# Fast Strong Planning for FOND Problems with Multi-Root Directed Acyclic Graphs

Jicheng Fu[1], Andres Calderon Jaramillo[1],
Vincent Ng[2], Farokh Bastani[2], and I-Ling Yen[2]

[1]The University of Central Oklahoma
[2]The University of Texas at Dallas

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

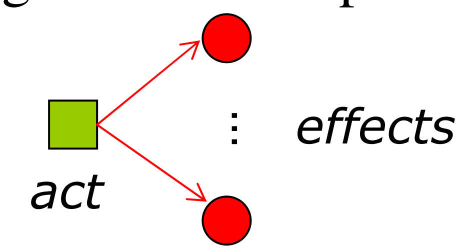
- A planning problem is a triple $\langle s_0, g, \Sigma \rangle$, where
  - $s_0$ is the initial state,
  - $g$ is the goal condition, and
  - $\Sigma$ is the planning domain

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

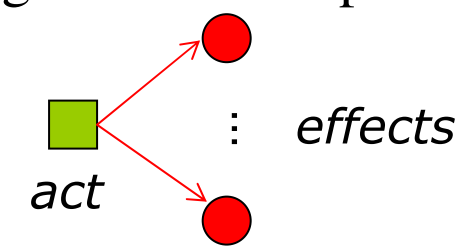
# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
- Informally, in a nondeterministic planning domain,
  - an action may generate multiple effects



# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain
- Informally, in a nondeterministic planning domain,
  - an action may generate multiple effects




- Formally, a nondeterministic domain
  - is a 4-tuple $\Sigma = (P, S, A, \gamma)$
    - $P$ is a finite set of propositions;
    - $S \subseteq 2^P$ is a finite set of states in the system;
    - $A$ is a finite set of actions; and
    - $\gamma: S \times A \to 2^S$ is the state-transition function

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

- Full observability
  - The states of the world are fully observable

# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

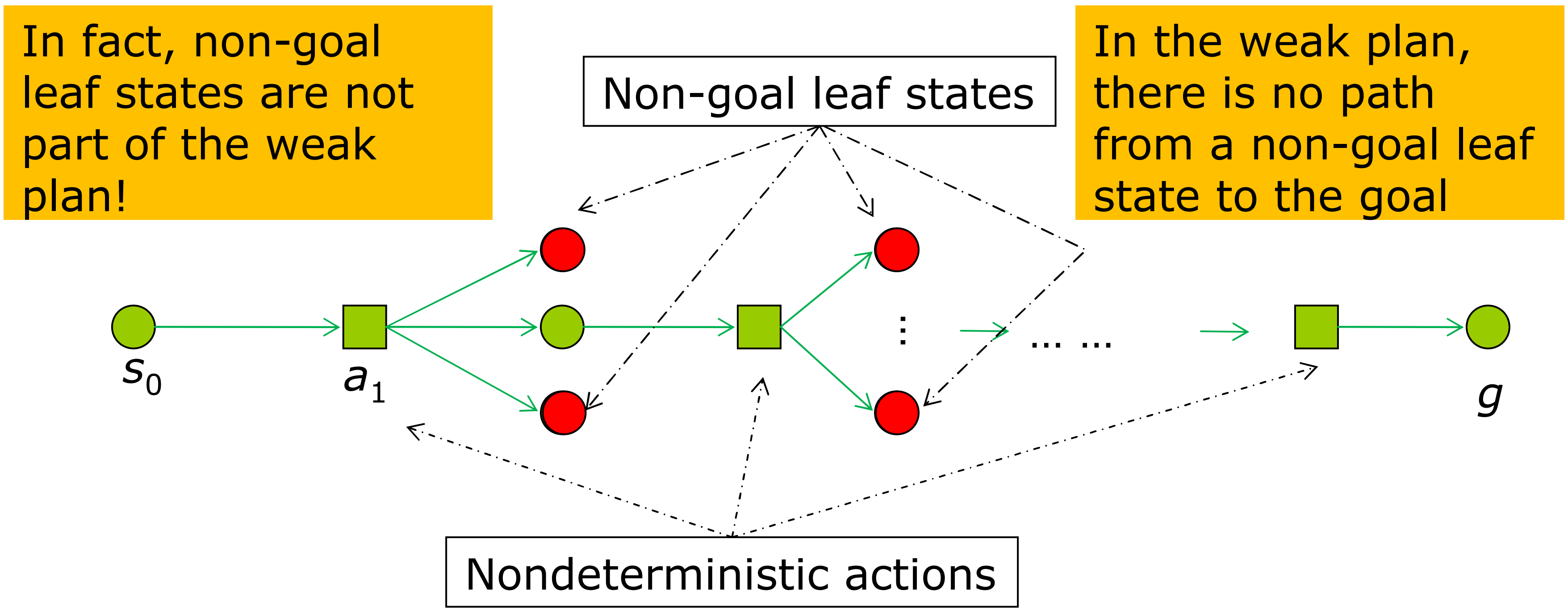
# Goal

- To solve strong planning problems from a Fully-Observable Nondeterministic planning domain

- Strong planning
  - refers to a particular type of solutions to nondeterministic problems
  - different from so-called weak planning and strong cyclic planning

# Weak Planning Solutions

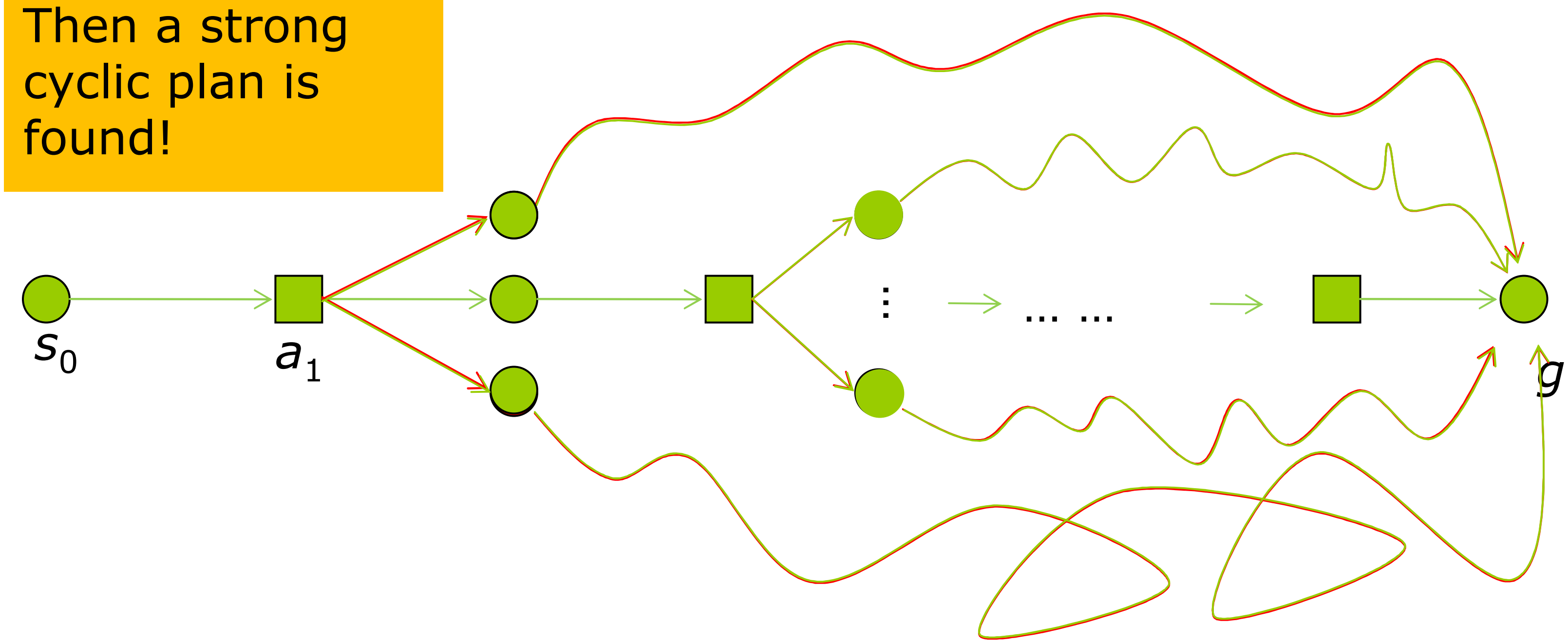
- Solutions where there is a chance to achieve the goal



# Strong Cyclic Planning Solutions

- prescribe actions for all possible non-goal leaf states
  - find a path for each non-goal leaf state to the goal state
  - May loop indefinitely
  - But contain no dead-ends
  - More difficult than finding weak planning solutions

