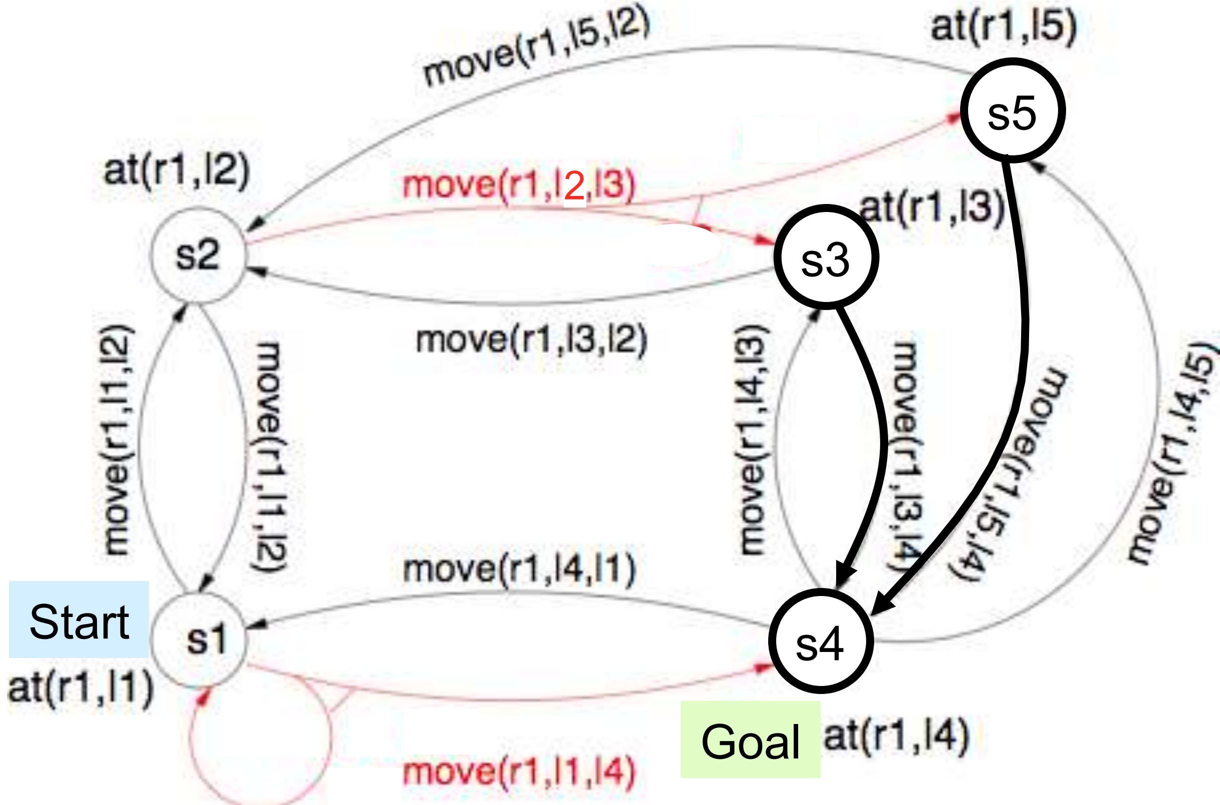
$S_{\pi'} = \varnothing$

$S_g \cup S_{\pi'} = \{s4\}$

$\pi'' \leftarrow PreImage =$
$\{(s3, move(r1,l3,l4)), (s5, move(r1,l5,l4))\}$

$\pi \leftarrow \pi' = \varnothing$

$\pi' \leftarrow \pi' \cup \pi'' =$
$\{(s3, move(r1,l3,l4)), (s5, move(r1,l5,l4))\}$



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
    $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

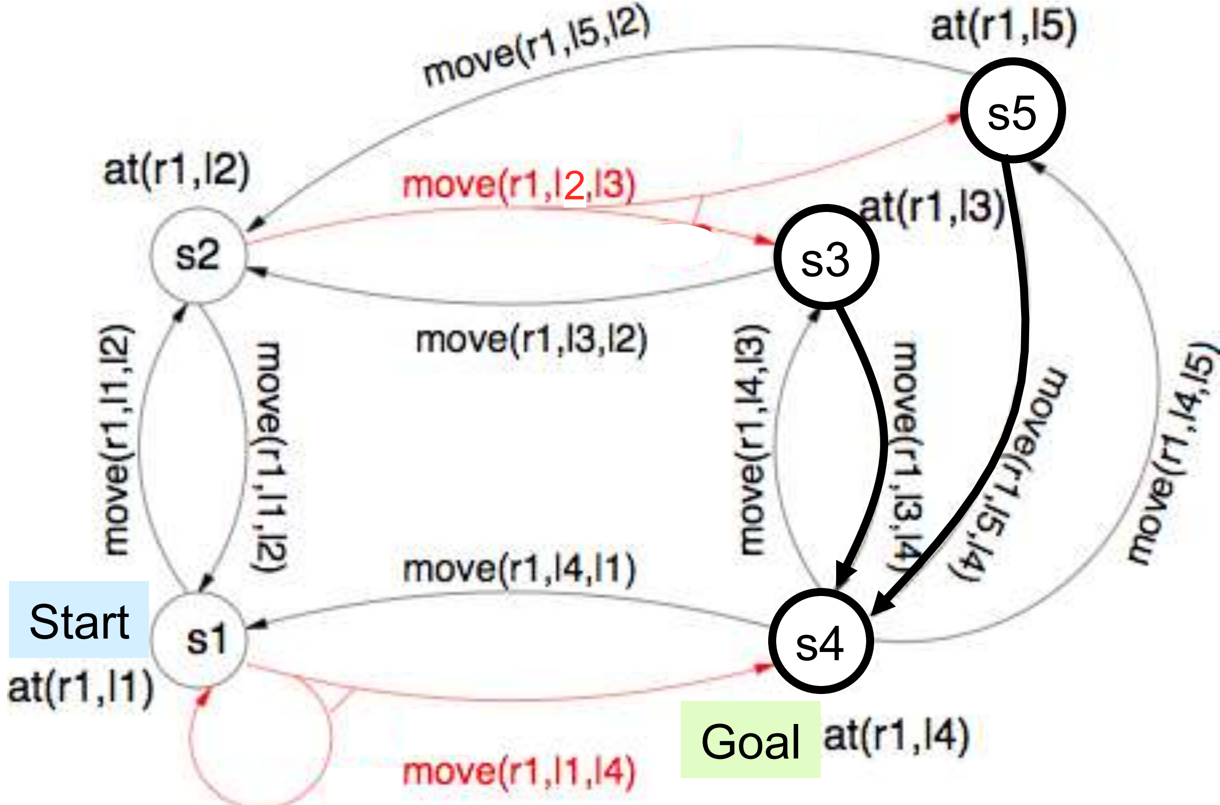
Example

$\pi = \varnothing$

$\pi' = \{(s3,move(r1,l3,l4)),$
$\quad (s5,move(r1,l5,l4))\}$

$S_{\pi'} = \{s3,s5\}$

$S_g \cup S_{\pi'} = \{s3,s4,s5\}$



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
    $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

Example

$\pi = \varnothing$

$\pi' = \{$(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$S_{\pi'} = \{$s3,s5$\}$

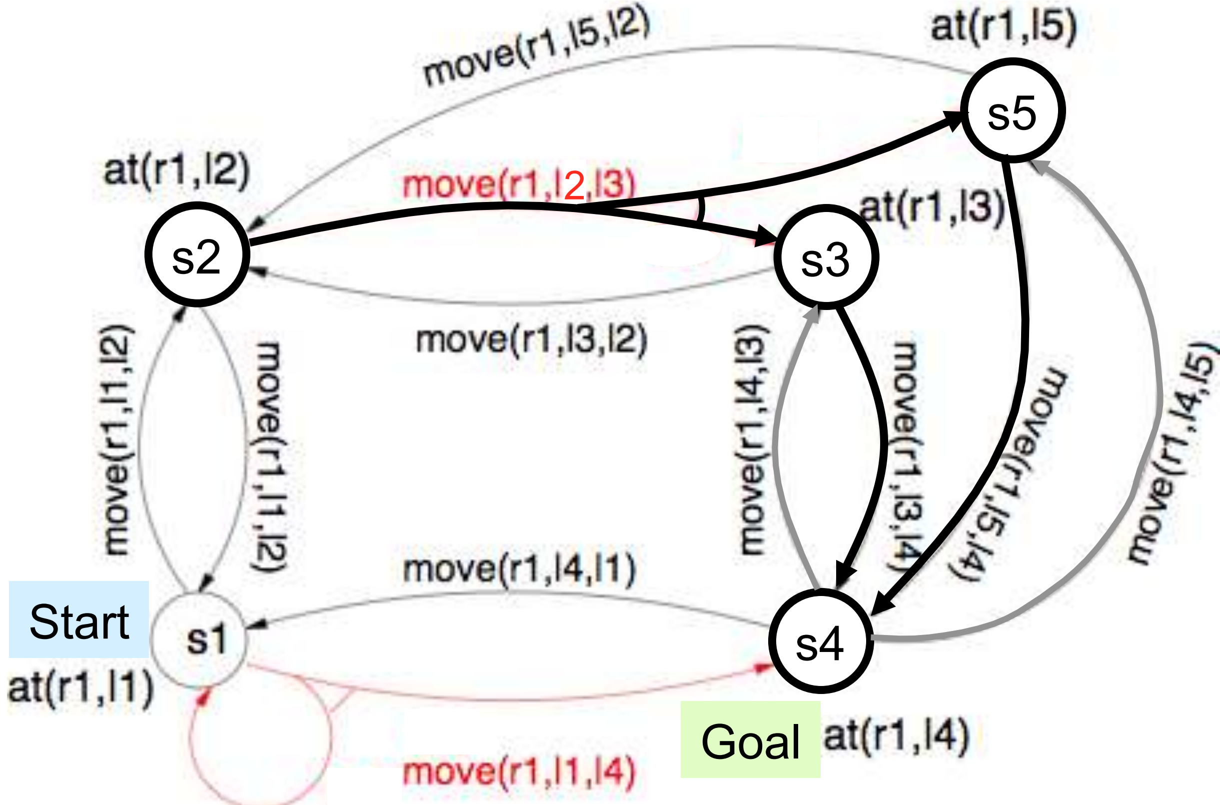
$S_g \cup S_{\pi'} = \{$s3,s4,s5$\}$

$PreImage \leftarrow$
$\{$(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4)),
(s3,move(r1,l4,l3)),
(s5,move(r1,l4,l5))$\}$

$\pi'' \leftarrow \{$(s2,move(r1,l2,l3))$\}$

$\pi \leftarrow \pi' = \{$(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$\pi' \leftarrow \{$(s2,move(r1,l2,l3),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$



Strong-Plan(P)
$\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
$PreImage \leftarrow$ StrongPreImg$(S_g \cup S_{\pi'})$
$\pi'' \leftarrow$ PruneStates$(PreImage, S_g \cup S_{\pi'})$
$\pi \leftarrow \pi'$
$\pi' \leftarrow \pi' \cup \pi''$
if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
else return(failure)
end

