Overview

Fact - Every planning algorithm uses some heuristic search strategy.

- Domain-dependent heuristics relate only to a specific domain.
- Domain-independent heuristics can be used to guide search in any domain.

Possible Claim - There is a single best domain-independent heuristic.

What we Found -

- **There is no single domain-independent search heuristic that performs more efficiently than others for all problems or in all domains.**
- Therefore planners must be flexible enough to use different search methods.

Motivation

- Barrett and Weld (1993)
  - Created several artificial domains.
  - Tested them on three planners: TOPI, TOCL, and POCL (SNLP).
- Their total-order planners performed poorly in these domains.
- PRODIGY uses a **total order**.
- PRODIGY can use **different domain-independent heuristics**.
- FLECS builds on PRODIGY to explicitly switch among these heuristics

PRODIGY's Algorithm

1. Terminate if the goal statement is satisfied in the current state.
2. Compute the set of *pending goals* $\mathcal{G}$, and the set of *applicable operators* $\mathcal{A}$.

   A goal is pending if it is a precondition, not satisfied in the current state, of an operator selected to be in the plan to achieve a particular goal. An operator is applicable when all its preconditions are satisfied in the state.
3. **Choose a goal G from $\mathcal{G}$ or select an operator A from $\mathcal{A}$.**
4. If G has been chosen, then
   1. *Expand goal* G, i.e., get the set $\mathcal{O}$ of *relevant instantiated operators* that could achieve the goal G,
   2. Choose an operator O from $\mathcal{O}$,
   3. Go to step 1.
5. If an operator A has been selected as directly applicable, then
   1. *Apply* A,
   2. Go to step 1.

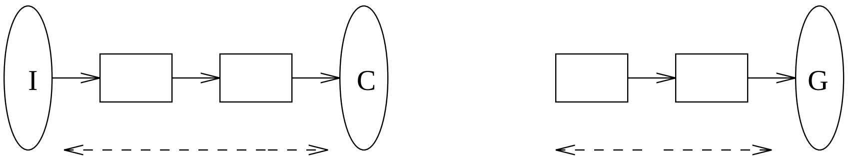
The Importance of Step 3: Apply or Subgoal?

- **Step 3**: PRODIGY's algorithm simplifies to the regular expression (**Subgoal Apply***)*.

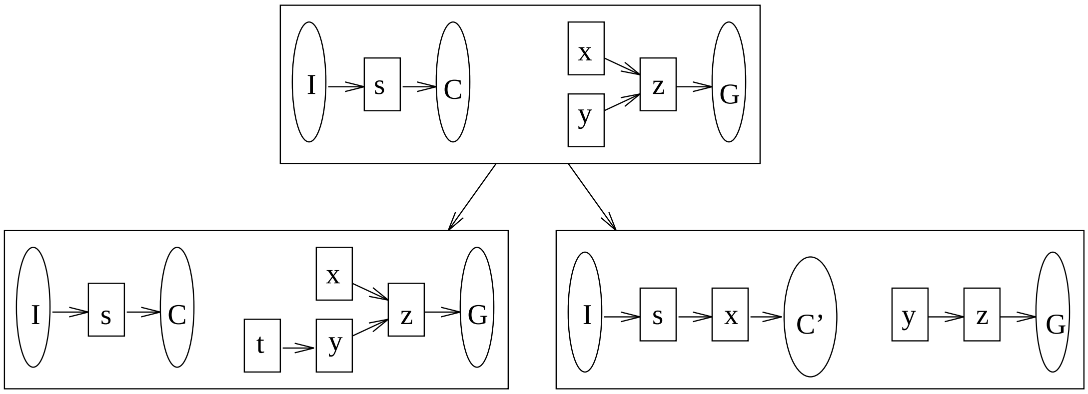
> Two main choices:
> - Plan for a goal OR
> - Change the state, i.e. simulate execution.

- PRODIGY uses **state** to determine ...
  - if the goal state has been reached (**step 1**).
  - which goals still need to be achieved (**step 2**).
  - which operators are applicable (**step 2**).
  - which operators to try first while planning (**step 4.2**).

Subgoaling and Applying



head-plan tail-plan



Subgoaling
(Planning for a goal)

Applying
(Simulating execution)

From Fink and Veloso, CMU-CS-94-123

The Three Heuristics: SAVTA, SABA-reactive, and SABA-memory

SAVTA: Eager application = **Eager** state changes

$\boxed{\textbf{S}\text{ubgoal }\textbf{A}\text{fter e}\textbf{V}\text{ery }\textbf{T}\text{ry to }\textbf{A}\text{pply}}$

SABA: Eager subgoaling = **Delayed** state changes

$\boxed{\textbf{S}\text{ubgoal }\textbf{A}\text{lways }\textbf{B}\text{efore }\textbf{A}\text{pplying}}$

Two SABA heuristics select differently from among applicable operators.

- SABA-reactive: only looks at currently applicable operators.
- SABA-memory: takes past sets of applicable operators into account.

Operator Ordering Choices Using the SABA's

- SABA checks for an operator that satisfies two conditions:
  1. It doesn't delete any preconditions of any other applicable operators;
  2. No other operators delete any of its effects.
- If none satisfy both criteria, preference to condition 1.
- SABA-memory uses information from the analysis of past sets of applicable operators.
- SABA-reactive and SABA-memory are just heuristics.
  - Provide guidance, but no guarantees.
  - Do NOT prune PRODIGY's search space.

Eager State Changes - SAVTA

3. Choose a goal G from $\mathcal{G}$.
   1. *Expand goal* G, i.e., get the set $\mathcal{O}$ of *relevant instantiated operators* for the goal G,
   2. Choose an operator O from $\mathcal{O}$.
   3. Let A=O be the current operator.
4. If A is directly applicable in the current state, then
   1. *Apply* A,
   2. Identify a new possible applicable operator A.
   3. Go to step 4.
5. Go to step 1.

Delayed state changes - SABA

3. If $\mathcal{G}$ is not empty then
   1. Choose a goal G from $\mathcal{G}$ and remove G from $\mathcal{G}$.
   2. *Expand goal* G, i.e., get the set $\mathcal{O}$ of *relevant instantiated operators* for the goal G,
   3. Choose an operator O from $\mathcal{O}$.
   4. Go to step 3.
4. Compute the set of *applicable operators* $\mathcal{A}$.
5. Select an operator A from $\mathcal{A}$.
   1. *Apply* A.
   2. Go to step 1.