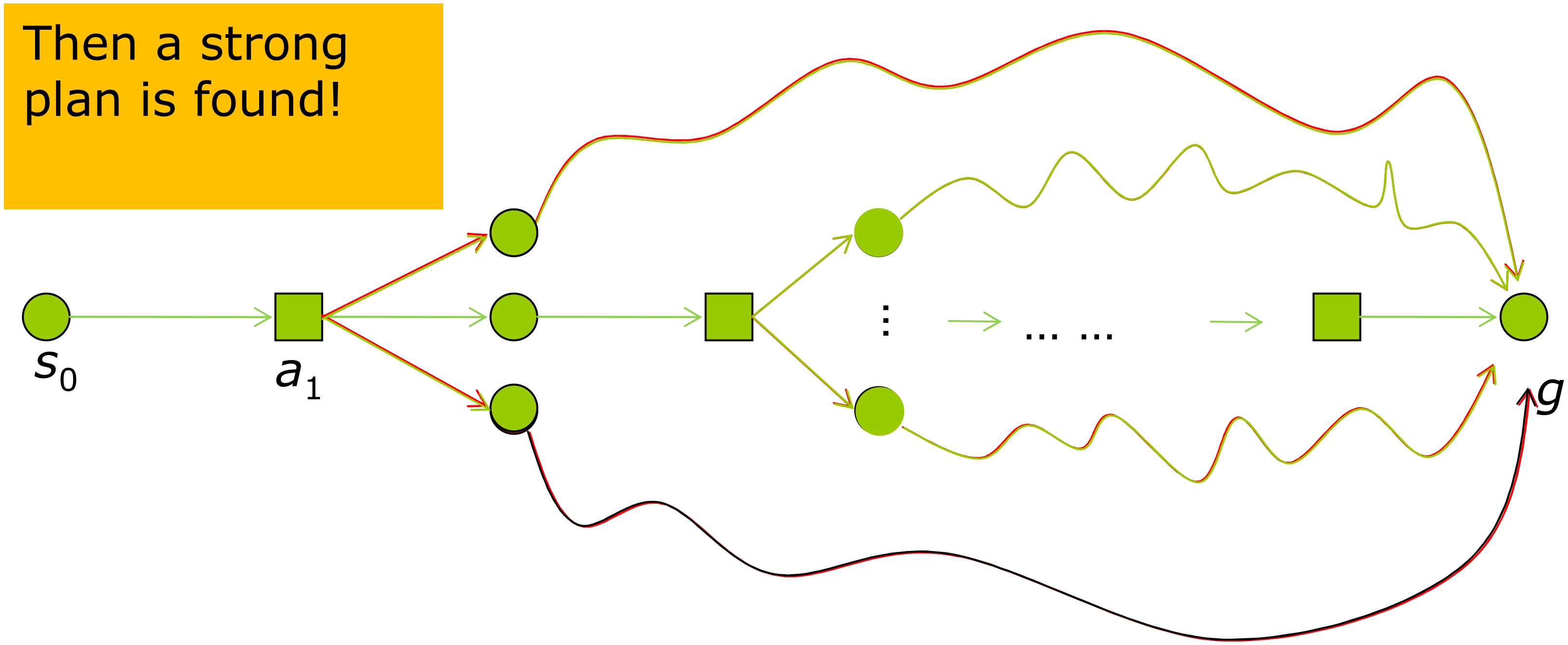
Then a strong cyclic plan is found!



# Strong Planning Solutions

- prescribe actions for all possible non-goal leaf states
  - find a path for each non-goal leaf state to the goal state
  - Contain no cycles
  - Contain no dead-ends

Then a strong plan is found!



# Representing a Plan

- Regardless of whether a plan is weak, strong cyclic, or strong, we can represent it as a policy $\pi$
  - a partial function mapping states to actions

- More formally, policy $\pi : S_\pi \rightarrow A$
  - consists of state action pairs ($s$, $a$) such that $\pi(s) = a$
  - defines which action to take under state $s$

# How to Generate a Strong Plan

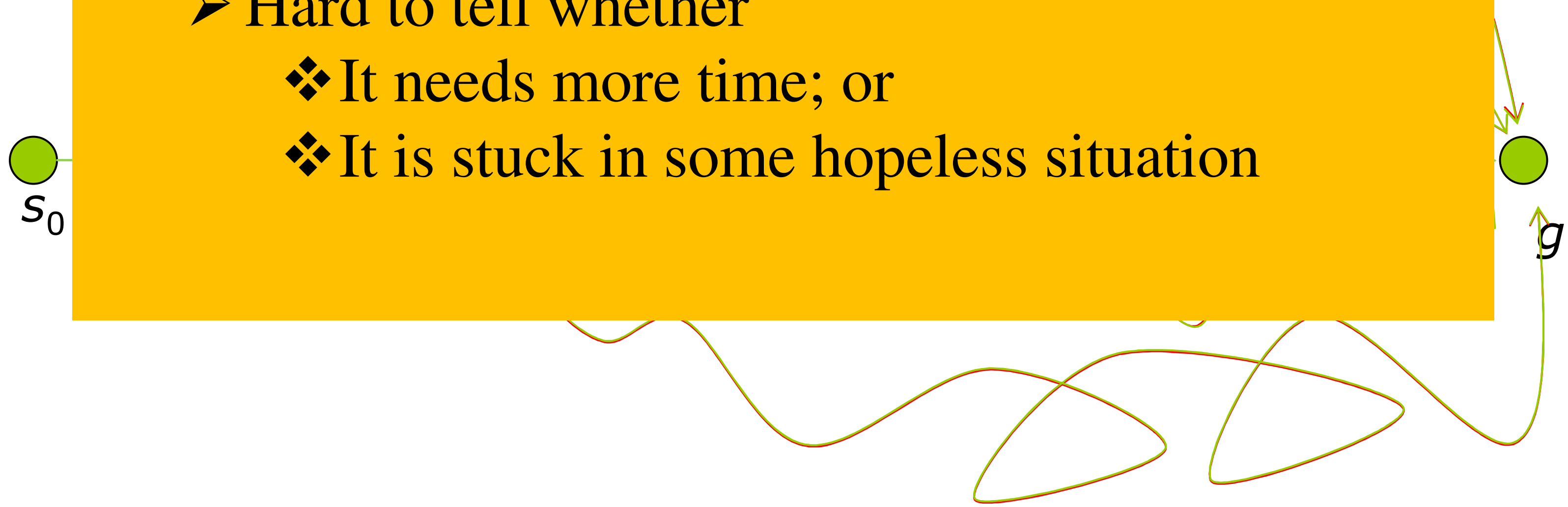
- Choice 1:
  - Upgrade a state-of-the-art strong cyclic planner
    - Such as our FIP [Fu et al., 2011] or PRP [Muise et al., 2012]
    - 3 orders of magnitude faster than other state-of-the-art planners, such as Gamer and MBP

# How to Generate a Strong Plan

- State-of-the-art strong cyclic planner tries to
  - find a path for each non-goal leaf state to the goal state
    - Using a classical planner

**Issue:**

- Lack of control over planning efficiency
  - If the classical planner runs longer than expected
  - Hard to tell whether
    - It needs more time; or
    - It is stuck in some hopeless situation