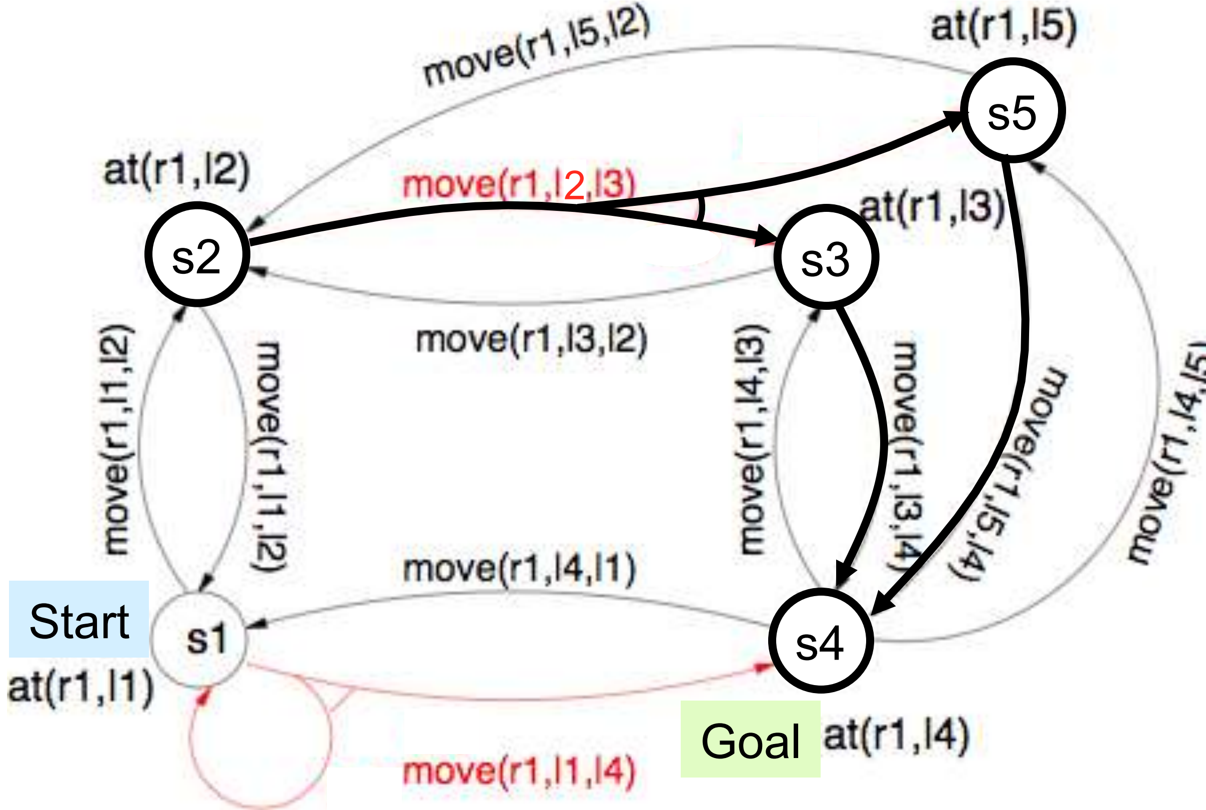
Example

$\pi = \{(\text{s3,move(r1,l3,l4)}), (\text{s5,move(r1,l5,l4)})\}$

$\pi' = \{(\text{s2,move(r1,l2,l3)}), (\text{s3,move(r1,l3,l4)}), (\text{s5,move(r1,l5,l4)})\}$

$S_{\pi'} = \{\text{s2,s3,s5}\}$

$S_g \cup S_{\pi'} = \{\text{s2,s3,s4,s5}\}$



Strong-Plan(P)
 $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
 While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
  $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
  $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
  $\pi \leftarrow \pi'$
  $\pi' \leftarrow \pi' \cup \pi''$
 if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
  else return(failure)
end

Example

$\pi = \{$(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$\pi' = \{$(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

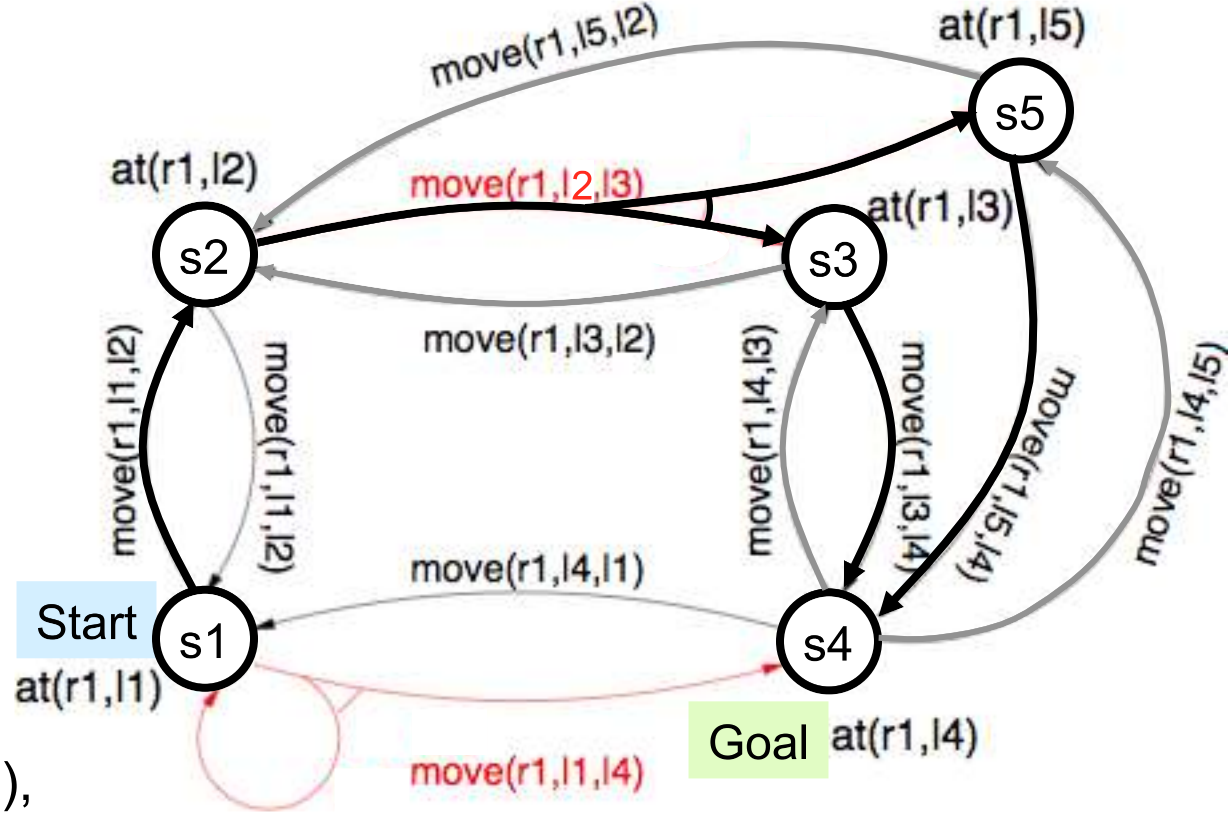
$S_{\pi'} = \{$s2,s3,s5$\}$

$S_g \cup S_{\pi'} = \{$s2,s3,s4,s5$\}$

$\pi'' \leftarrow \{$(s1,move(r1,l1,l2))$\}$

$\pi \leftarrow \pi' = \{$(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$\pi' \leftarrow \{$(s1,move(r1,l1,l2)),
(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
    $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

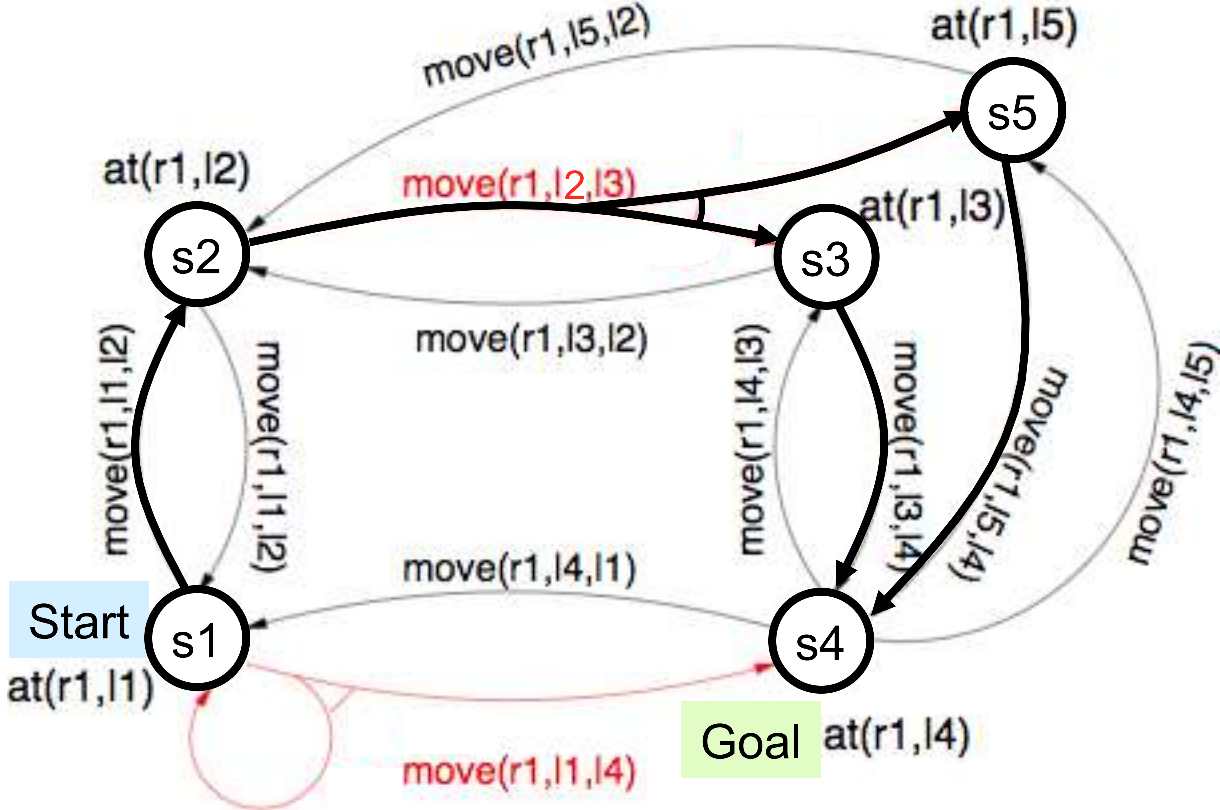
Example

$\pi = \{$(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$\pi' = \{$(s1,move(r1,l1,l2)),
(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$S_{\pi'} = \{$s1,s2,s3,s5$\}$

$S_g \cup S_{\pi'} = \{$s1,s2,s3,s4,s5$\}$



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
    $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