1. Terminate if the goal statement is satisfied in the current state.
2. Compute the set of *pending goals* $\mathcal{G}$.

Experiments

The Domains:

- 4 from Barret and Weld (1993): D^0S^1, D^mS^1, D^1S^{1*}, and D^mS^{2*}.
- 1 of our own: D^0-side-effect.

The Problems:

- 150 problems: 10 each with 1 to 15 goals.
- Goals randomly ordered.
- Same problems for each heuristic.

The Hardware: a SPARC station.

The Data:

- Time PRODIGY took to solve vs. number of goals.
- Averaged results from problems with the same number of goals.

D^0S^1: All heuristics work

$$\begin{aligned} \text{Operator:} \quad & A_i \\ \text{preconds:} \quad & \{I_i\} \\ \text{adds:} \quad & \{G_i\} \\ \text{deletes:} \quad & \{\} \end{aligned}$$

- Operators can be executed in **any order**.

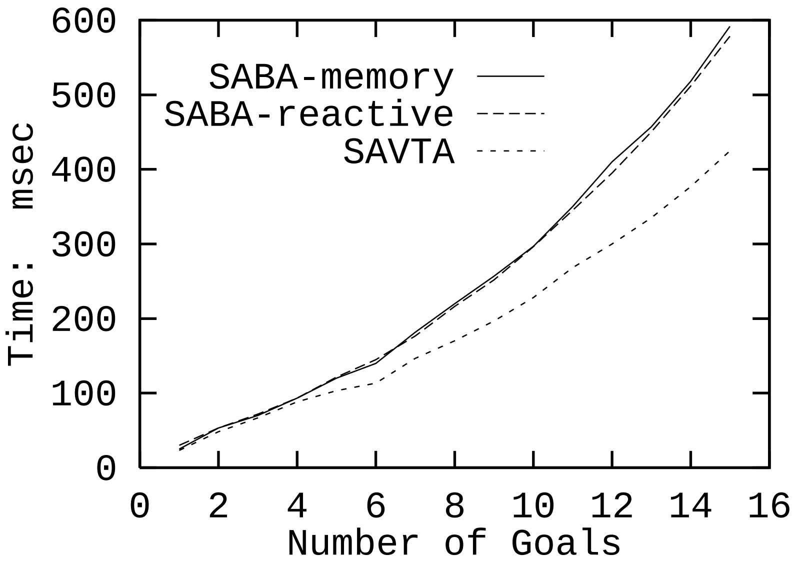
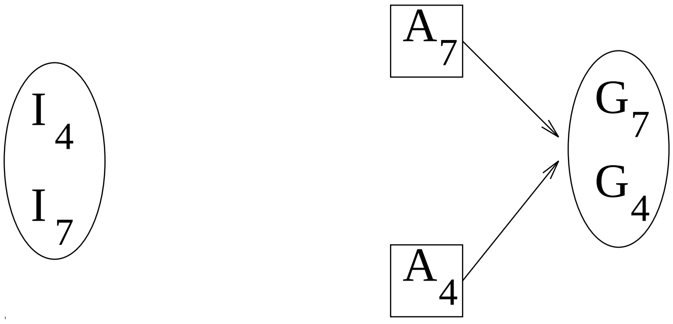


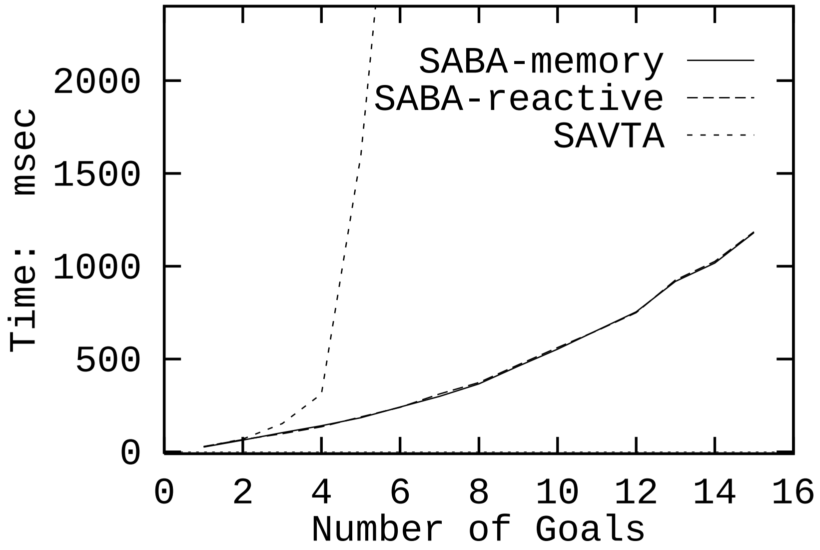
Diagram: D^0S^1 and D^mS^1



D^mS^1: **SAVTA fails**

$$\begin{aligned} \text{Operator:}\quad & A_i \\ \text{preconds:}\quad & \{I_i\} \\ \text{adds:}\quad & \{G_i\} \\ \text{deletes:}\quad & \{I_j \mid j < i\} \end{aligned}$$

- Operators must be executed in a **specific order**.



D^mS^{2*}: SABA-reactive fails

$$\begin{array}{lllll}
\text{Operator:} & A_i^1 & \text{preconds:} & \{I_i\} & \text{adds: } \{P_i\} \\
 & & \text{deletes:} & \{P_j \mid j < i\} & \\
\text{Operator:} & A_i^2 & \text{preconds:} & \{P_i\} & \text{adds: } \{G_i\} \\
 & & \text{deletes:} & \{P_j \mid j < i\} & \\
\text{Operator:} & A_* & \text{preconds:} & \{I_*\} & \text{adds: } \{G_*\} \\
 & & \text{deletes:} & \{I_i \mid \forall i\} \cup \{G_i \mid \forall i\} &
\end{array}$$

Example: With goals G_4, G_7, and G_*, the optimal solution is A_7^1, A_4^1, A_*, A_4^2, A_7^2.