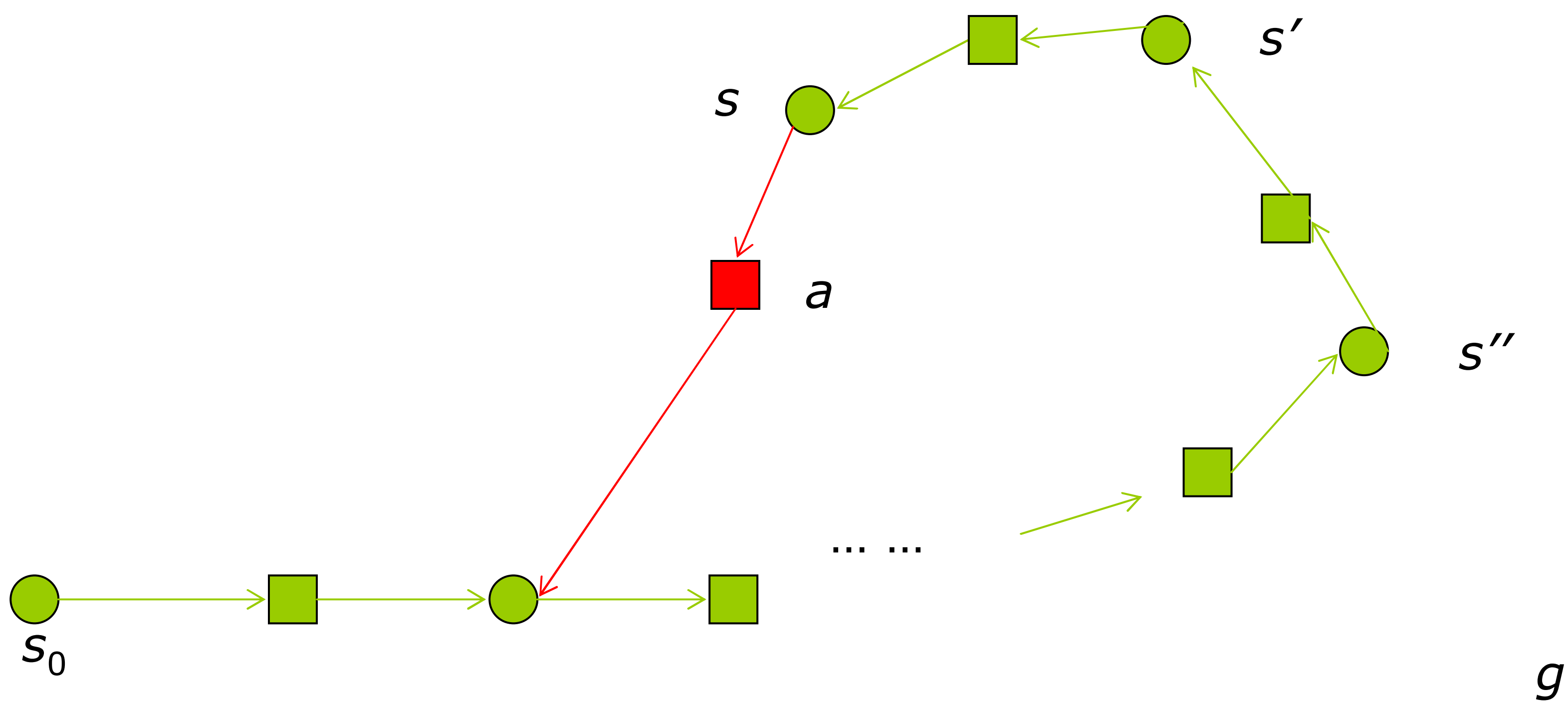


# Desirable Characteristics

- Has full control over planning
- Has heuristics to ensure planning towards the relevant search direction

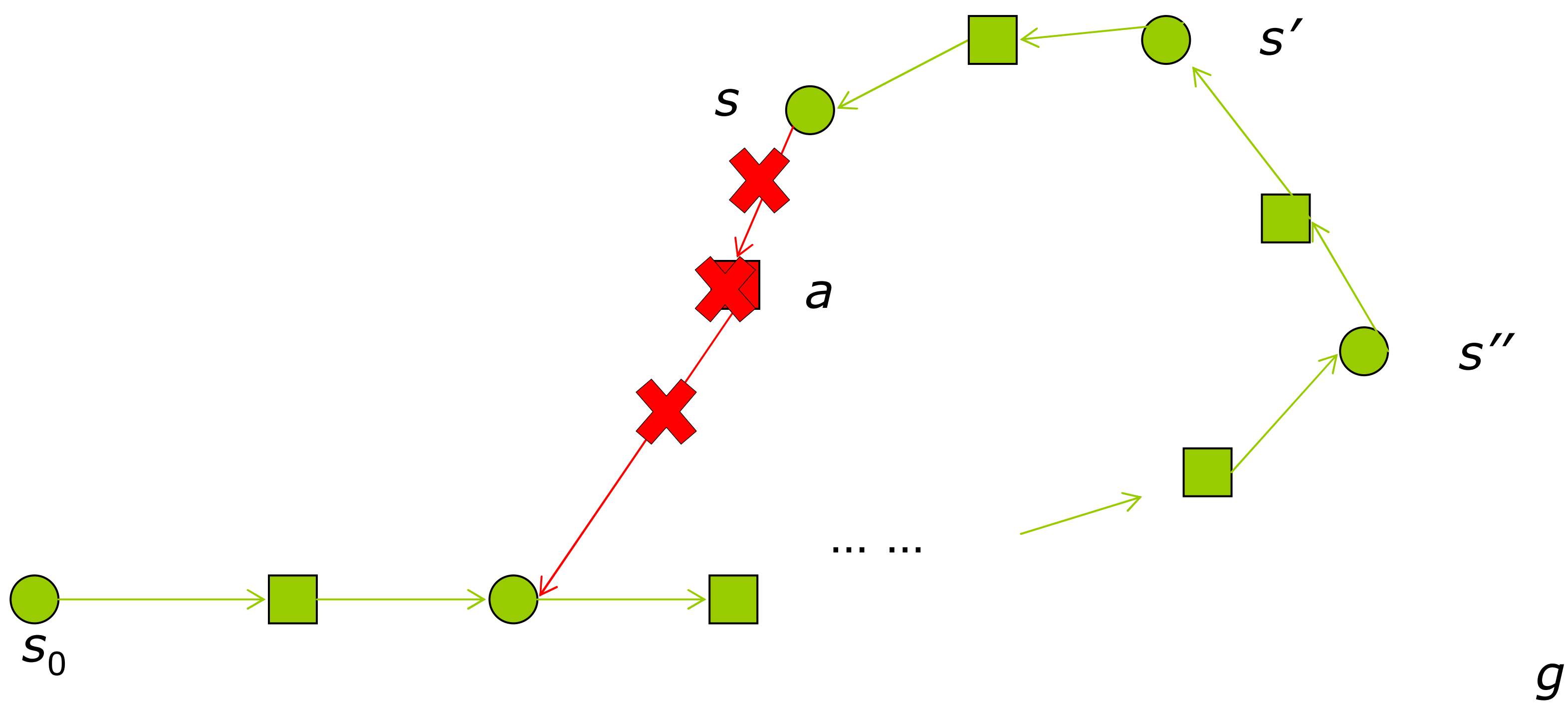
# An Observation

- Applying action *a* to state *s* leads to a cycle
  - Backtrack: make action *a* inapplicable to *s*



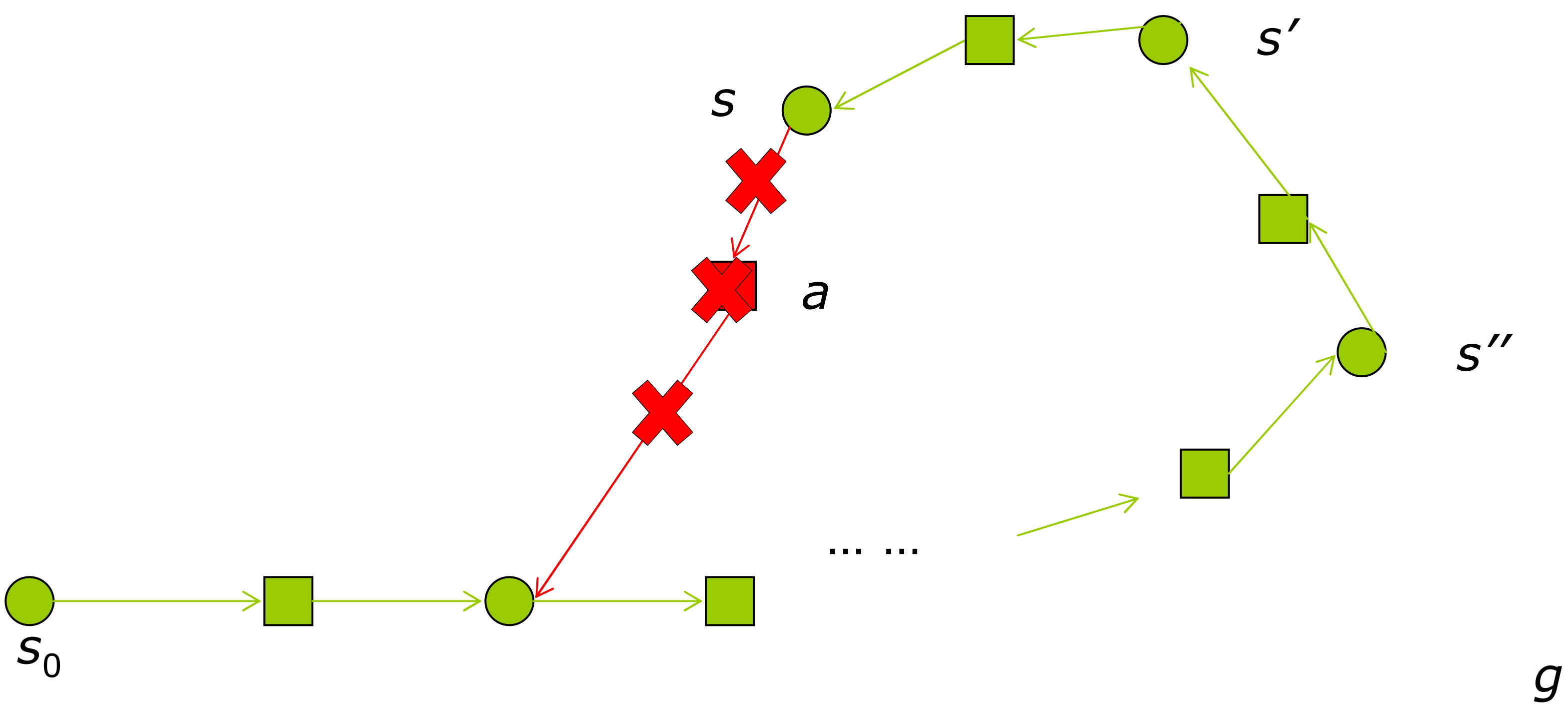
# An Observation

- Applying action *a* to state *s* leads to a cycle
  - Backtrack: make action *a* inapplicable to *s*