Example

$\pi = \{$(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

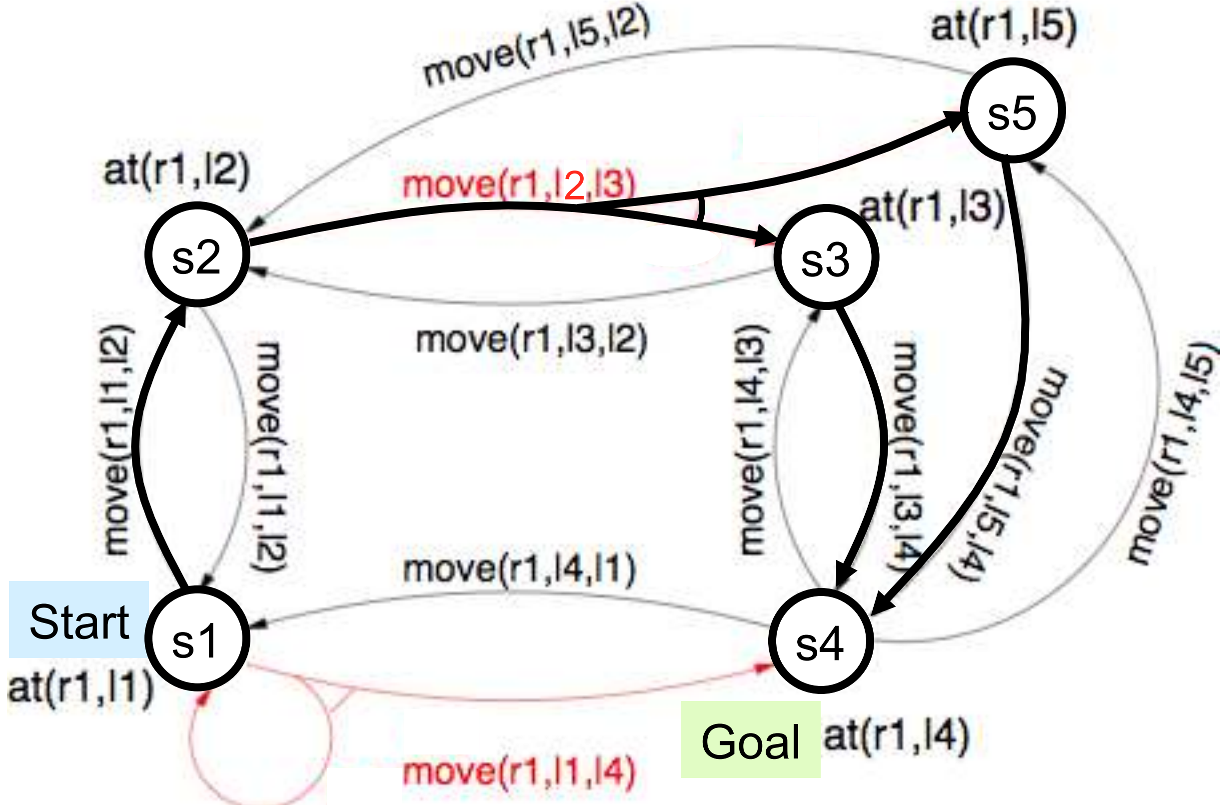
$\pi' = \{$(s1,move(r1,l1,l2)),
(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$S_{\pi'} = \{$s1,s2,s3,s5$\}$

$S_g \cup S_{\pi'} = \{$s1,s2,s3,s4,s5$\}$

$S_0 \subseteq S_g \cup S_{\pi'}$

MkDet(π') = π'



Strong-Plan(P)
  $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
  While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    $PreImage \leftarrow$ StrongPreImg($S_g \cup S_{\pi'}$)
    $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow \pi' \cup \pi''$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    else return(failure)
end

Finding Weak Solutions

- Weak-Plan is just like Strong-Plan except for this:

- WeakPreImg(S) = $\{(s,a) : \gamma(s,a)\ \mathrm{i}\ S \neq \emptyset\}$
  - at least one successor is in S

```
Weak-Plan(P)
    π ← failure; π′ ← ∅
    While π′ ≠ π and S0 ⊈ (Sg ∪ Sπ′) do
        PreImage ← WeakPreImg(Sg ∪ Sπ′)
        π″ ← PruneStates(PreImage, Sg ∪ Sπ′)
        π ← π′
        π′ ← π′ ∪ π″
    if S0 ⊆ (Sg ∪ Sπ′) then return(MkDet(π′))
        else return(failure)
end
```

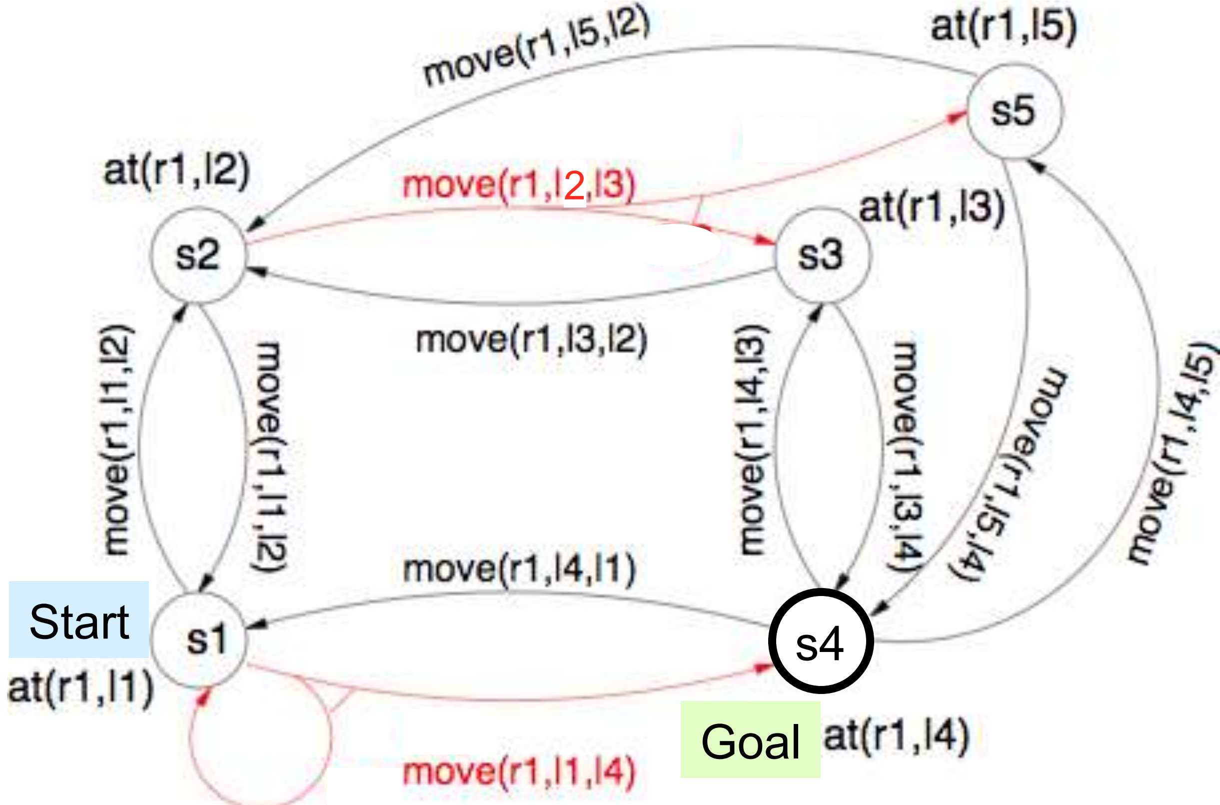
Example

π = failure

$\pi' = \varnothing$

$S_{\pi'} = \varnothing$

$S_g \cup S_{\pi'} = \{s4\}$



Weak-Plan(P)
- $\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
- While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
    - $PreImage \leftarrow$ WeakPreImg($S_g \cup S_{\pi'}$)
    - $\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
    - $\pi \leftarrow \pi'$
    - $\pi' \leftarrow \pi' \cup \pi''$
- if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
    - else return(failure)

end

Example

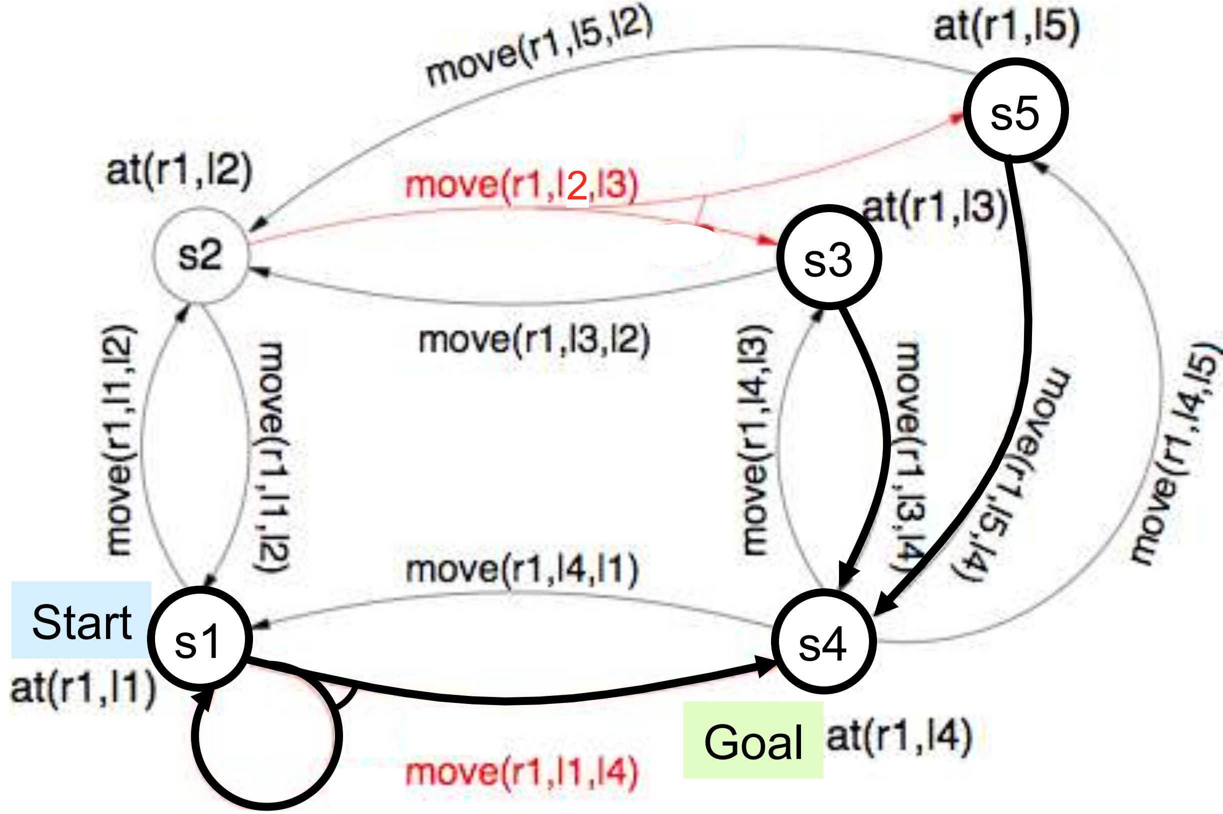
π = failure

$\pi' = \varnothing$

$S_{\pi'} = \varnothing$

$S_g \cup S_{\pi'} = \{s4\}$

$\pi'' = PreImage =$
{(s1,move(r1,l1,l4)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))}



$\pi \leftarrow \pi' = \varnothing$

$\pi' \leftarrow \pi' \cup \pi'' =$
{(s1,move(r1,l1,l4)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))}

Weak-Plan(P)
$\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
$PreImage \leftarrow$ WeakPreImg$(S_g \cup S_{\pi'})$
$\pi'' \leftarrow$ PruneStates$(PreImage, S_g \cup S_{\pi'})$
$\pi \leftarrow \pi'$
$\pi' \leftarrow \pi' \cup \pi''$
if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
else return(failure)
end