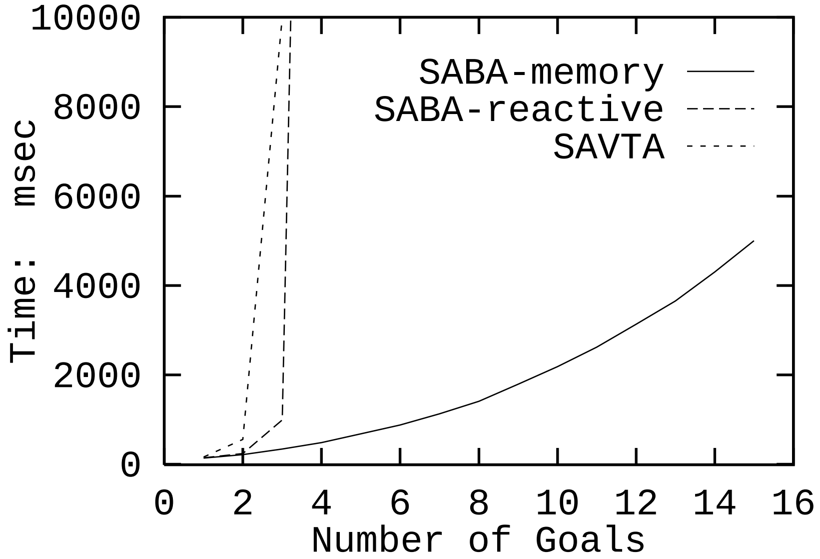
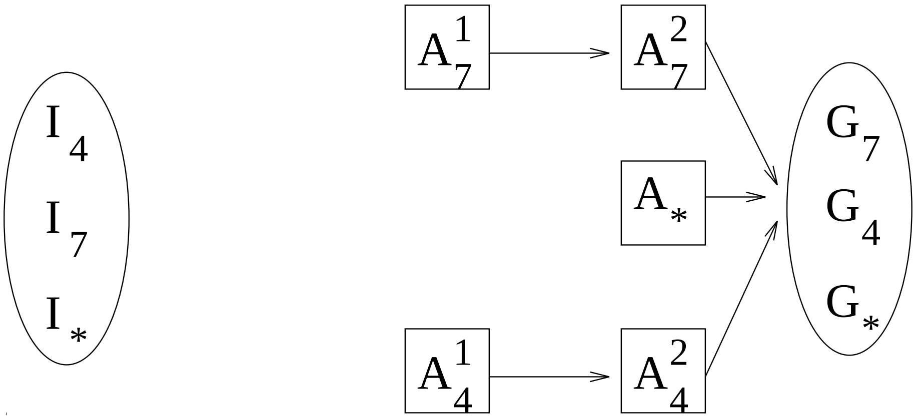


Diagram: $D^m S^{2*}$



D^0-side-effect: The SABA's fail

$$\begin{aligned} \text{Operator:}\ & A \\ \text{preconds:}\ & \{\} \\ \text{adds:}\ & \{G_i \mid 0 < i \leq 15\} \\ \text{deletes:}\ & \{\} \end{aligned}$$

- All goals achieved by **one operator**.
- Advantageous to **apply as soon as possible**.

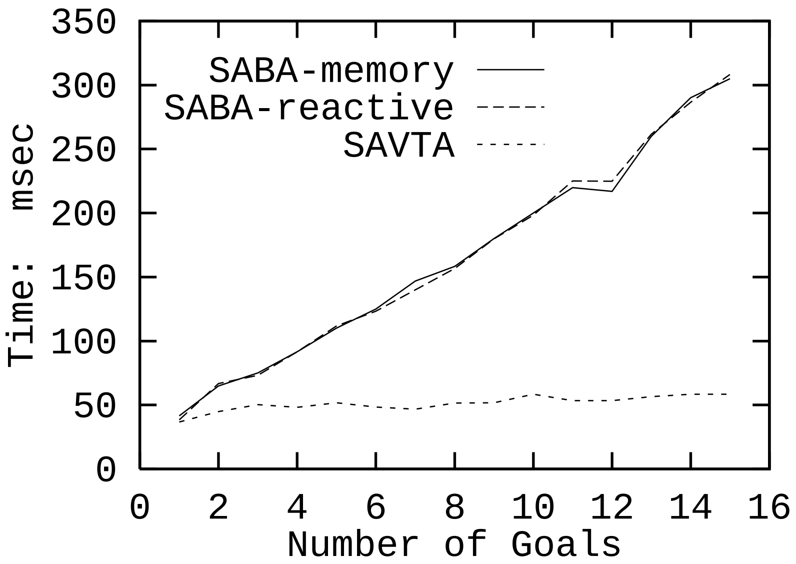
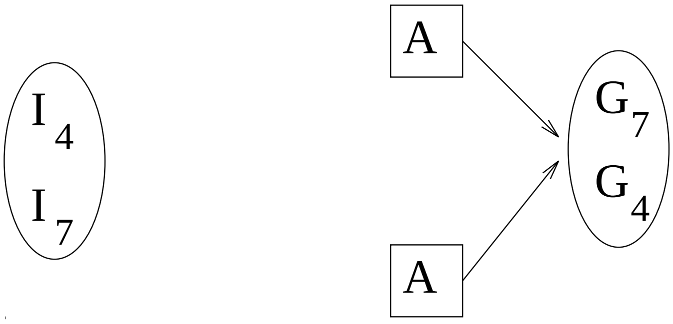