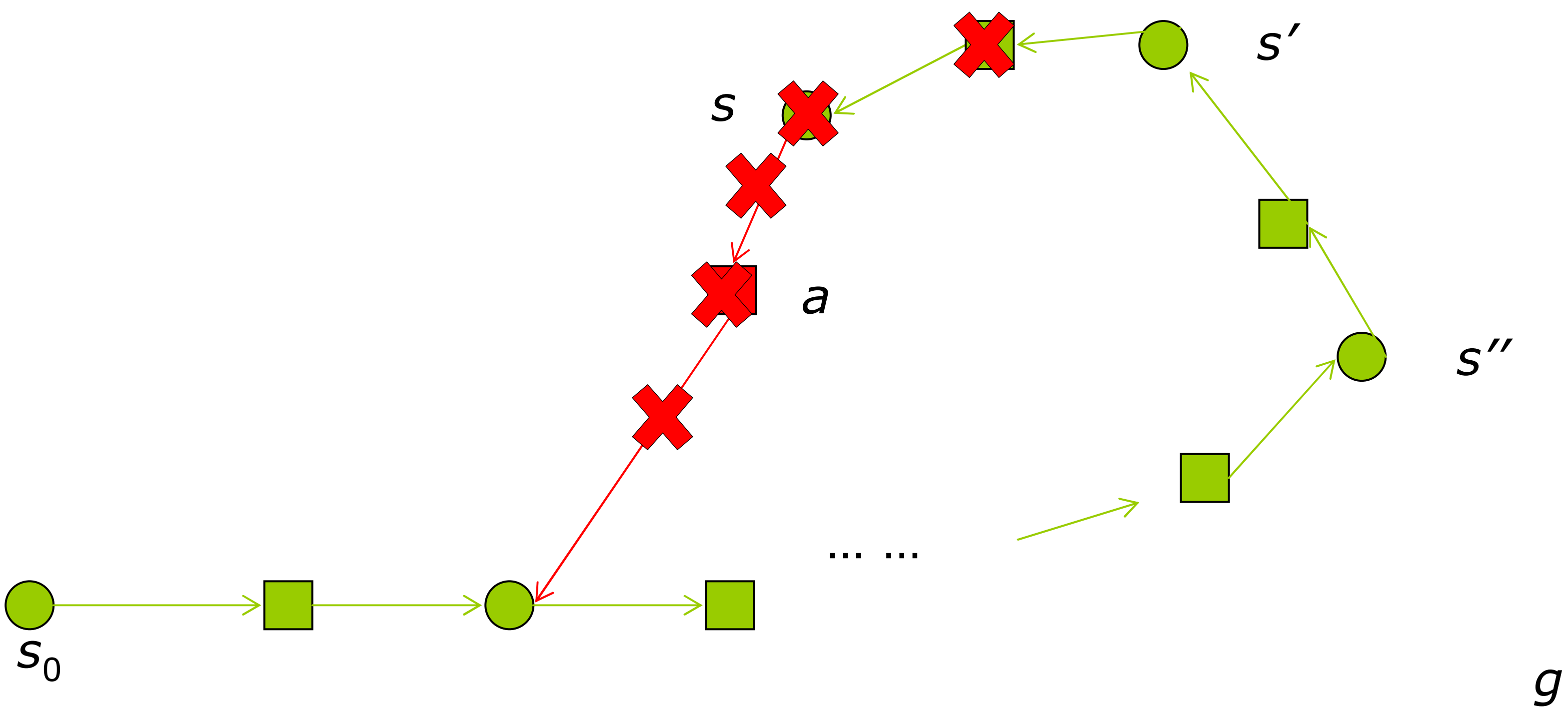
# An Observation

- Applying action *a* to state *s* leads to a cycle
  - Backtrack: make action *a* inapplicable to *s*
  - If state *s* only has one applicable action
    - It becomes a dead-end now
    - Backtrack continues to *s′*



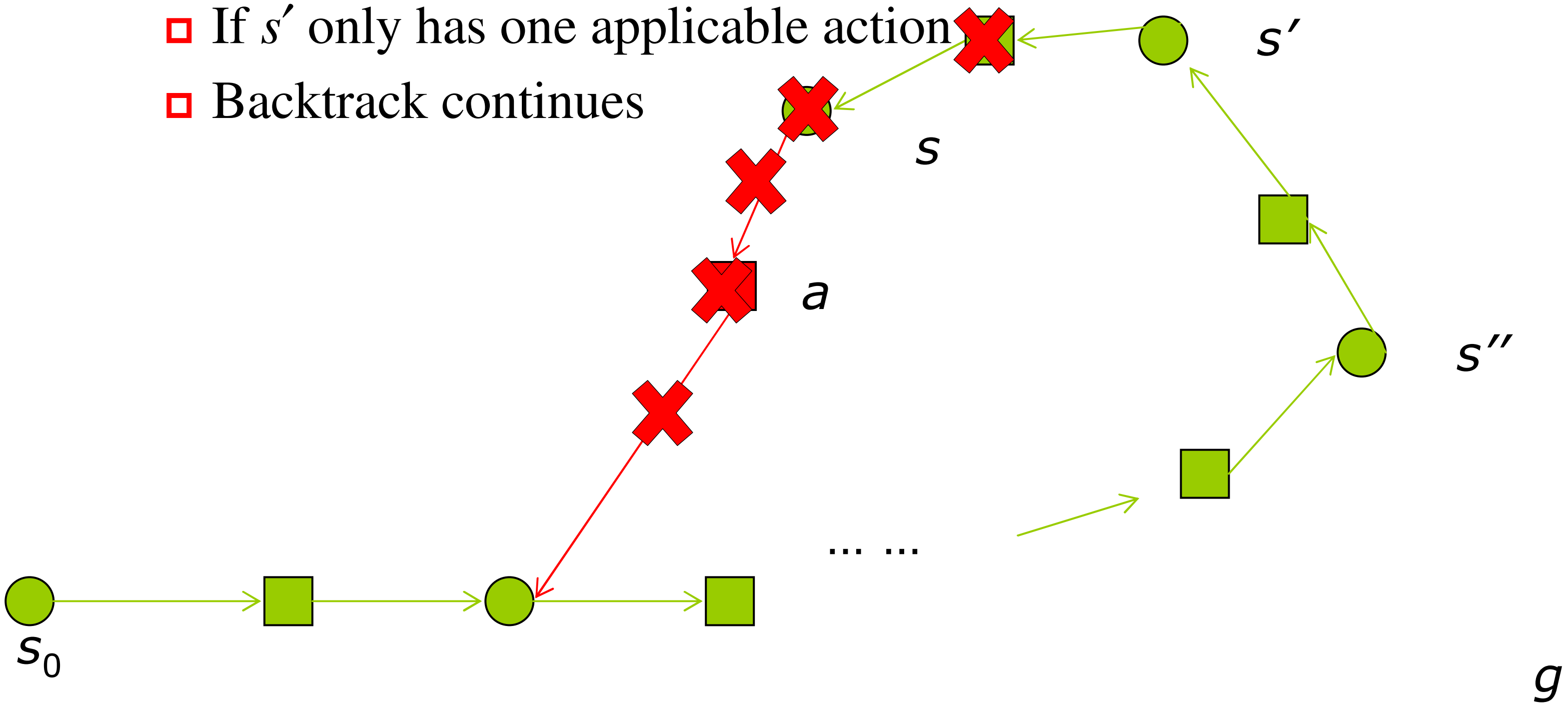
# An Observation

- Applying action *a* to state *s* leads to a cycle
  - Backtrack: make action *a* inapplicable to *s*
  - If state *s* only has one applicable action
    - It is a dead-end now
    - Backtrack continues to *s′*