Example

$\pi = \varnothing$

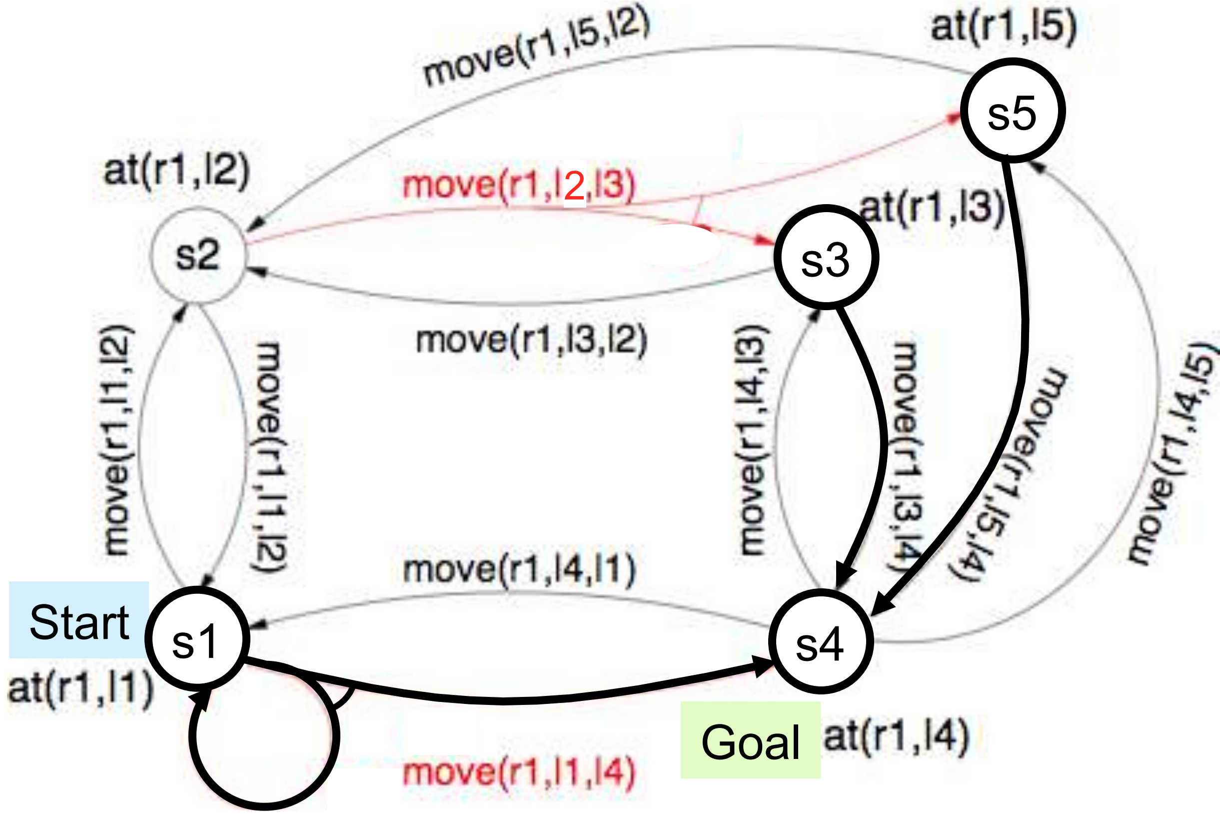
$\pi' = \{$(s1,move(r1,l1,l4)),
(s3,move(r1,l3,l4)),
(s5,move(r1,l5,l4))$\}$

$S_{\pi'} = \{$s1,s3,s5$\}$

$S_g \cup S_{\pi'} = \{$s1,s3,s4,s5$\}$

$S_0 \subseteq S_g \cup S_{\pi'}$

MkDet(π') = π'



Weak-Plan(P)
$\pi \leftarrow$ failure; $\pi' \leftarrow \emptyset$
While $\pi' \neq \pi$ and $S_0 \not\subseteq (S_g \cup S_{\pi'})$ do
$PreImage \leftarrow$ WeakPreImg($S_g \cup S_{\pi'}$)
$\pi'' \leftarrow$ PruneStates($PreImage$, $S_g \cup S_{\pi'}$)
$\pi \leftarrow \pi'$
$\pi' \leftarrow \pi' \cup \pi''$
if $S_0 \subseteq (S_g \cup S_{\pi'})$ then return(MkDet(π'))
else return(failure)
end

Finding Strong-Cyclic Solutions

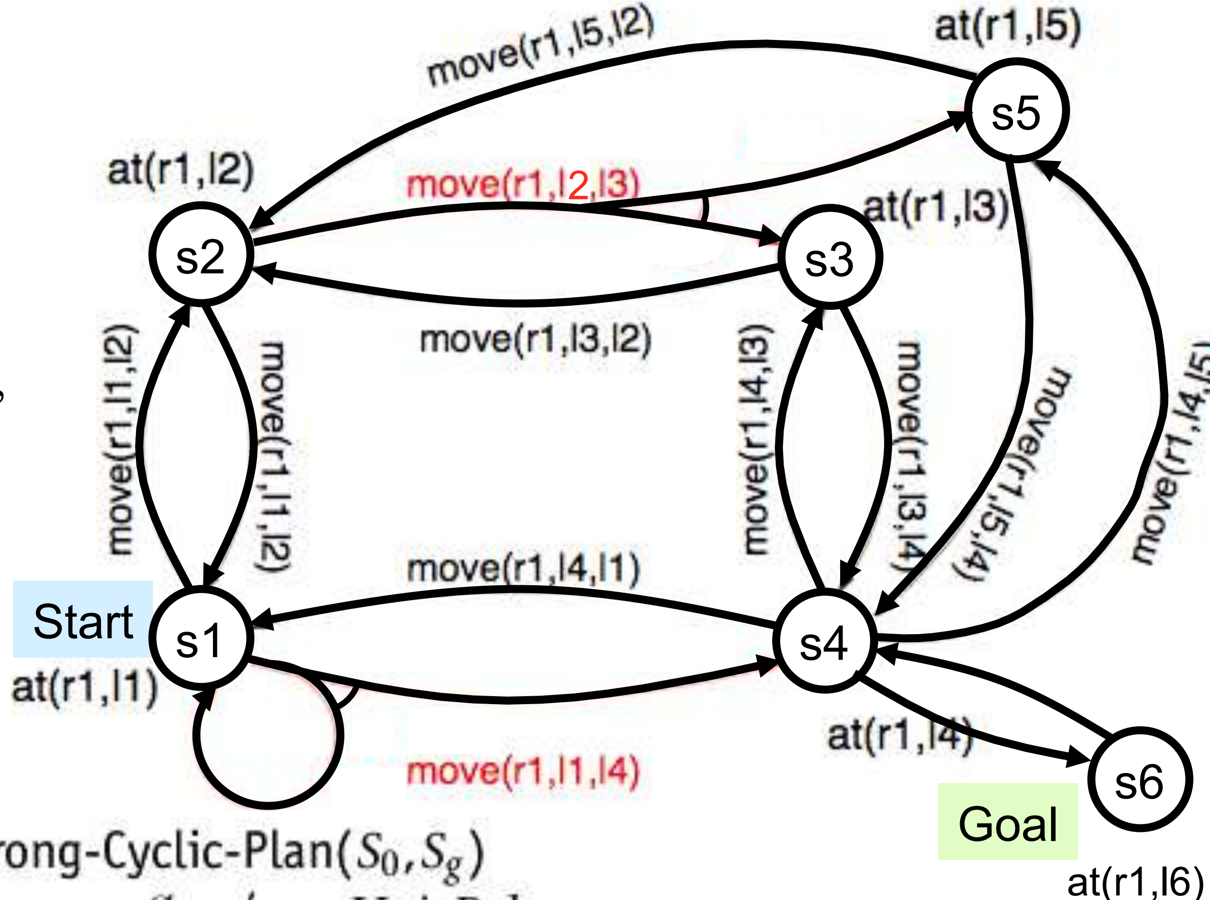
- Begin with a “universal policy” π' that contains *all* state-action pairs
- Repeatedly, eliminate state-action pairs that take us to bad states
  - PruneOutgoing removes state-action pairs that go to states not in $S_g \cup S_\pi$
    - » PruneOutgoing$(\pi, S) = \pi - \{(s,a) \in \pi : \gamma(s,a) \subseteq S \cup S_\pi$
  - PruneUnconnected removes states from which it is impossible to get to S_g
    - » Start with $\pi' = \varnothing$, compute fixpoint of $\pi' \leftarrow \pi \cap$ WeakPreImg$(S_g \cup S_{\pi'})$

Strong-Cyclic-Plan(S_0, S_g)
  $\pi \leftarrow \emptyset;\ \pi' \leftarrow UnivPol$
  while $\pi' \neq \pi$ do
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing$(\pi', S_g), S_g)$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$
    then return(MkDet(RemoveNonProgress(π', S_g))))
    else return(failure)
end

Finding Strong-Cyclic Solutions

Once the policy stops changing,

- If it's not a solution, return failure
- RemoveNonProgress removes state-action pairs that don't go toward the goal
  - implement as backward search from the goal
- MkDet makes sure there's only one action for each state

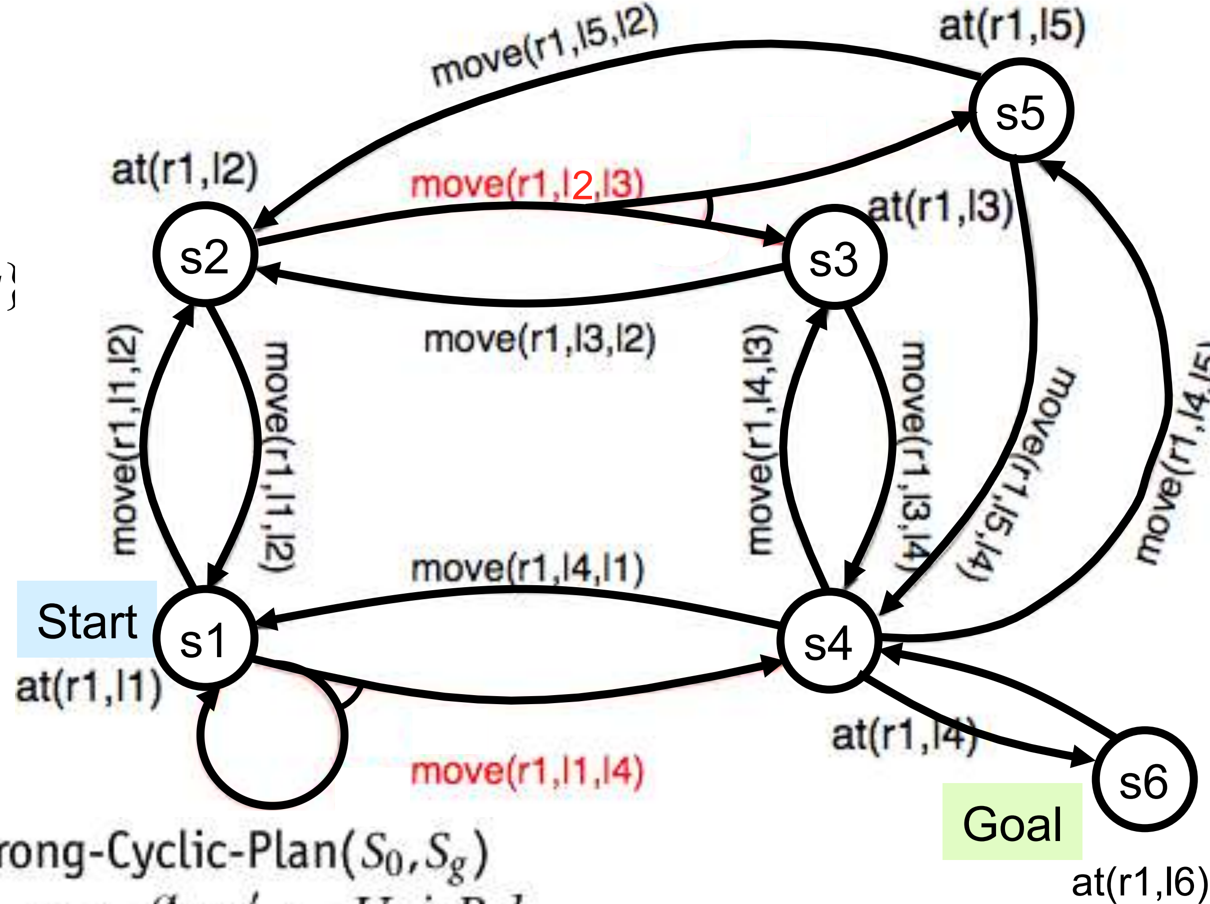


Strong-Cyclic-Plan(S_0,S_g)
    $\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$
    while $\pi' \neq \pi$ do
        $\pi \leftarrow \pi'$
        $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π',S_g),S_g)
    if $S_0 \subseteq (S_g \cup S_{\pi'})$
        then return(MkDet(RemoveNonProgress(π',S_g)))
        else return(failure)
end