Diagram: D^0-side-effect



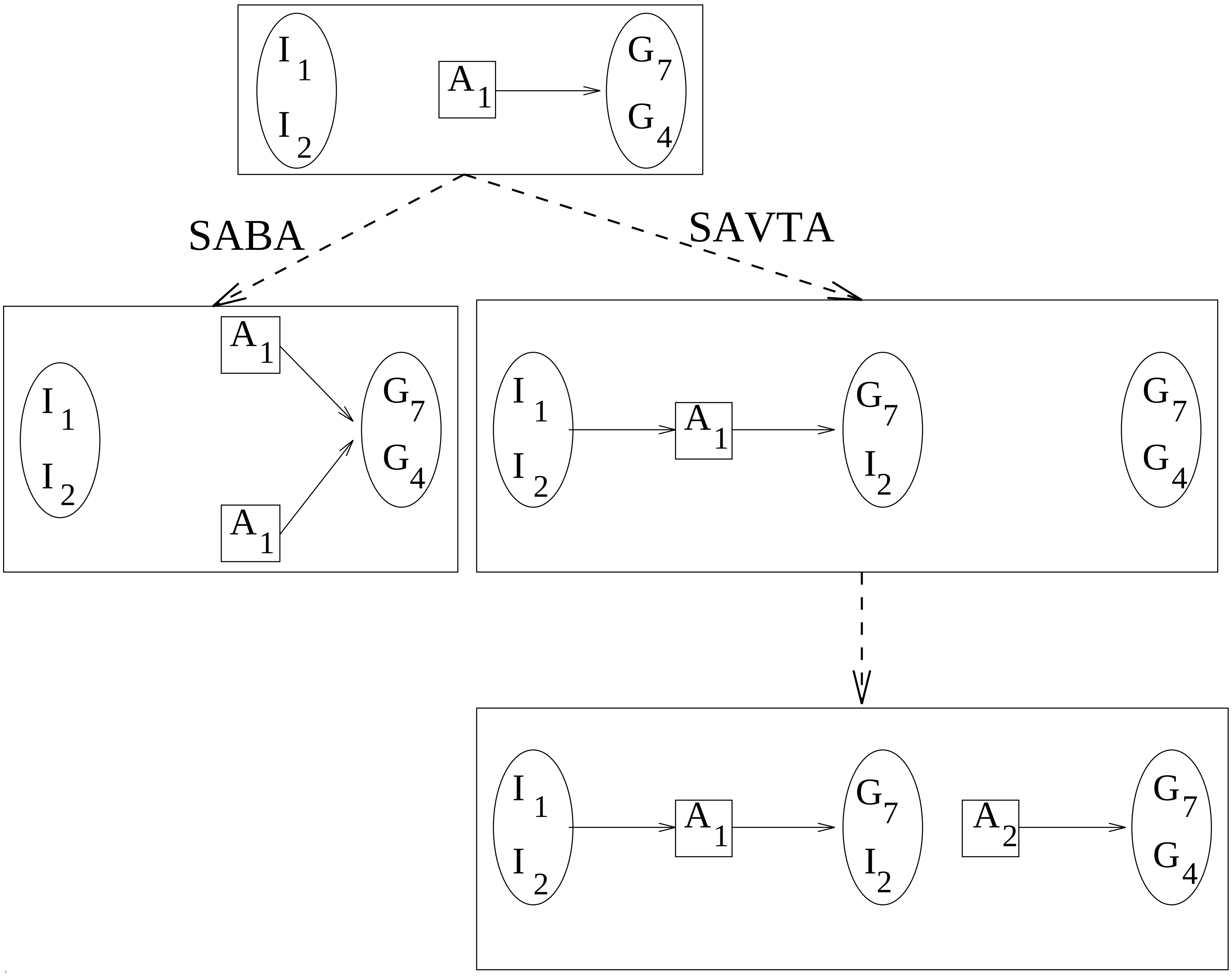
D^1-use-once: The SABA's fail again

$$\begin{aligned} \text{Operator:} \quad & A_i \\ \text{preconds:} \quad & \{I_i\} \\ \text{adds:} \quad & \{< g >\} \\ \text{deletes:} \quad & \{I_i\} \end{aligned}$$

- **Any operator** can achieve any goal.
- Each operator can only be **used once**.
- Advantageous to **apply as soon as possible**.

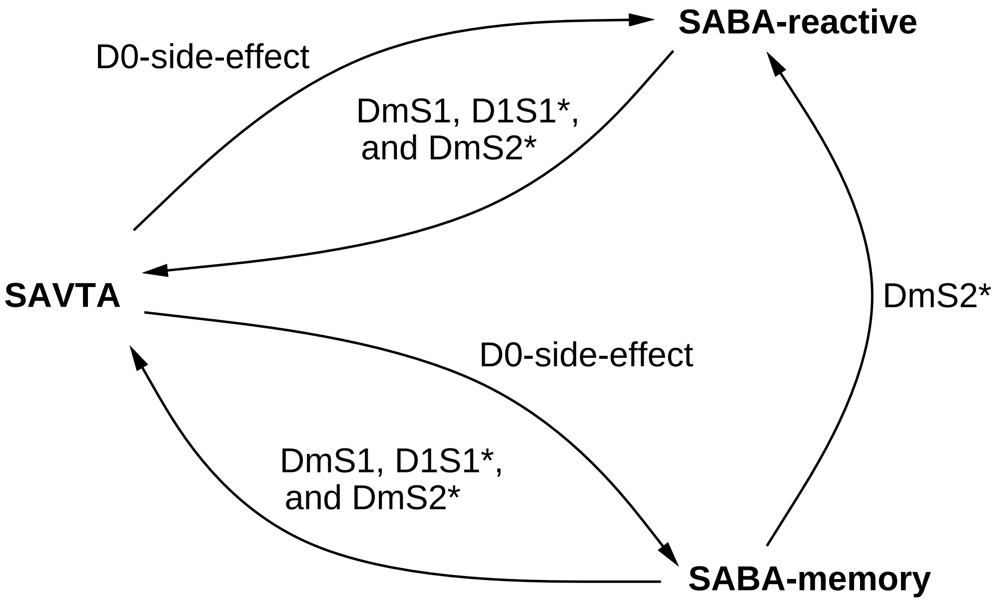
	SAVTA	SABA-reactive	SABA-memory
4 goals	0.8 sec	25.7 sec	25.3 sec
5 goals	1.2 sec	401 sec	404 sec

Diagram: D^1-use-once



No Dominance

Recall claim: **There is no single domain-independent search heuristic that performs more efficiently than others for all problems or in all domains.**



A new perspective on total-order planning

- PRODIGY searches using the state and a **total order**.
- It can **delay commitment** to operator orderings (SABA).
- It performs ≈linearly (sub-quadratically) in domains where TOPI and TOCL perform poorly.
- Success in these domains results from the ability to **delay ordering commitments**, NOT the ability to use **partial orders**.
- There is no single domain-independent search heuristic that performs more efficiently than others for all problems or in all domains.

Resulting Challenges

1. **Learn** which domain-independent search heuristics are suited to which domains and problems.
2. Develop planners that are **flexible** enough to use **different search strategies**.