# An Observation

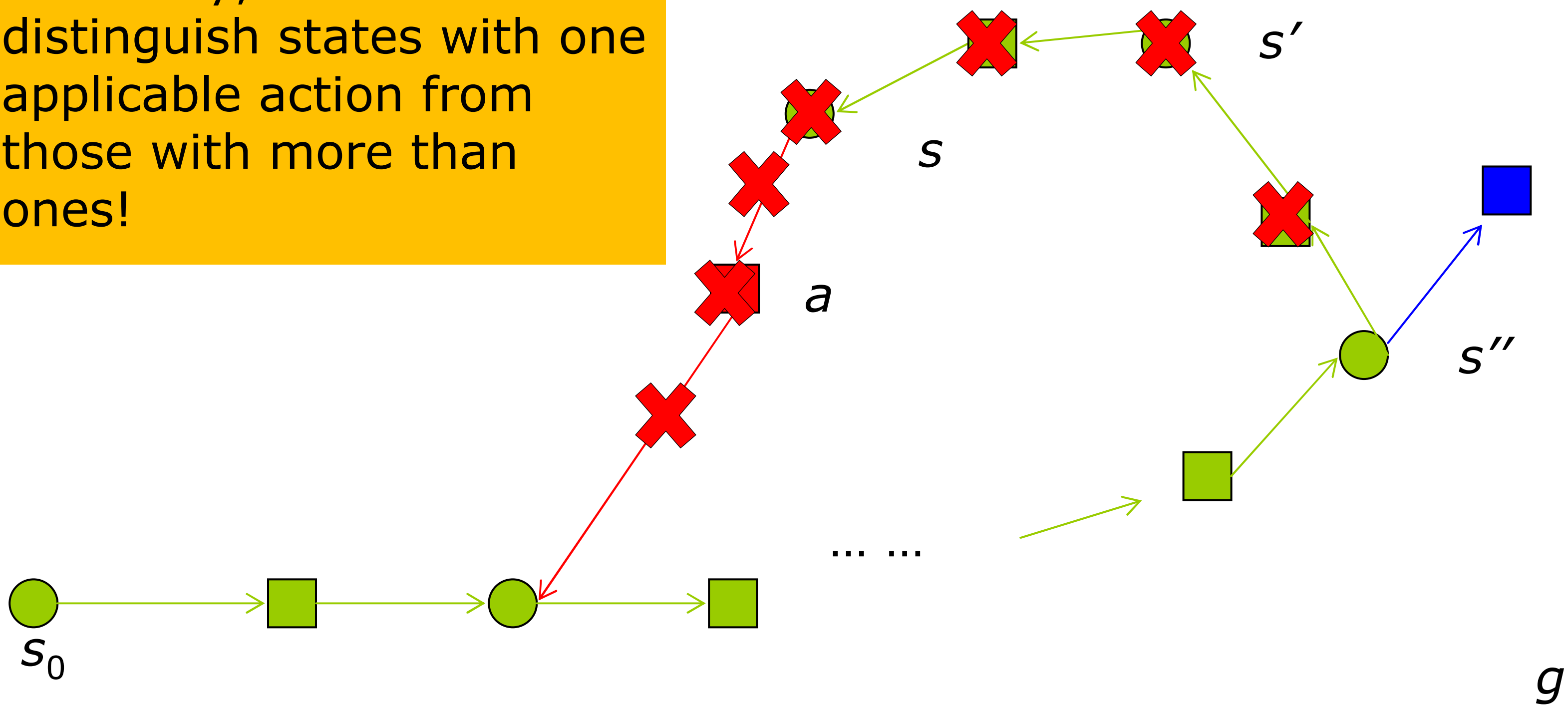
- Applying action *a* to state *s* leads to a cycle
  - Backtrack: make action *a* inapplicable to *s*
  - If state *s* only has one applicable action
    - It is a dead-end now
    - Backtrack continues to *s′*
    - If *s′* only has one applicable action
    - Backtrack continues



# An Observation

- Applying action *a* to state *s* leads to a cycle
  - Backtrack continues until
    - It reaches a state *s″* that has more than one applicable action

To handle cycles efficiently, we should distinguish states with one applicable action from those with more than ones!

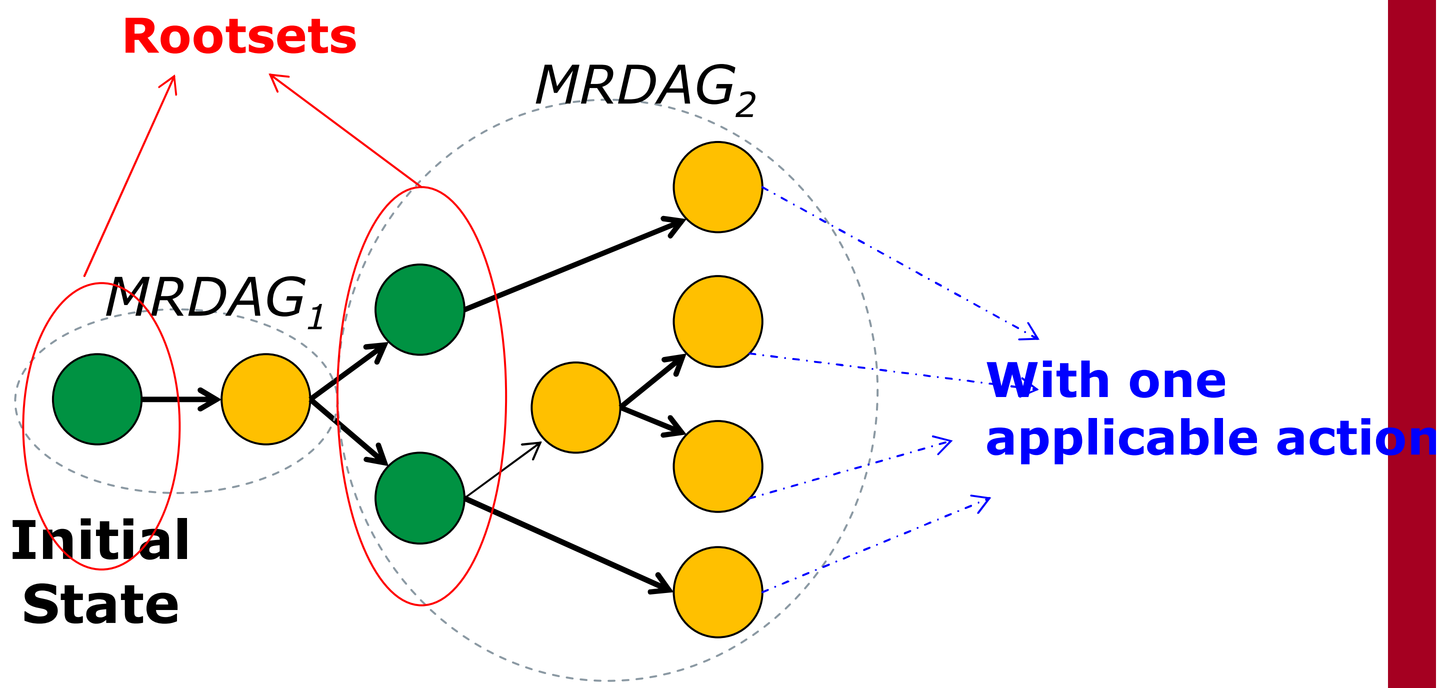


# States with One Applicable Action

- Very common
  - 25% of the states have only one applicable action
    - Based on benchmark problems in the International Planning Competition 2008 (IPC 2008)
  - More states will become those with only one applicable action as planning goes on
    - Actions are made inapplicable if they lead to cycles or dead-ends