Example 1

$\pi \leftarrow \varnothing$

$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$



Strong-Cyclic-Plan(S_0,S_g)
  $\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$
  while $\pi' \neq \pi$ do
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π',S_g),S_g)
  if $S_0 \subseteq (S_g \cup S_{\pi'})$
    then return(MkDet(RemoveNonProgress(π',S_g)))
    else return(failure)
end

Example 1

~~$\pi \leftarrow \emptyset$~~

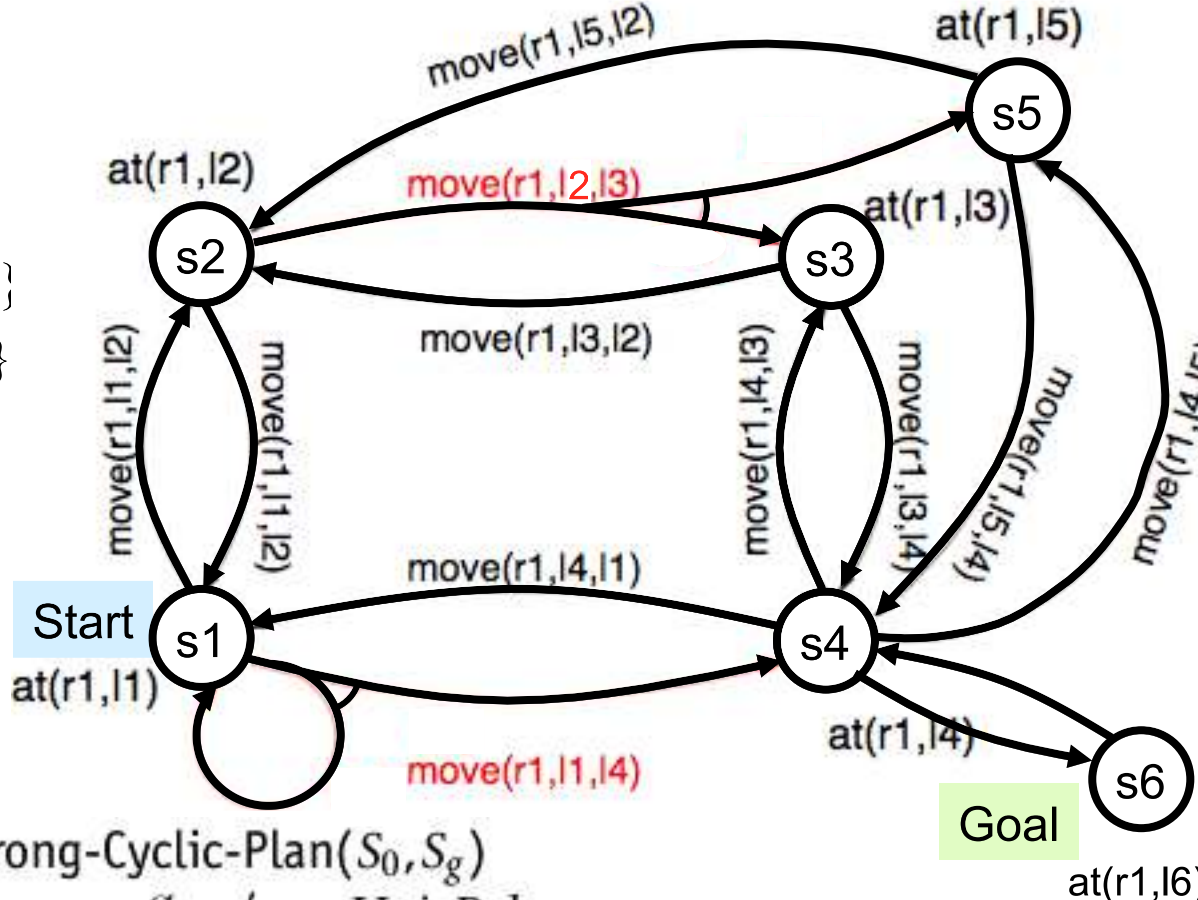
$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi', S_g) = \pi'$

$\text{PruneUnconnected}(\pi', S_g) = \pi'$

$\text{RemoveNonProgress}(\pi') = ?$



Strong-Cyclic-Plan(S_0, S_g)

$\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$

while $\pi' \neq \pi$ do

$\pi \leftarrow \pi'$

$\pi' \leftarrow$ PruneUnconnected(PruneOutGoing$(\pi', S_g), S_g)$

if $S_0 \subseteq (S_g \cup S_{\pi'})$

then return(MkDet(RemoveNonProgress(π', S_g)))

else return(failure)

end

Example 1

~~$\pi \leftarrow \varnothing$~~

$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

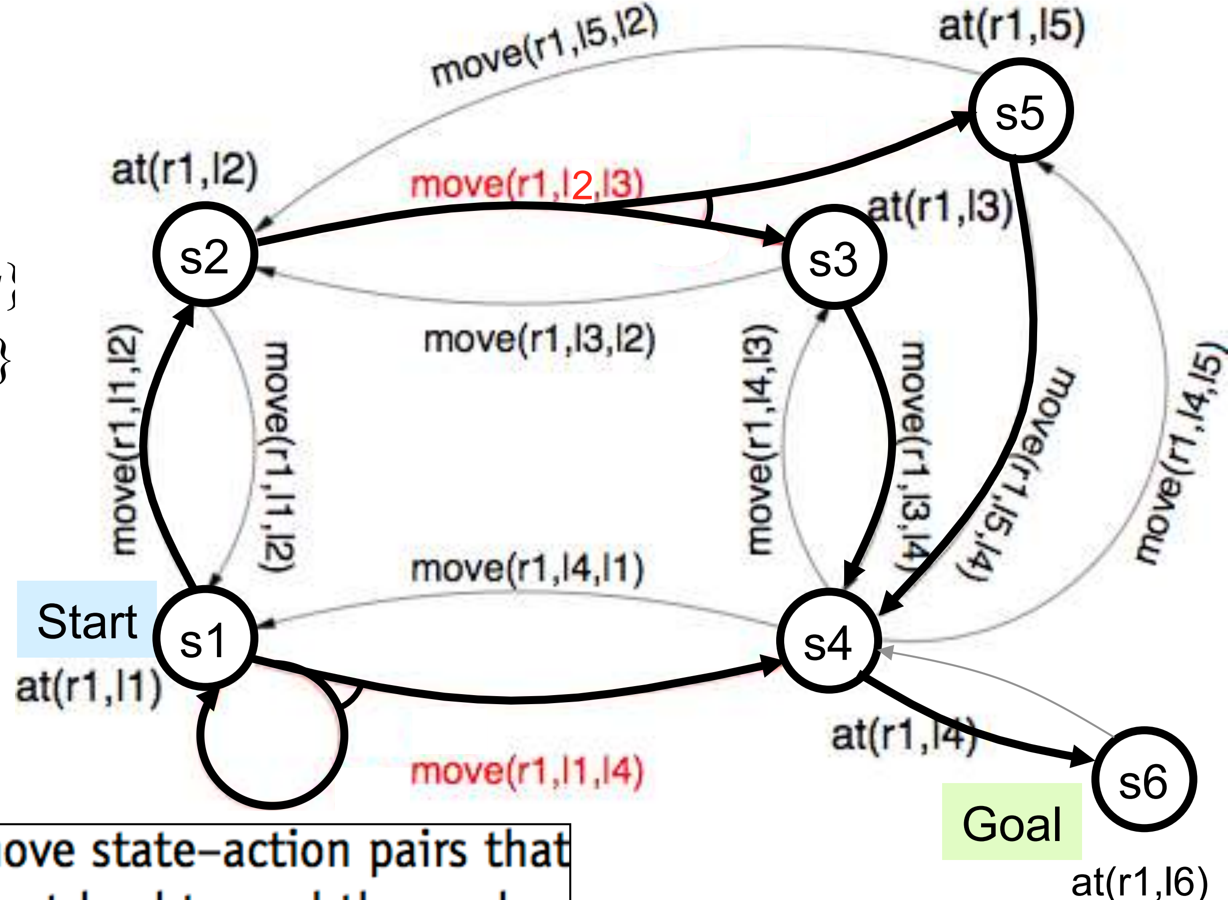
$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi',S_g) = \pi'$

$\text{PruneUnconnected}(\pi',S_g) = \pi'$

$\text{RemoveNonProgress}(\pi') =$

as shown



RemoveNonProgress(π, S_g) ;; remove state–action pairs that
;; do not lead toward the goal

$\pi^* \leftarrow \emptyset$

repeat

$\quad PreImage \leftarrow \pi \cap \text{WeakPreImg}(S_g \cup S_{\pi^*})$

$\quad \pi^*_{old} \leftarrow \pi^*$

$\quad \pi^* \leftarrow \pi^* \cup \text{PruneStates}(PreImage, S_g \cup S_{\pi^*})$

until $\pi^*_{old} = \pi^*$

return(π^*)