FLECS – A New Planning Algorithm

- Planning with a **FLE**xible **C**ommitment **S**trategy
- Adds an explicit choice of whether to subgoal or apply
- Emobodies SAVTA, SABA, and hybrid commitment strategies

FLECS – A Top-Level View

1. Initialize.
2. Terminate if the goal statement has been satisfied.
3. Compute the pending goals and applicable operators.
   - Pending goals are the yet-to-be-achieved preconditions of operators that have been selected to be in the plan.
   - Applicable operators are those that have all their preconditions satisfied in the current state.
4. Choose subgoal or apply: *(backtrack point)*
   - To subgoal, go to step 5.
   - To apply, go to step 6.
5. Select a pending goal *(no backtrack point)* and an operator that can achieve it *(backtrack point)*; go to step 3.
6. Change the state as specified by an applicable operator *(backtrack point)*; go to step 2.

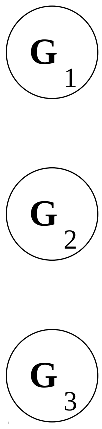
The variables and functions:

- $\mathcal{C}$ represents the *current internal state* of the planner.
- $\mathcal{G}$ is the set of *goals and subgoals* that the planner is aiming to achieve. These are the goals that are on the *fringe* of the subgoal tree. Goals in $\mathcal{G}$ may be goals that have not yet been planned for, or goals that have been achieved (perhaps trivially) but not yet used by the operator that needs them as one of its preconditions (i.e., this operator has not been applied yet).
- $\mathcal{P}$ is the set of *pending goals*: goals in $\mathcal{G}$ that may need to be planned for in the current state.
- $\mathcal{O}$ stands for the set of *instantiated operators* that have been selected to achieve goals and subgoals.
- $\mathcal{A}$ is the set of *applicable operators*: operators in $\mathcal{O}$ whose preconditions are all satisfied in the current state and which are needed in the current state to achieve some goal.
- For a goal G, $a(G)$ is the set of its *ancestor goal sets* – the sequences of goals that caused G to become a member of $\mathcal{G}$. Trivially, a goal is an ancestor of each of the preconditions of the operator selected to achieve this goal. $a(G)$ is a set of sets because G can have different sets of ancestors. This concept will become clearer through the example.
- For an operator O, $c(O)$ is the set of goals which O was selected to achieve – its *causes*. Applying O establishes each member of $c(O)$. As illustrated below, the functions a and c are needed to determine which goals are pending and which operators are applicable. They are analogous to causal links used to determine threats in other planners.

The initial problem

Consider a planning problem with three literals in the goal statement, G_1, G_2, and G_3, i.e., $\mathcal{G} = \{G_1, G_2, G_3\}$. There is one literal in the initial state, G_7, i.e., $\mathcal{C} = \{G_7\}$. As none of the goals is true in the initial state, $\mathcal{P} = \mathcal{G}$. There are no operators selected, i.e., $\mathcal{O} = \emptyset$, and therefore also no operators applicable, i.e., $\mathcal{A} = \emptyset$. At this point, since they are all top-level goals, none of the goals has any ancestors: $\underline{a(G_1) = a(G_2) = a(G_3) = \emptyset}$. As there are no applicable operators, the next step must be to subgoal on one of the pending goals.

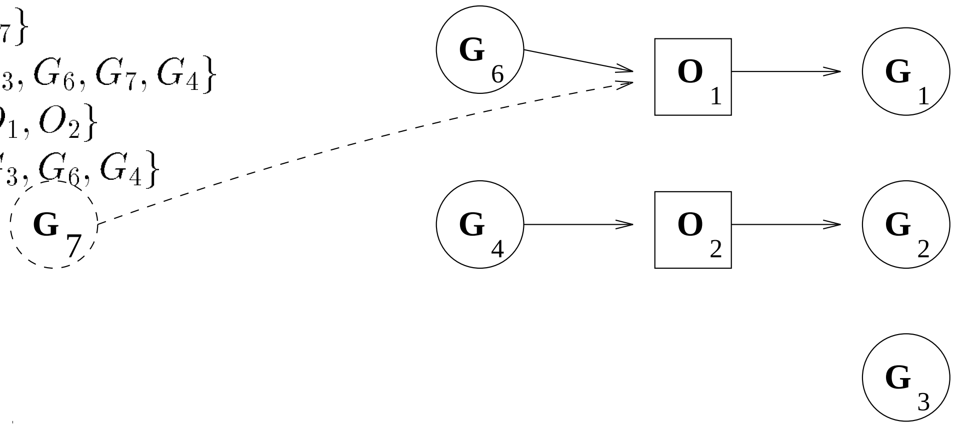
$\mathcal{C} = \{G_7\}$
$\mathcal{G} = \{G_1, G_2, G_3\}$
$\mathcal{O} = \emptyset$
$\mathcal{P} = \{G_1, G_2, G_3\}$
$\mathcal{A} = \emptyset$



After subgoaling on G_1 and G_2

Suppose that operator O_1, with preconditions G_6 and G_7, is selected to achieve G_1, while O_2 is chosen to achieve G_2 as indicated below. Note that the operators' preconditions replace their causes in the set of fringe goals $\mathcal{G}$; since G_7 is true in the current state, it is NOT included in the set of pending goals $\mathcal{P}$. Here G_1 is the *cause* of O_1, so $\underline{c(O_1) = \{G_1\}}$; similarly, $\underline{c(O_2) = \{G_2\}}$. The new goals all have nonempty ancestor sets: $\underline{a(G_6) = a(G_7) = \{\{G_1\}\}}$, and $\underline{a(G_4) = \{\{G_2\}\}}$. There are still no applicable operators: O_1 cannot be applied because $G_6 \notin \mathcal{C}$ and O_2 cannot be applied because $G_4 \notin \mathcal{C}$. Therefore, FLECS subgoals again.