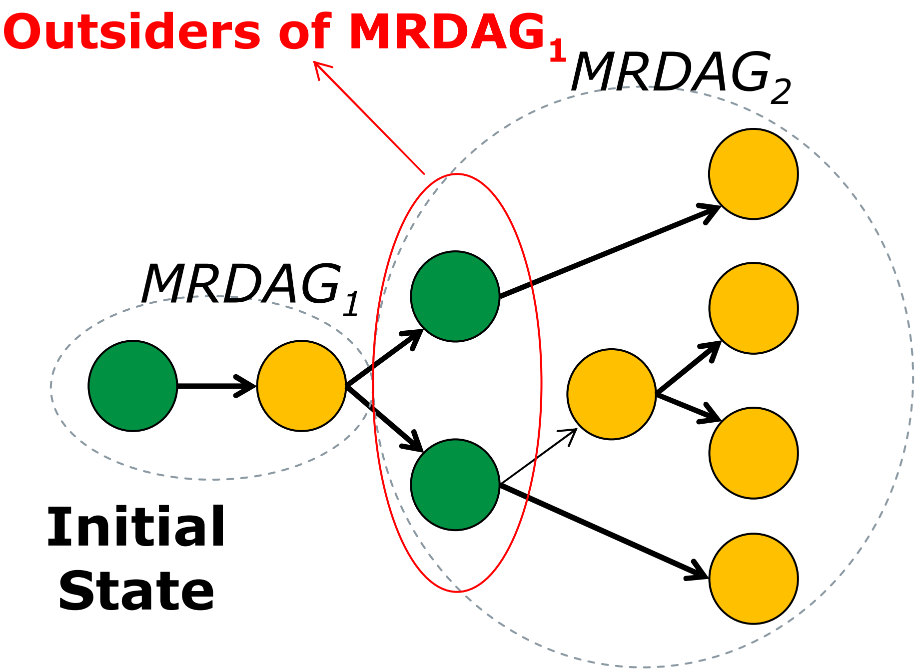
# MRDAG: Multi-Root Directed Acyclic Graph

- A MRDAG $M = \{S_{Mr}, \pi_M\}$ consists of two elements, namely, a rootset $S_{Mr}$ and a policy $\pi_M$.
  - $S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\} \subseteq S_{\pi M}$ consists of a set of states
  - States not in $S_{Mr}$ have only one applicable action



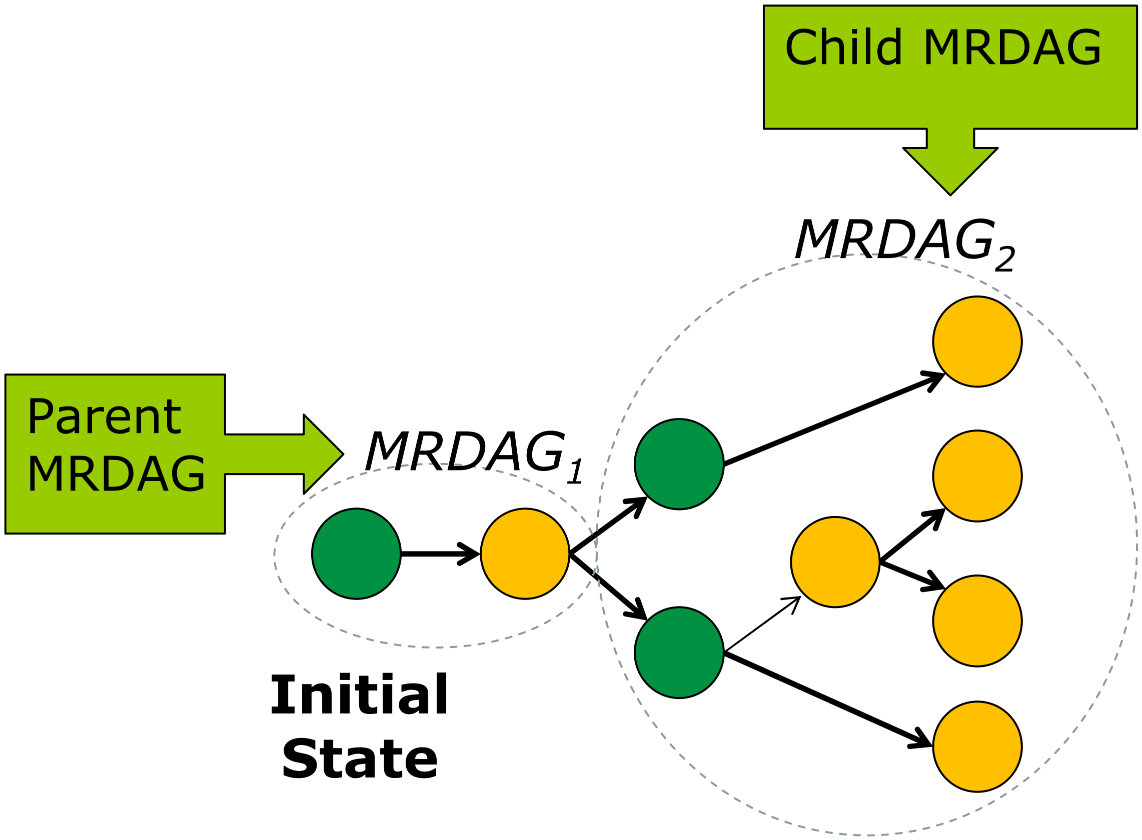
# Outsider of MRDAG

- A state $s$ is called an <u>outsider</u> of a MRDAG $M = \{S_{Mr}, \pi_M\}$ if one of the following two conditions is satisfied:
  - $s$ is a goal; or
  - there exists $(s', a') \in \pi_M$ such that $s \in \gamma(s', a')$; in addition, $|A(s)| > 1$ and $s$ does not belong to any of $M$'s *ancestry* MRDAGs (i.e., MRDAGs constructed prior to $M$)



# Child MRDAG

- A MRDAG $M_c$ rooted at $S_{Mcr}$ is a <u>child</u> of MRDAG $M_p$ if $S_{Mcr}$ is the set of all non-goal outsiders of $M_p$. $M_p$ is called the <u>parent</u> of $M_c$.



# A Feasible MRDAG

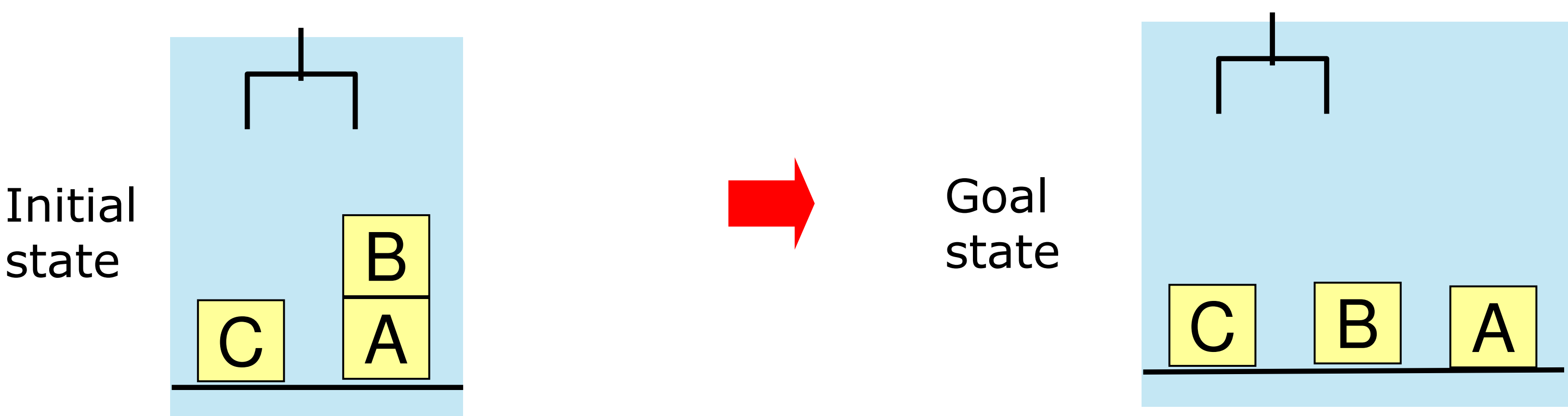
- A MRDAG $M = \{S_{Mr}, \pi_M\}$ is <u>feasible</u> if the following three conditions are satisfied:
  - $\forall(s, a) \in \pi_M$, applying $a$ to $s$ does not lead to a cycle in $G_\pi(s_0)$;
  - $\forall(s, a) \in \pi_M$, applying $a$ to $s$ does not lead to a dead-end; and
  - the child of $M$, if any, is also feasible

# A Strong Solution

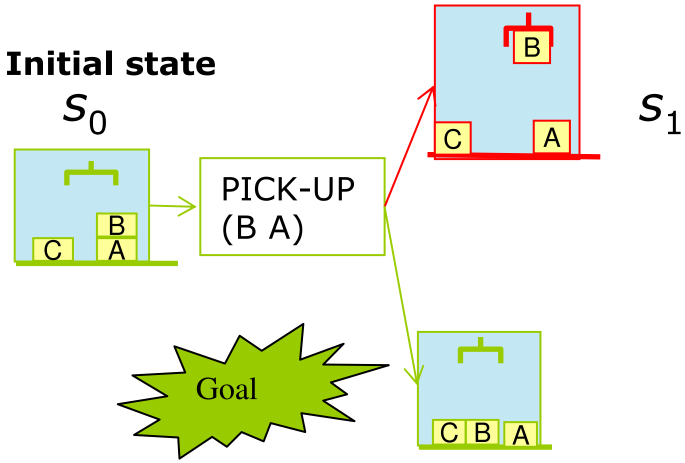
- A <u>strong solution</u> is $\pi = \pi_{M1} \cup \pi_{M2} \cup \ldots \cup \pi_{Mn}$, where $\pi_{M1}, \pi_{M2}, \ldots, \pi_{Mn}$ are the policies of a sequence of MRDAGs $M_1, M_2, \ldots, M_n$, if the following three conditions are satisfied:
  - $M_1$ is rooted at $s_0$, i.e., the initial state;
  - $M_i$ is the parent of $M_{i+1}$ for $i = 1, 2, 3, \ldots, n - 1$; and
  - all the outsiders of $M_n$ are goal states

# Example: Simplified Blocksworld Domain

- Deterministic action *put-down*(B)
  - puts block B onto the table
- Two nondeterministic actions
  - *pick-up*(A, B)
  - *put-on*(A, B)
  - Both actions may drop the held block A onto the table.