end

$\text{PruneStates}(\pi, S) = \{(s,a) \in \pi \mid s \notin S\}$

Example 1

~~$\pi \leftarrow \varnothing$~~

$\pi' \leftarrow \{(s,a) : a$ is applicable to $s\}$

$\pi \leftarrow \{(s,a) : a$ is applicable to $s\}$

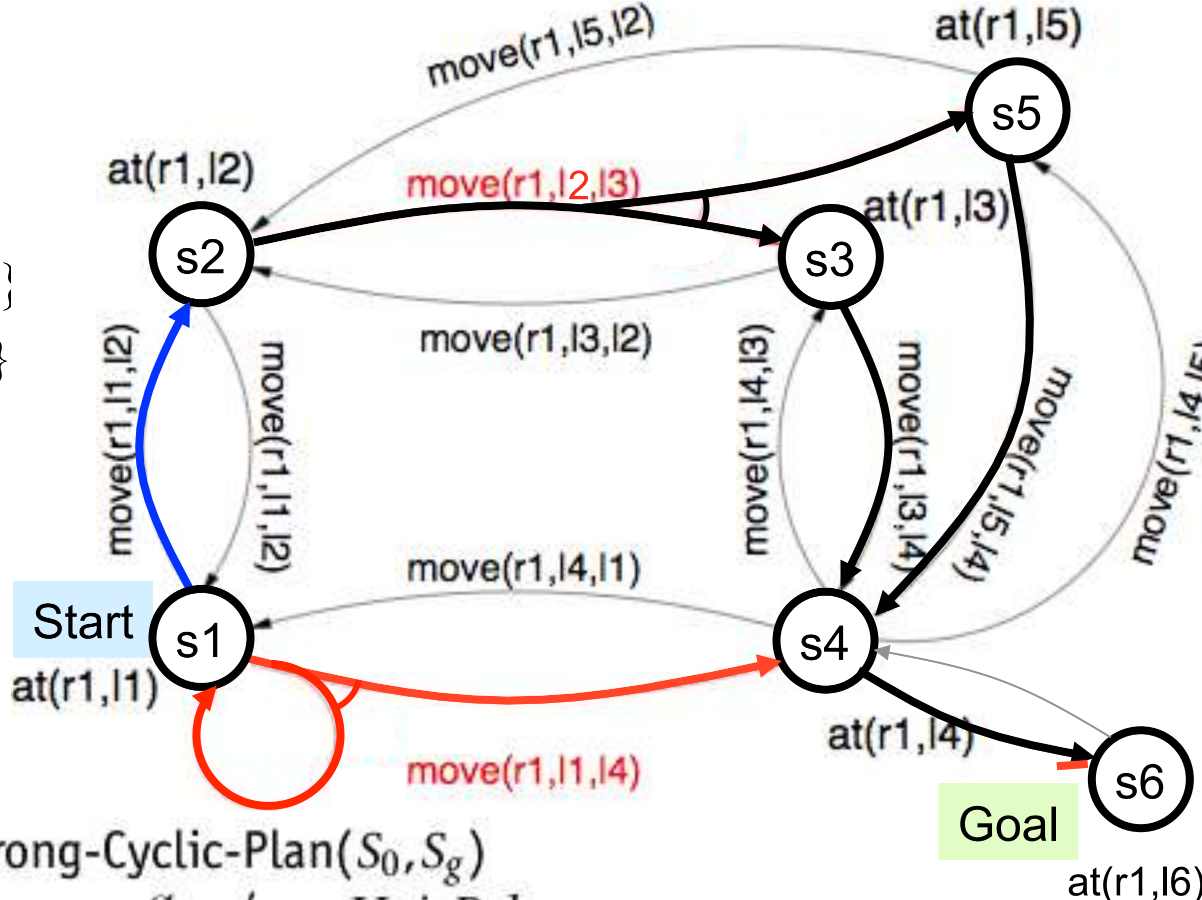
PruneOutgoing$(\pi', S_g) = \pi'$

PruneUnconnected$(\pi', S_g) = \pi'$

RemoveNonProgress(π') =
as shown

MkDet(...) = either
{(s1,move(r1,l1,l4),
(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4),
(s4,move(r1,l4,l6),
(s5,move(r1,l5,l4)}

or {(s1,move(r1,l1,l2),
(s2,move(r1,l2,l3)),
(s3,move(r1,l3,l4),
(s4,move(r1,l4,l6),
(s5,move(r1,l5,l4)}

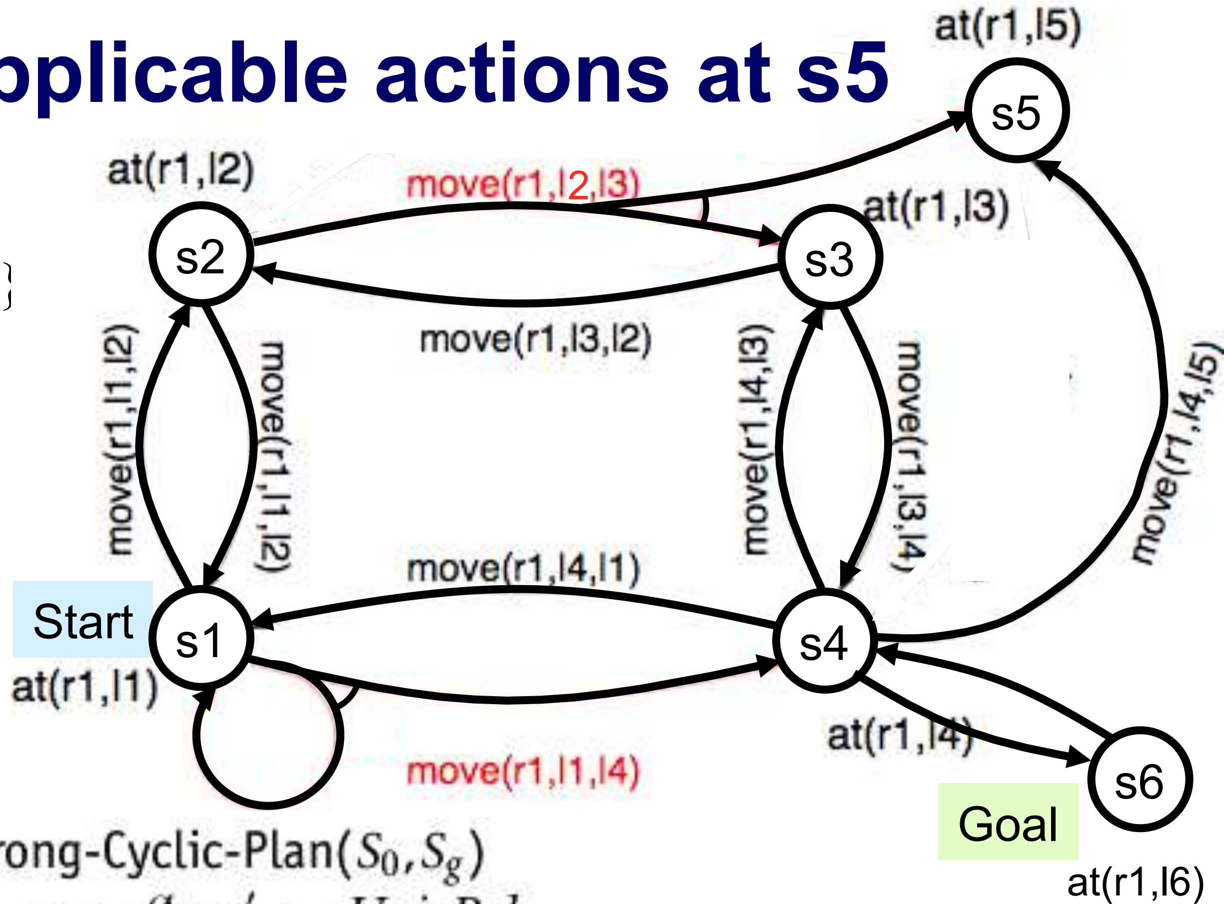


Strong-Cyclic-Plan(S_0, S_g)
  $\pi \leftarrow \emptyset;\ \pi' \leftarrow UnivPol$
  while $\pi' \neq \pi$ do
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing$(\pi', S_g), S_g)$
  if $S_0 \subseteq (S_g \cup S_{\pi'})$
    then return(MkDet(RemoveNonProgress(π', S_g)))
    else return(failure)
end

Example 2: no applicable actions at s5

$\pi \leftarrow \varnothing$

$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$



Strong-Cyclic-Plan(S_0, S_g)

$\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$

while $\pi' \neq \pi$ do

$\pi \leftarrow \pi'$

$\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π', S_g), S_g)

if $S_0 \subseteq (S_g \cup S_{\pi'})$

then return(MkDet(RemoveNonProgress(π', S_g)))

else return(failure)

end

Example 2: no applicable actions at s5

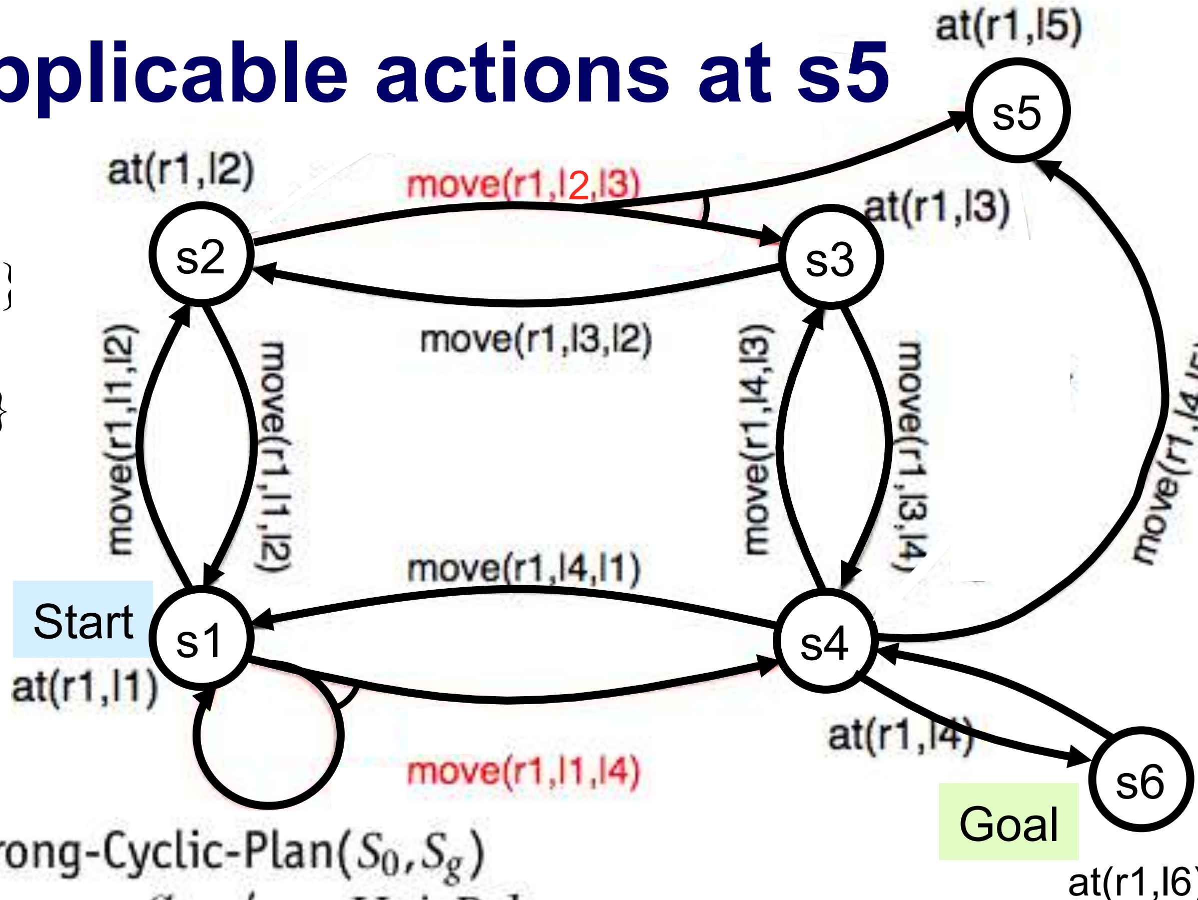
~~$\pi \leftarrow \emptyset$~~

$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi', S_g)$

$= \ldots$



Strong-Cyclic-Plan(S_0, S_g)