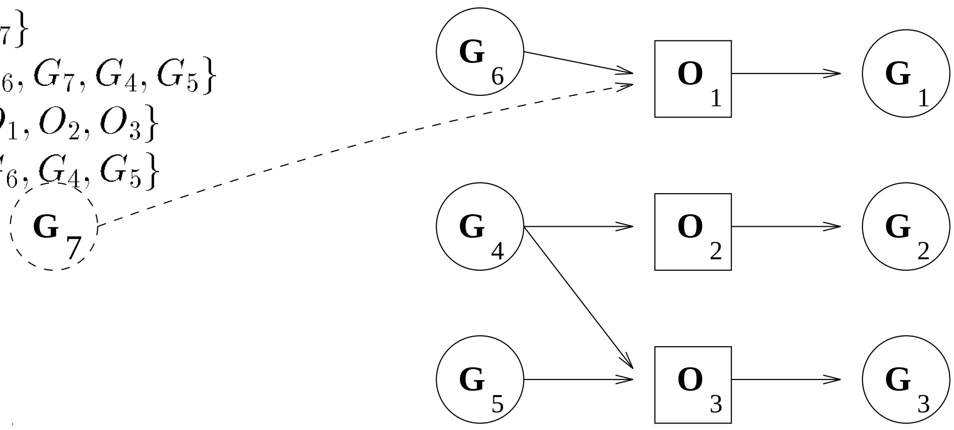
$\mathcal{C} = \{G_7\}$
$\mathcal{G} = \{G_3, G_6, G_7, G_4\}$
$\mathcal{O} = \{O_1, O_2\}$
$\mathcal{P} = \{G_3, G_6, G_4\}$
$\mathcal{A} = \emptyset$



After subgoaling on G_3

Suppose that the operator selected to achieve G_3 has preconditions G_4 and G_5. We now have $\underline{c(O_3) = \{G_3\}}$, and $\underline{a(G_5) = \{\{G_3\}\}}$. The causes of operators O_1 and O_2 do not change, so $c(O_1) = \{G_1\}$ and $c(O_2) = \{G_2\}$ as in the previous step. Similarly, $a(G_6)$ and $a(G_7)$ remain unchanged. However, G_4 now has *two* sets of ancestor goals: $\underline{a(G_4) = \{\{G_2\}, \{G_3\}\}}$. To understand the need to keep both ancestor sets, consider the possibility that G_2 could be achieved unexpectedly as a side-effect of some unrelated operator instead of being achieved by O_2 as planned for. In this case, G_4 would remain a pending goal since it would be needed to achieve G_3. Again, since there are no applicable operators, FLECS must subgoal on one of the pending goals, i.e., G_6, G_4, or G_5.

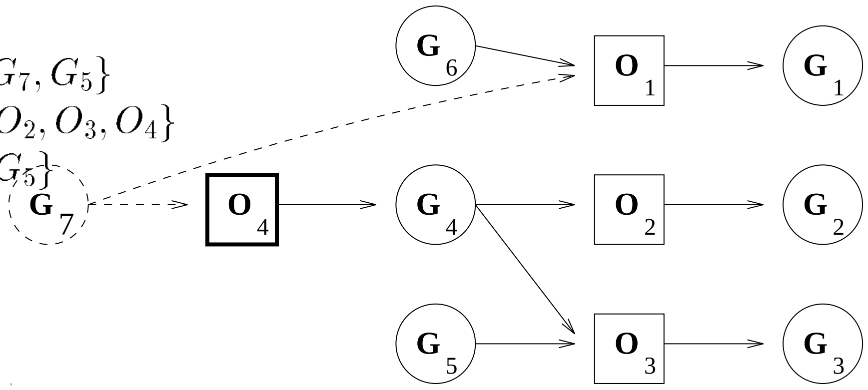
$\mathcal{C} = \{G_7\}$
$\mathcal{G} = \{G_6, G_7, G_4, G_5\}$
$\mathcal{O} = \{O_1, O_2, O_3\}$
$\mathcal{P} = \{G_6, G_4, G_5\}$
$\mathcal{A} = \emptyset$



After subgoaling on G_4

Suppose that O_4 — an operator with precondition G_7 — is selected to achieve G_4. Since G_7 is true in the current state, O_4 is our first applicable operator. Note that it is necessarily ordered before O_2 and O_3 since its cause is a precondition of these operators. As usual, the cause of the new operator is stored: $\underline{c(O_4) = \{G_4\}}$. In addition, the ancestors of G_7 must be augmented to include two new ancestor sets:
$\underline{a(G_7) = \{\{G_1\}, \{G_4, G_2\}, \{G_4, G_3\}\}}$. Although there is now an applicable operator, let us assume that FLECS chooses to delay its commitment to order O_4 as the first step in the plan and subgoals again on a pending goal.

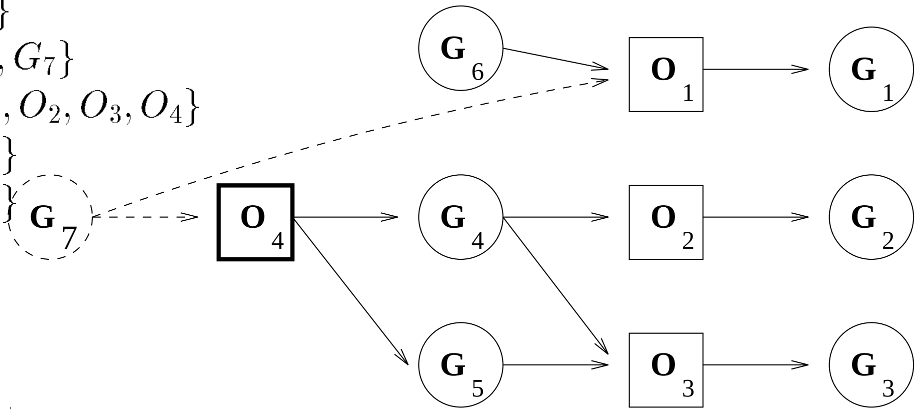
$\mathcal{C} = \{G_7\}$
$\mathcal{G} = \{G_6, G_7, G_5\}$
$\mathcal{O} = \{O_1, O_2, O_3, O_4\}$
$\mathcal{P} = \{G_6, \cancel{G_7}, G_5\}$
$\mathcal{A} = \{O_4\}$



After subgoaling on G_5

Suppose that operator O_4 can also achieve G_5 and that it is selected to do so. We now need to update both the causes of this operator and the ancestors of its precondition: $\underline{c(O_4) = \{G_4, G_5\}}$ and $\underline{a(G_7) = \{\{G_1\}, \{G_4, G_2\},}$ $\underline{\{G_4, G_3\}, \{G_5, G_3\}\}}$. Now rather than subgoaling on the last remaining pending goal (G_6), let us apply O_4. Note that this decision corresponds to an early commitment in terms of ordering the operators O_1, O_4, and any operators later selected to achieve G_6 which are unordered by the current planning constraints. FLECS changes here from its delayed-commitment strategy to an eager-commitment strategy.