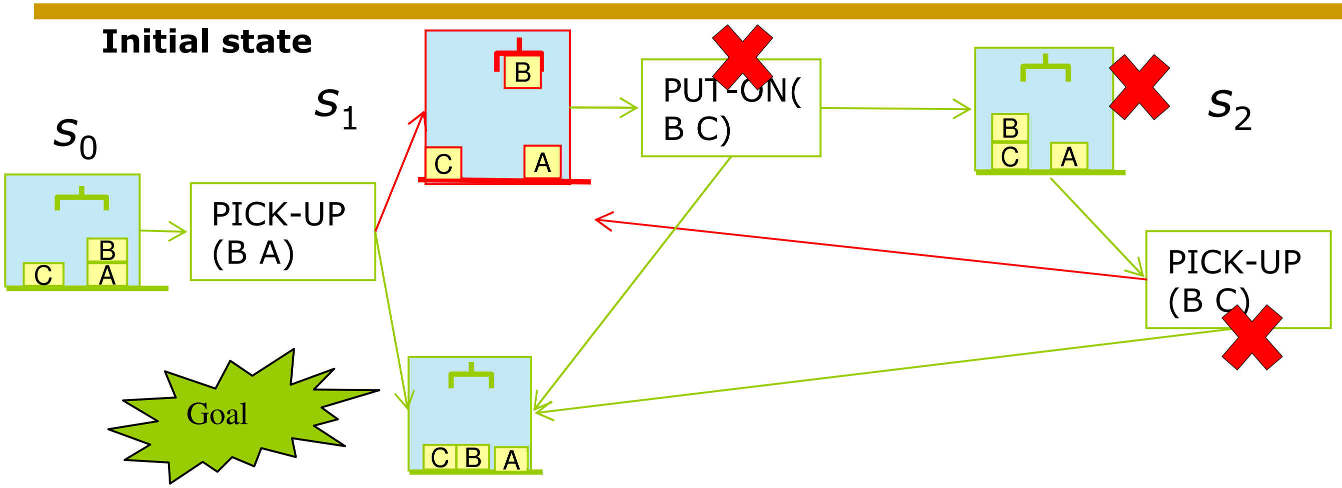


# Blocksworld Example – The First Weak Plan




$$\text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\} \rangle$$

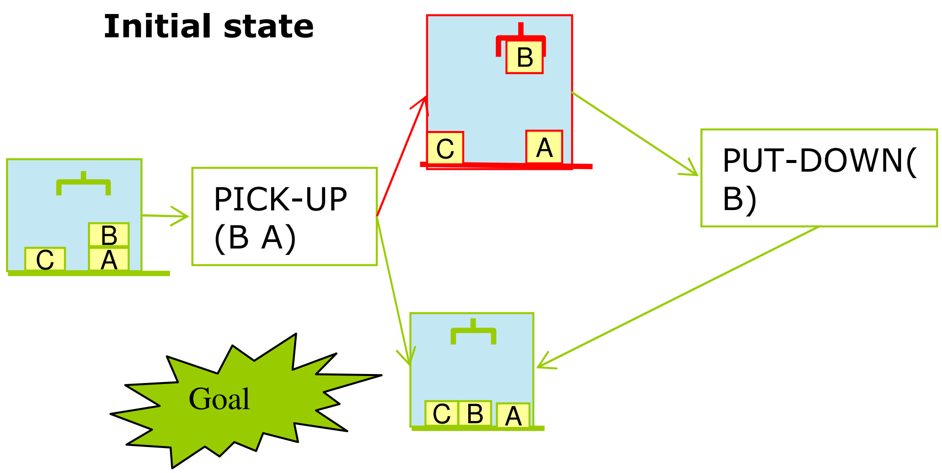
# Blocksworld Example – The First Weak Plan




$\text{MRDAG}_1 = \langle\{s_0\}, \{(s_0, \text{PICK-UP(B A)})\}\rangle$

$\text{MRDAG}_2 = \langle\{s_1\}, \{(s_1, \text{PUT-ON(B C)})\}, (s_2, \text{PICK-UP(B C)}\}\rangle$

# Blocksworld Example – The First Weak Plan



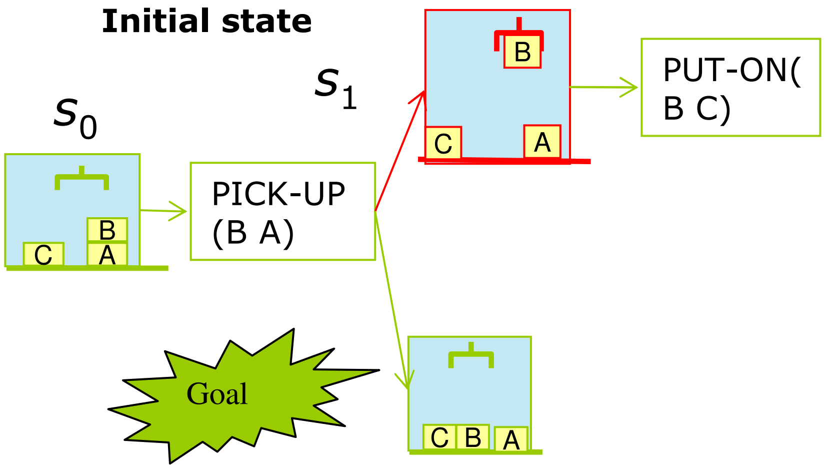

$$\text{MRDAG}_1 = \langle\{s_0\}, \{(s_0, \text{PICK-UP(B A)})\}\rangle$$

$$\text{MRDAG}_2 = \langle\{s_1\}, \{(s_1, \text{PUT-DOWN(B)})\}\rangle$$

# Outline of the Strong Planning Algorithm

```
Global Variables: π, ⟨s0, g, Σ⟩
Function STRONG_PLANNING
R ← {s0}; π ← φ          /*R is the rootset of the MRDAG*/
while R ≠ φ do
    πM ← GET-NEXT-SET-OF-ACTIONS(R)
    if πM = φ then
        if R = {s0} then return FAILURE else
            BACKTRACK(R)
        endif
    else
        if BUILD-MRDAG(πM) <> FAILURE then
            π ← π ∪ πM
            if All-GOAL-OUTSIDERS(R, πM) then
                    return π
            else
                    R ← GET-OUTSIDERS(R, πM)
            endif
        endif
    endif
endwhile
```