$\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$

while $\pi' \neq \pi$ do

$\pi \leftarrow \pi'$

$\pi' \leftarrow$ PruneUnconnected(PruneOutGoing$(\pi', S_g), S_g)$

if $S_0 \subseteq (S_g \cup S_{\pi'})$

then return(MkDet(RemoveNonProgress(π', S_g))))

else return(failure)

end

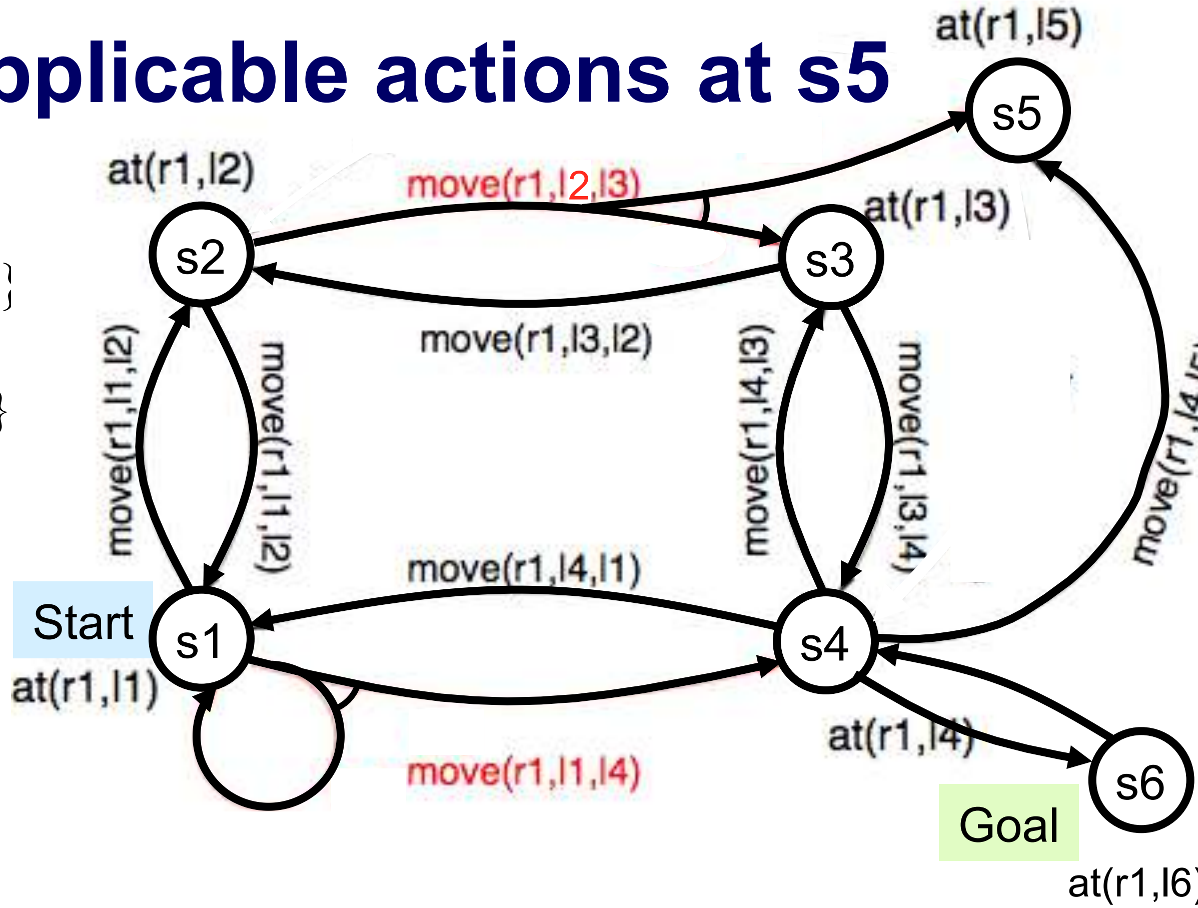
Example 2: no applicable actions at s5

~~$\pi \leftarrow \emptyset$~~

$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi', S_g) = \pi'$



PruneOutgoing(π, S_g) ;; removes outgoing state-action pairs

$\pi' \leftarrow \pi - \text{ComputeOutgoing}(\pi, S_g \cup S_\pi)$

return(π')

end

$$\text{ComputeOutgoing}(\pi, S) = \{(s,a) \in \pi \mid \gamma(s,a) \not\subseteq S\}$$

Example 2: no applicable actions at s5

~~$\pi \leftarrow \emptyset$~~

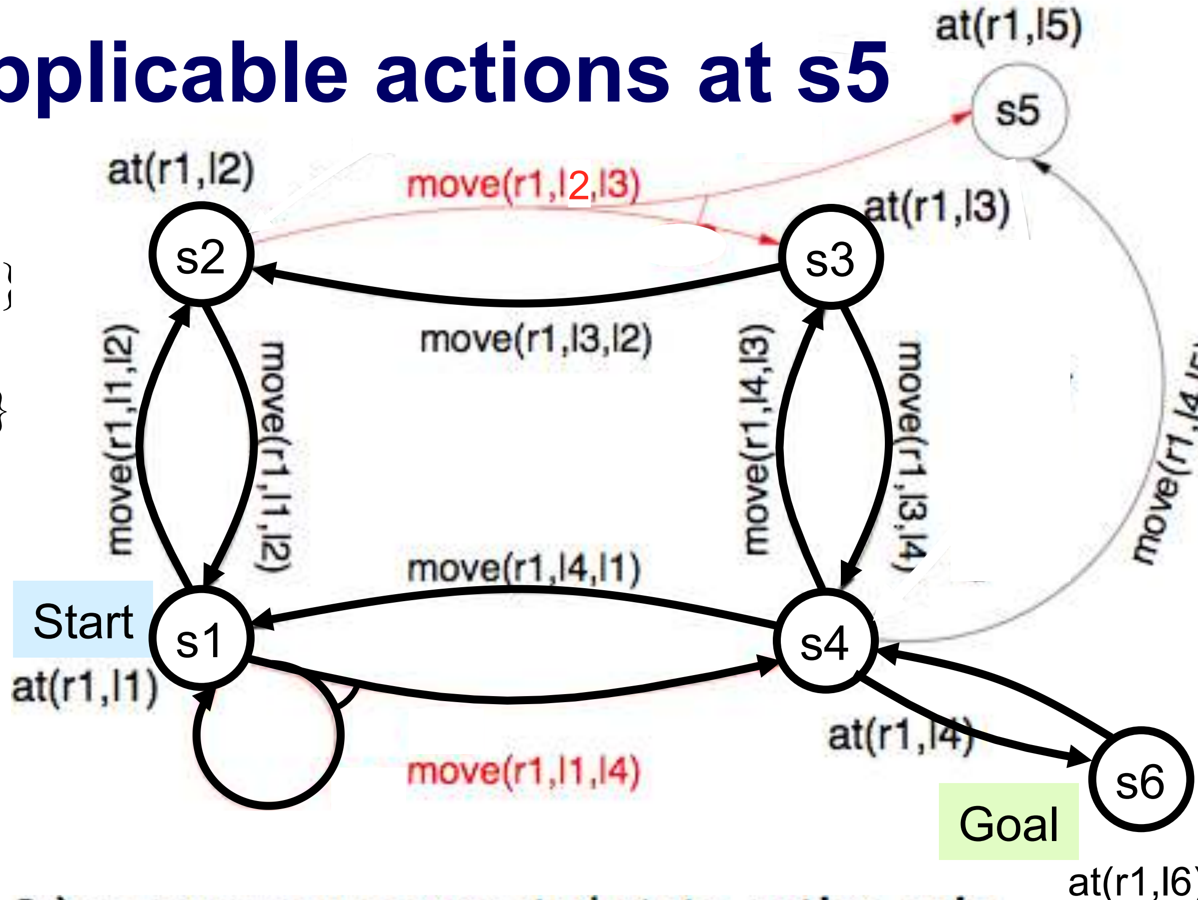
$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi', S_g) = \pi'$

$\text{PruneUnconnected}(\pi', S_g)$

$= \textit{as shown}$



PruneUnconnected(π, S_g) ;; removes unconnected state-action pairs

$\pi' \leftarrow \emptyset$

repeat

$\pi'' \leftarrow \pi'$

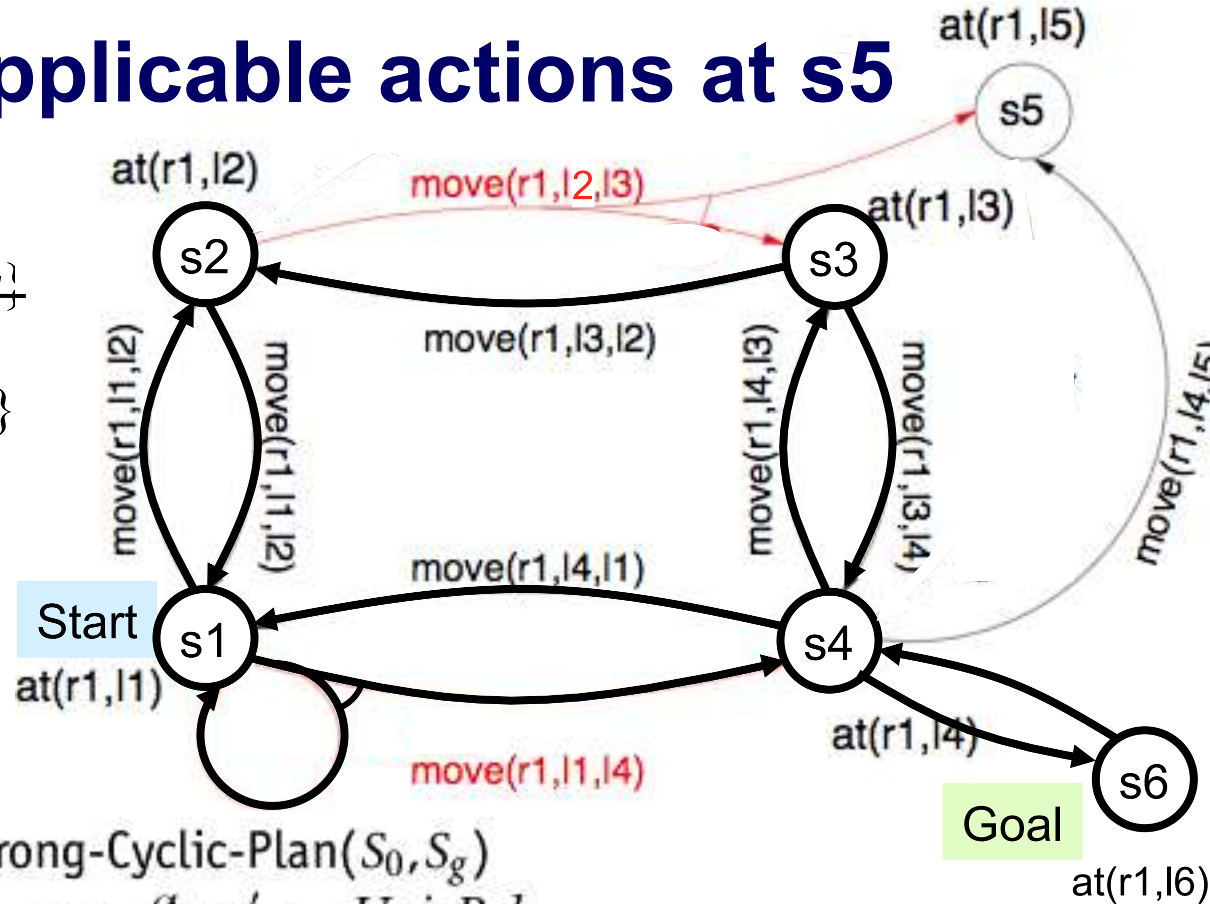
$\pi' \leftarrow \pi \cap \text{WeakPreImg}(S_g \cup S_{\pi'})$

until $\pi'' = \pi'$

return(π') end

Dana Nau: Lecture slides for *Automated Planning*

Example 2: no applicable actions at s5



~~$\pi \leftarrow \varnothing$~~

~~$\pi' \leftarrow \{(s,a) : a \text{ is applicable to } s\}$~~

$\pi \leftarrow \{(s,a) : a \text{ is applicable to } s\}$

$\text{PruneOutgoing}(\pi', S_g) = \pi'$

$\text{PruneUnconnected}(\pi', S_g)$

$= \textit{as shown}$

$\pi' \leftarrow \textit{as shown}$

Strong-Cyclic-Plan(S_0, S_g)
    $\pi \leftarrow \emptyset$; $\pi' \leftarrow \mathit{UnivPol}$
    while $\pi' \neq \pi$ do
        $\pi \leftarrow \pi'$
        $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π', S_g), S_g)
    if $S_0 \subseteq (S_g \cup S_{\pi'})$
        then return(MkDet(RemoveNonProgress(π', S_g)))
        else return(failure)
end