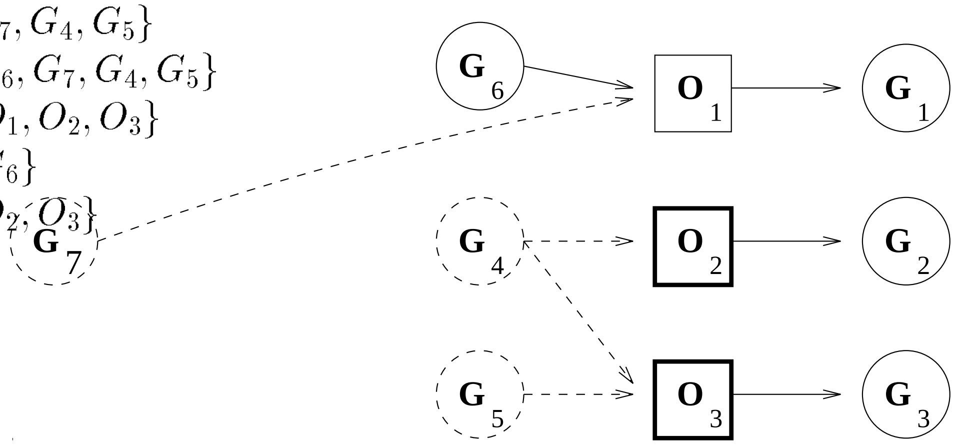
$\mathcal{C} = \{G_7\}$
$\mathcal{G} = \{G_6, G_7\}$
$\mathcal{O} = \{O_1, O_2, O_3, O_4\}$
$\mathcal{P} = \{G_6\}$
$\mathcal{A} = \{O_4\}$



After applying O_4

Since operator O_4 was applied in order to achieve goals G_4 and G_5, they are both true in the current state and back on the fringe of the goal tree, i.e., they are in $\mathcal{C}$ and $\mathcal{G}$. Notice that they stay in $\mathcal{G}$ until eventually they have been "consumed" by O_2 and O_3. However, since they are true in the current state, they are not pending goals. Since G_7 is once again the precondition of only one selected operator, $\underline{a(G_7) = \{\{G_1\}\}}$ as before. O_2 and O_3 are now applicable as their preconditions are all true in the current state thanks to O_4. Let us assume that FLECS maintains the eager-commitment strategy and continues applying applicable operators. FLECS orders O_2 before O_3, since O_3 deletes a precondition of O_2 (effects are not shown).

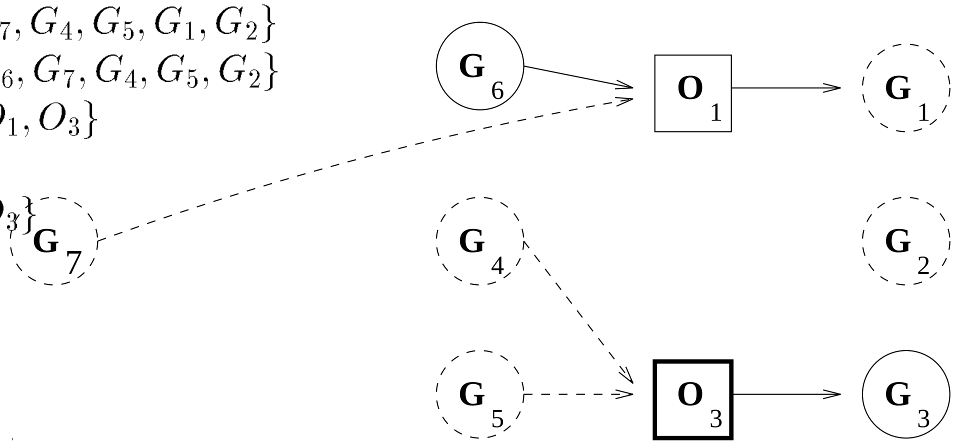
$\mathcal{C} = \{G_7, G_4, G_5\}$
$\mathcal{G} = \{G_6, G_7, G_4, G_5\}$
$\mathcal{O} = \{O_1, O_2, O_3\}$
$\mathcal{P} = \{G_6\}$
$\mathcal{A} = \{O_2, O_3\}$



After applying O_2

Suppose that, although it was not selected to do so, operator O_2 achieves G_1 as a side-effect. Perhaps O_2 has a conditional effect that was not visible to the planner, or perhaps O_1 simply looked more promising than O_2 as an operator to achieve G_1 at the time when it was selected. In any case, G_1 is now in $\mathcal{C}$ and the planning done for it is no longer needed: G_6 is no longer a pending goal, since its sole ancestor is already in $\mathcal{C}$. This fortuitous achievement of a goal is the reason that we need to use the functions c and a to adjust the sets of pending goals $\mathcal{P}$ and applicable operators $\mathcal{A}$: it would be wasted effort for FLECS to plan to achieve G_6. Note that were G_6 a precondition of O_3 as well as O_1, it *would* be a pending goal since it would still be relevant to achieving G_3. At this point, only the ancestors of G_4 must be reset: $\underline{a(G_4) = \{\{G_3\}\}}$. Since there are no more pending goals, FLECS must now apply the last remaining applicable operator, O_3.

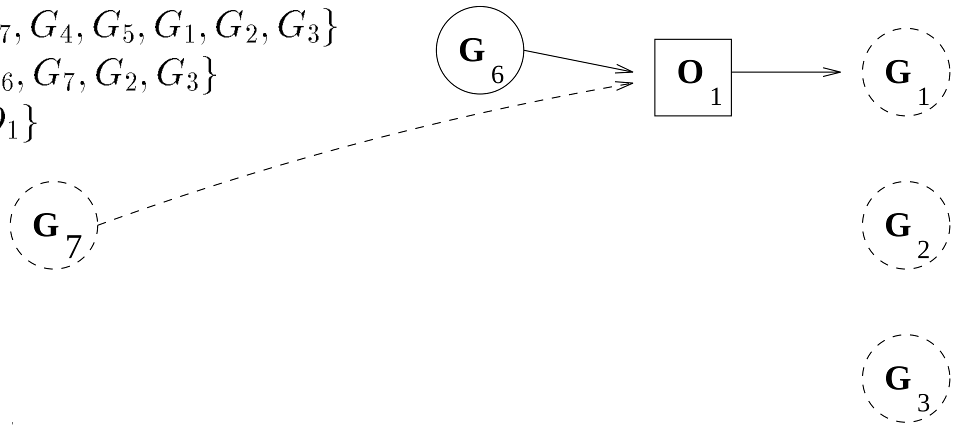
$\mathcal{C} = \{G_7, G_4, G_5, G_1, G_2\}$
$\mathcal{G} = \{G_6, G_7, G_4, G_5, G_2\}$
$\mathcal{O} = \{O_1, O_3\}$
$\mathcal{P} = \emptyset$
$\mathcal{A} = \{O_3\}$