# Blocksworld Example – The First Weak Plan



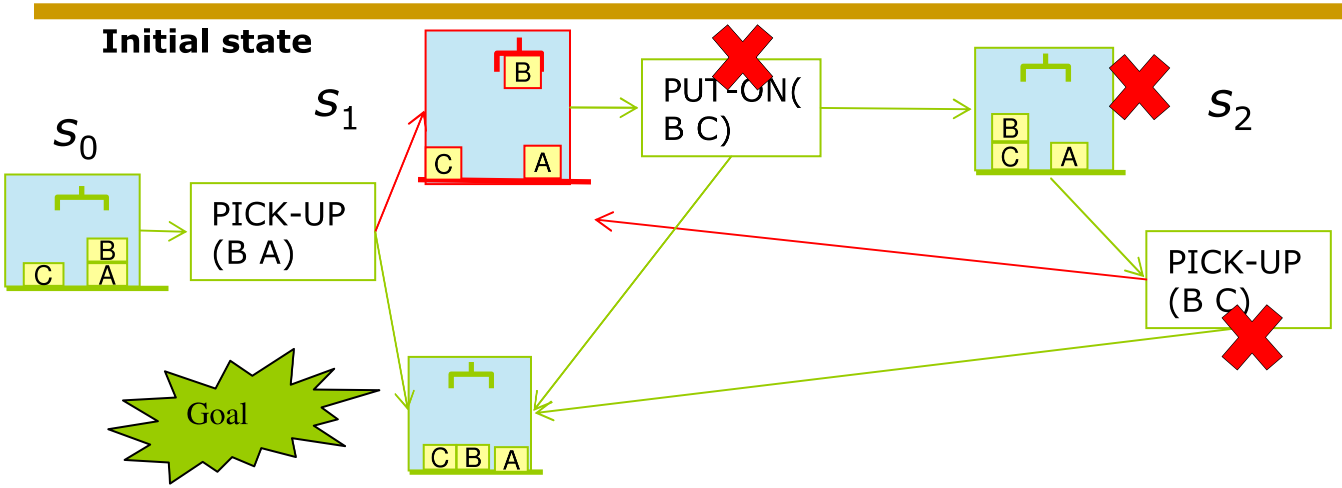

$$\text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\} \rangle$$

$$\text{MRDAG}_2 = \langle \{s_1\}, \{(s_1, \text{PUT-ON(B C)})\} \rangle$$

# Building a Feasible MRDAG

**Function** EXPAND-MRDAG ($\pi_M$, $s$, $a$)
**foreach** $s' \in \gamma(s, a)$ & $NOT\text{-}GOAL(s')$ **do**
  **if** $s' \in S_\pi$ or $s' \in S_{\pi M}$ **then**
    **if** $DETECT\text{-}CYCLE(\pi \cup \pi_M)$ = TRUE **then**
      **return** FAILURE
    **endif**
  **elseif** $|A(s')| = 1$ **then**
    $\pi_M \leftarrow \pi_M \cup \{(s', a')\}$ with $a' \in A(s')$
    **if** $EXPAND\text{-}MRDAG$ ($\pi M$, $s'$, $a'$) = FAILURE **then**
      **return** FAILURE
    **endif**
  **elseif** $|A(s')| = 0$ **then** /*dead-end*/
      **return** FAILURE
  **endif**
**endfor**
**return** SUCCESS

# Blocksworld Example – The First Weak Plan




$\text{MRDAG}_1 = \langle \{s_0\}, \{(s_0, \text{PICK-UP(B A)})\} \rangle$

$\text{MRDAG}_2 = \langle \{s_1\}, \{(s_1, \text{PUT-ON(B C)})\}, \{(s_2, \text{PICK-UP(B C)}\} \rangle$

# Two Heuristics

- Try to answer
  - How to impose an ordering on the states to be expanded in the same rootset?
  - How to impose an ordering on the actions to be chosen for a state in the rootset?

# Most Constrained State (MCS) Heuristic

- Assume that the rootset of a MRDAG is $S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\}$.
- Sort the states in $S_{Mr}$ in increasing order of the number of actions applicable to a state.

| $s_{r1}$ | $s_{r2}$ | … | $s_{rk}$ |
|---|---|---|---|
| $a_{11}$ | $a_{21}$ | … | $a_{k1}$ |
| $a_{12}$ | $a_{21}$ | … | $a_{k1}$ |
| $a_{13}$ | $a_{21}$ | … | $a_{k1}$ |
| ⋮ | | | |
| $a_{1<m1>}$ | $a_{21}$ | … | $a_{k1}$ |
| $a_{11}$ | $a_{22}$ | … | $a_{k1}$ |
| $a_{12}$ | $a_{22}$ | … | $a_{k1}$ |
| $a_{13}$ | $a_{22}$ | … | $a_{k1}$ |
| ⋮ | | | |
| $a_{1<m1>}$ | $a_{2<m2>}$ | … | $a_{k<mk>}$ |

# Least Heuristic Distance (LHD)

- For each state $s_{ri} \in S_{Mr} = \{s_{r1}, s_{r2}, \ldots, s_{rk}\}$ $(1 \leq i \leq k)$, we sort its applicable actions in increasing order of the heuristic distance to the goal.

# Evaluation

- All problem instances were derived from the benchmark domains of the IPC2008 FOND track
  - Blocksworld, Tireworld, Faults, and First-responders
- Goal
  - For comparison, we implemented four versions
    - SP uses both heuristics,
    - MCS uses only the MCS heuristic,
    - LHD uses only the LHD heuristic, and
    - NOH uses none of the heuristics.
    - Two state-of-the-art strong planners: Gamer and MBP
  - give each planner 1200 seconds to solve each problem instance

# Evaluation 1: Problem Coverage

| Domain | Gamer | MBP | SP | LHD | MCS | NOH |
|---|---|---|---|---|---|---|
| ***scbw* (30)** | 10 | 10 | 29 | 30 | 30 | 30 |
| **bw(30)** | 10 | 0 | 30 | 30 | 10 | 10 |
| **ft (10)** | 6 | 4 | 10 | 10 | 3 | 3 |
| **tw (12)** | 11 | 0 | 12 | 12 | 5 | 4 |
| **fr (50)** | 20 | 10 | 49 | 49 | 46 | 45 |
| **Total (132)** | 57 | 24 | 130 | 131 | 94 | 92 |

Our planners solve more problems than Gamer and MBP within the time limit

# Evaluation 2: Efficiency

| | Gamer | | MBP | SP | | LHD | | MCS | | NOH | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| bw-6 | 87.177 | 14 | --- | 0.002 | 14 | 0.002 | 14 | 0.001 | 14 | 0.001 | 21 |
| bw-7 | 87.738 | 28 | --- | 0.004 | 28 | 0.005 | 28 | 0.002 | 47 | 0.001 | 52 |
| fr-10-2 | --- | --- | --- | 0.013 | 12 | 0.012 | 11 | 0.081 | 505 | 0.030 | 197 |

- Comparing with Gamer and MBP
  - SP and LHD are about 4 orders of magnitude faster on strong blocksworld, first-responders, and tiresworld,
  - about 3 orders of magnitude faster than Gamer on faults, and
  - 2 orders of magnitude faster on strong cyclic blocksworld.

- In terms of the contributions made by the two heuristics
  - LHD is on average 5 times faster on first-responders, and up to 2 orders of magnitude faster on tireworld and 3 orders of magnitude faster on faults than MCS.
  - MCS is about 3 times faster than LHD on strong and strong cyclic blocksworld domains.
  - In terms of plan size, LHD consistently generates much compacter plans than MCS.

[1] MBP often outputs too much information to count policy size.

# Summary

- Proposed a novel data structure, MRDAG (multi-root directed acyclic graph)
- Conducted extensive experiments to evaluate how planning performance is affected by
  - the order in which the actions applicable to a state are chosen and
  - the order in which the states in the rootset of a MRDAG are expanded via the proposal of two heuristics, MCS and LHD.

# Summary

- Experimental results showed that
  - the use of MRDAG indeed made cycle handling easier and more efficient, and
  - the use of the LHD heuristic significantly improved planning performance.
  - our planner significantly outperformed two state-of-the-art planners, Gamer and MBP, by solving more problems in less time.

# Reference


- [1] U. Kuter, D. Nau, E. Reisner, and R. P. Goldman, "Using classical planners to solve nondeterministic planning problems," *in 18th International Conference on Automated Planning and Scheduling (ICAPS)*, 2008.
- [2] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "Weak, strong, and strong cyclic planning via symbolic model checking," *Artif. Intell.*, vol. 147, pp. 35-84, 2003.
- [3] P. Kissmann and S. Edelkamp, "Solving Fully-Observable Non-deterministic Planning Problems via Translation into a General Game," *in KI 2009: Advances in Artificial Intelligence*. vol. 5803, B. Mertsching, et al., Eds., ed: Springer Berlin Heidelberg, 2009, pp. 1-8.
- [4] J. Fu, V. Ng, F. B. Bastani, and I.-L. Yen, "Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems," *in Proceedings of the Twenty-Second international joint conference on Artificial Intelligence* - Volume Three, Barcelona, Catalonia, Spain, 2011.
- [5] C. J. Muise, S. A. McIlraith, and J. C. Beck, "Improved Non-Deterministic Planning by Exploiting State Relevance," *in ICAPS*, 2012.
- [6] D. Bryce and O. Buffet, "International Planning Competition Uncertainty Part: Benchmarks and Results," *in Proceedings of International Planning Competition*, 2008.
- [7] D. Bryce and O. Buffet, "6th International Planning Competition: Uncertainty Part," *in Proceedings of International Planning Competition*, 2008.
- [8] R. Tarjan, "Depth-first search and linear graph algorithms," *in 12th Annual Symposium on Switching and Automata Theory*, 1971, pp. 114-121.
- [9] J. Hoffmann and B. Nebel, "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253-302, 2001.