Example 2: no applicable actions at s5

$\pi' \leftarrow$ *as hown*

$\pi \leftarrow \pi'$

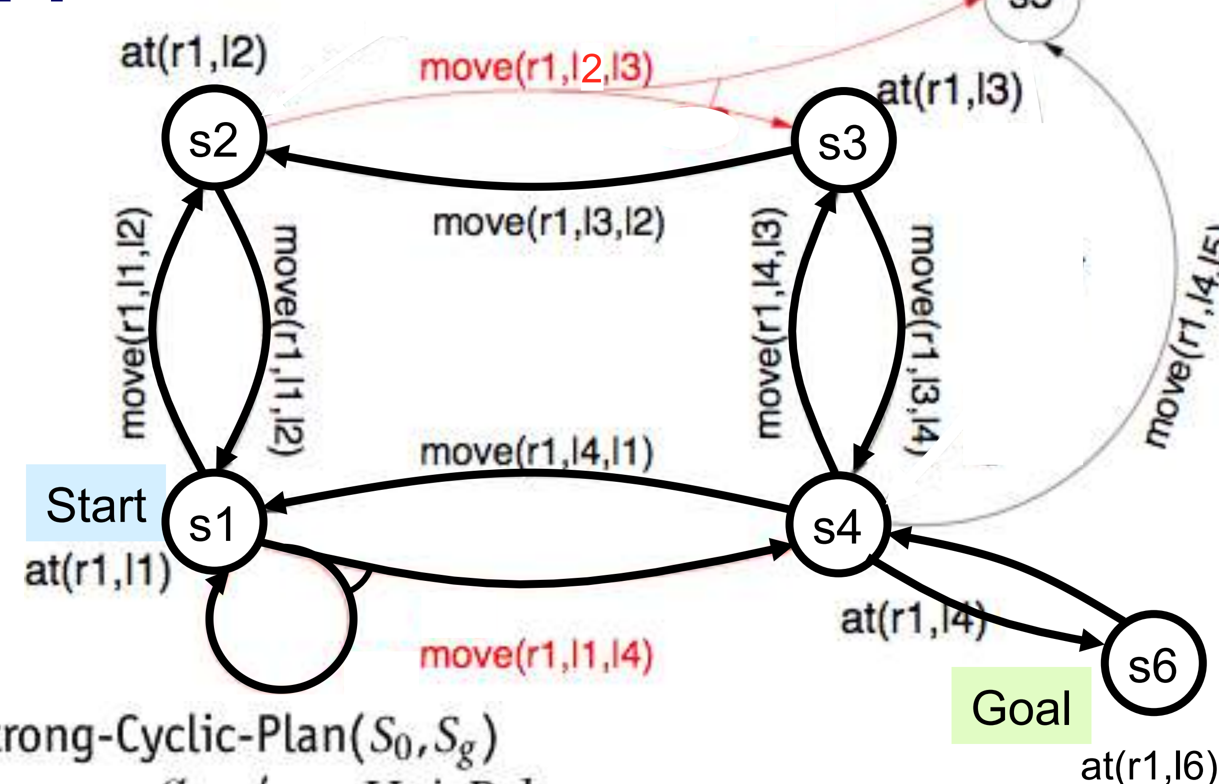
PruneOutgoing(π',S_g) = π'

PruneUnconnected(π',S_g) = π'

so $\pi = \pi'$

RemoveNonProgress(π') =

…



Strong-Cyclic-Plan(S_0,S_g)
  $\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$
  while $\pi' \neq \pi$ do
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π',S_g),S_g)
  if $S_0 \subseteq (S_g \cup S_{\pi'})$
    then return(MkDet(RemoveNonProgress(π',S_g)))
    else return(failure)
end

Example 2: no applicable actions at s5

~~$\pi' \leftarrow shown$~~

$\pi \leftarrow \pi'$

PruneOutgoing(π',S_g) = π'

PruneUnconnected(π'',S_g) = π'

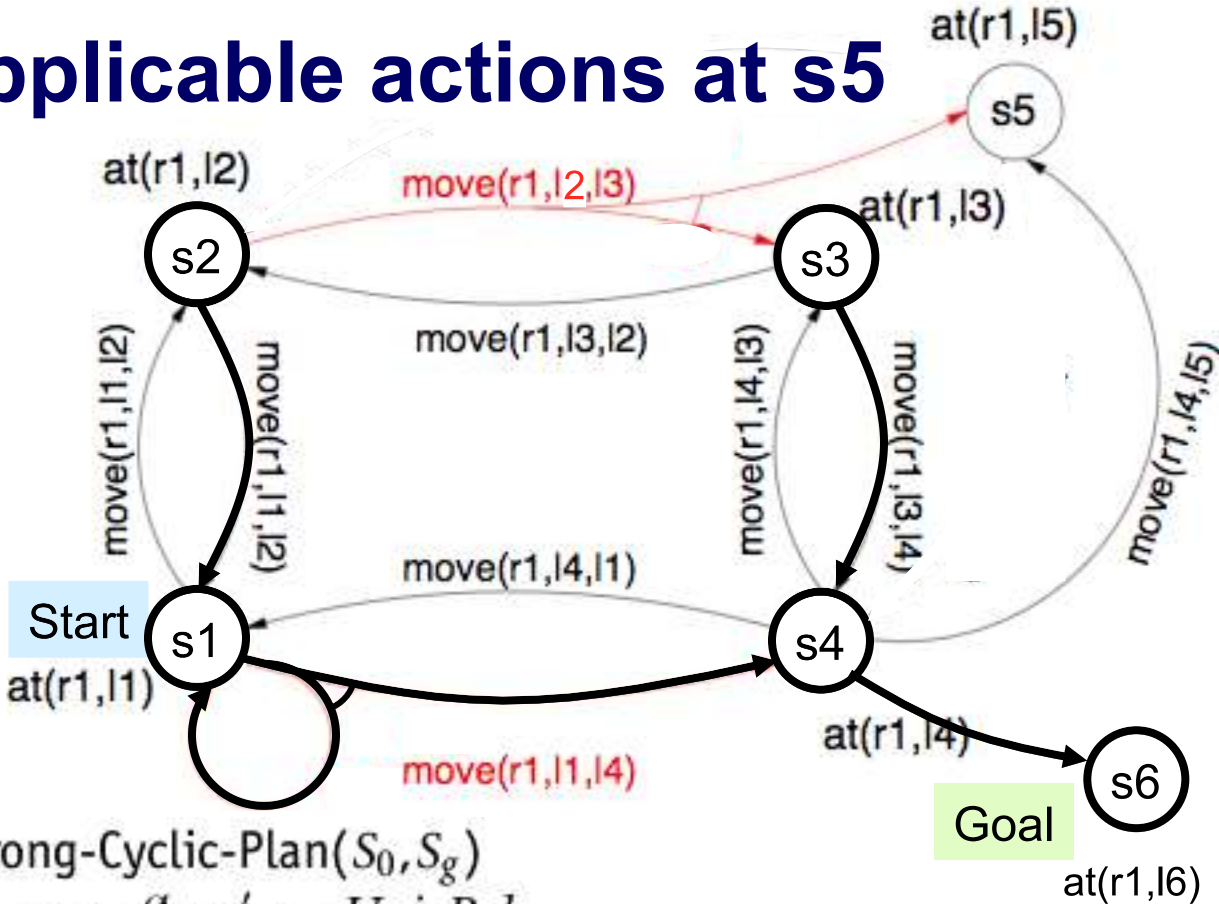
so $\pi = \pi'$

RemoveNonProgress(π') =

as shown

MkDet(*shown*) =

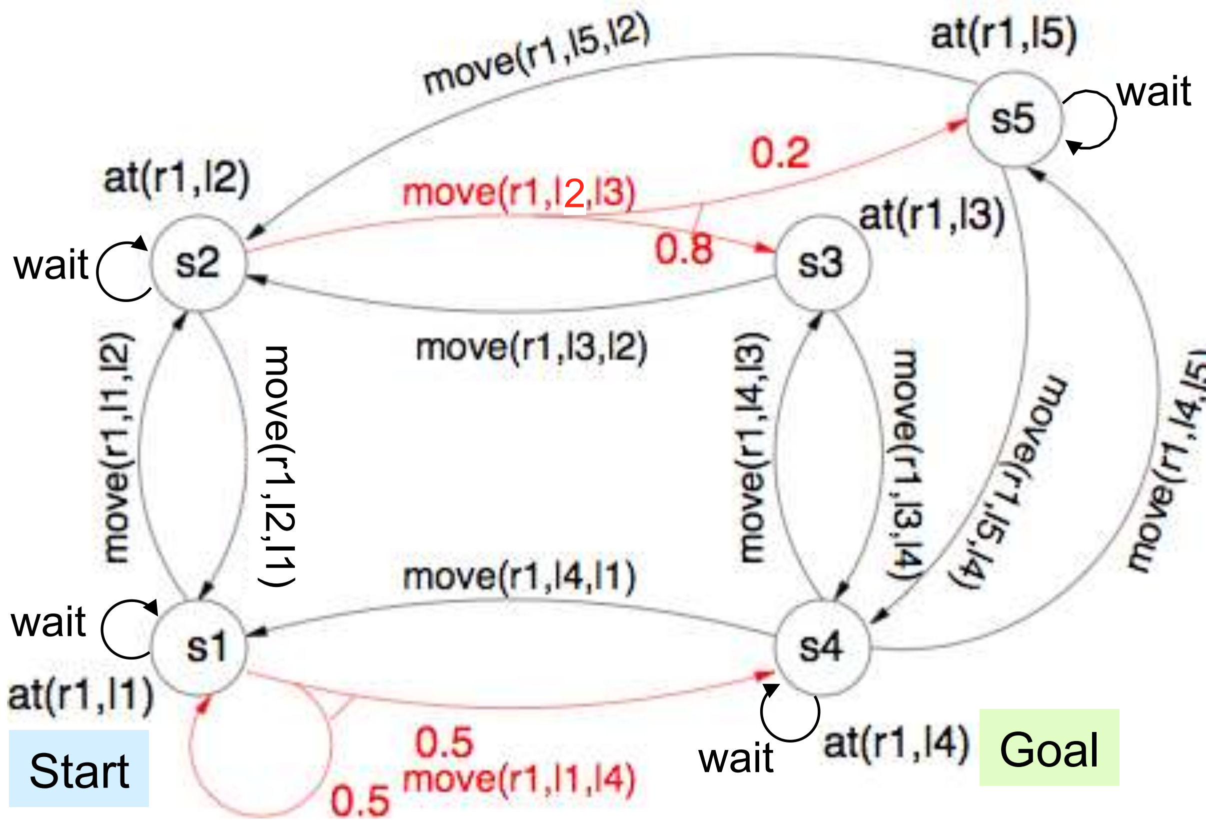
no change



Strong-Cyclic-Plan(S_0,S_g)
  $\pi \leftarrow \emptyset$; $\pi' \leftarrow UnivPol$
  while $\pi' \neq \pi$ do
    $\pi \leftarrow \pi'$
    $\pi' \leftarrow$ PruneUnconnected(PruneOutGoing(π',S_g),S_g)
  if $S_0 \subseteq (S_g \cup S_{\pi'})$
    then return(MkDet(RemoveNonProgress(π',S_g)))
    else return(failure)
end

Planning for Extended Goals

- Here, "extended" means *temporally extended*
  - Constraints that apply to some sequence of states
- Examples:
  - want to move to l3, and then to l5
  - want to keep going back and forth between l3 and l5



Planning for Extended Goals

- *Context*: the internal state of the controller
- *Plan*: $(C, c_0, act, ctxt)$
  - C = a set of execution contexts
  - c_0 is the initial context
  - $act: S \times C \rightarrow A$
  - $ctxt: S \times C \times S \rightarrow C$

- Sections 17.3 extends the ideas in Sections 17.1 and 17.2 to deal with extended goals
  - We'll skip the details