After applying O_3 we are done

At this point all of the top level goals are true in the current state. Despite the fact that some of the planning tree remains, FLECS recognizes that there is no more work to be done and terminates. The final plan is O_4, O_2, O_3, which is the sequence of operators applied in the head-plan (not shown) corresponding to the steps in Figures , , and . An a posteriori algorithm can convert the sequence into a partially ordered plan capturing the dependencies: O_4, $\{O_2, O_3\}$.

$\mathcal{C} = \{G_7, G_4, G_5, G_1, G_2, G_3\}$
$\mathcal{G} = \{G_6, G_7, G_2, G_3\}$
$\mathcal{O} = \{O_1\}$
$\mathcal{P} = \emptyset$
$\mathcal{A} = \emptyset$



$\mathcal{C}$:	*current state*
$\mathcal{G}$:	*fringe goals*
$\mathcal{P}$:	*pending goals*
$\mathcal{O}$:	*instantiated operators*
$\mathcal{A}$:	*applicable operators*
a:	*ancestor goal sets*
c:	*causes*

1. Initialize:
   a. $\mathcal{G}$ = Goal Statement.
   b. $\mathcal{C}$ = Initial State.
   c. $\mathcal{O} = \emptyset$.
   d. $\forall G.a(G) = \emptyset$.
   e. $\forall O.c(O) = \emptyset$.

2. Terminate if Goal Statement $\subseteq \mathcal{C}$.

3. Compute applicable operators $\mathcal{A}$ and pending goals $\mathcal{P}$:
   a. $\mathcal{P} = \{G \in \mathcal{G} \mid G \notin \mathcal{C} \vee G \in \text{Initial State}\}$.
   b. $\mathcal{A} = \{A \in \mathcal{O} \mid pre(A) \subseteq \mathcal{C}\}$.

4. Adjust $\mathcal{P}$ and $\mathcal{A}$ to contain only active members:
   a. $\mathcal{P} = \mathcal{P} - \{P \in \mathcal{P} \mid \forall S \in a(P).\exists G \in S \text{ s.t. } G \in \mathcal{C}\}$.
   b. $\mathcal{A} = \mathcal{A} - \{A \in \mathcal{A} \mid \forall G \in c(A).[(G \in \mathcal{C}) \vee (\forall S \in a(G).\exists G' \in S \text{ s.t. } G' \in \mathcal{C})]\}$.

5. Subgoal or Apply:
   a. **Set or reset toggle to** *sub* **or** *app*, i.e. Set default to delayed or eager commitment.
   b. If $\mathcal{A} = \emptyset$, go to step 6.
   c. If $\mathcal{P} = \emptyset$, go to step 7.
   d. Choose to apply or to subgoal (*backtrack point*):
      - If **toggle** $= sub \wedge P \not\subseteq \mathcal{C}$, subgoal first: go to step 6.
      - If **toggle** $= app$, apply first: go to step 7.

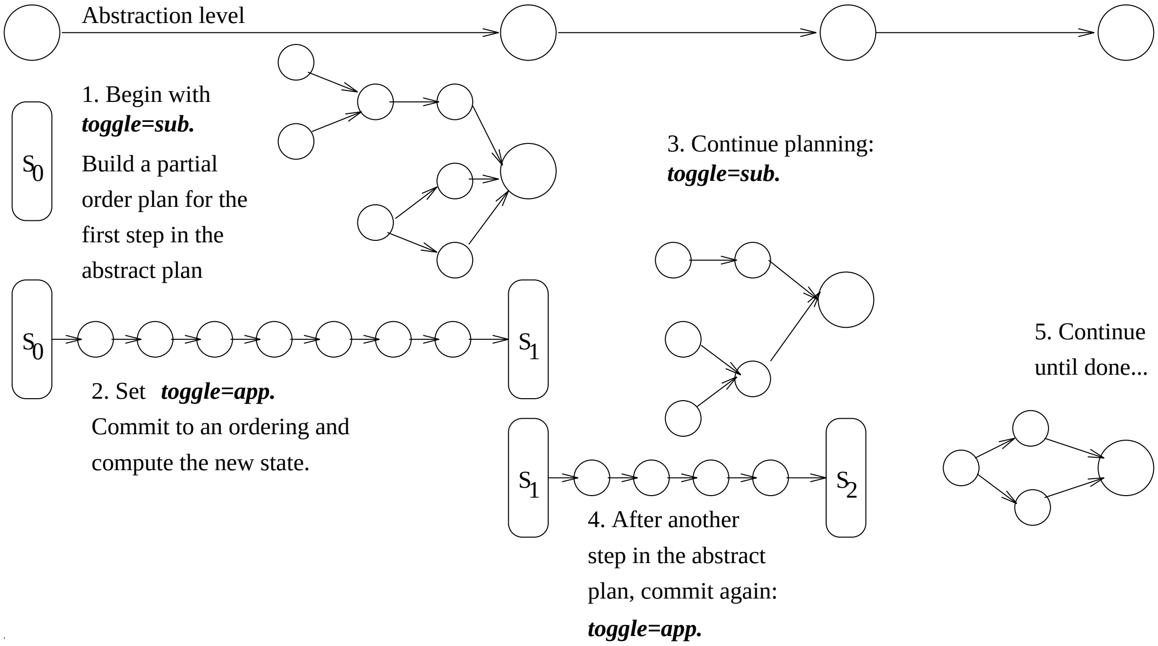
6. Choose a goal P from $\mathcal{P}$ (*not a backtrack point*).
   - Choose a goal not true in the Current State using means-ends analysis.

   a. Get the set $\mathcal{R}$ of relevant instantiated operators that could achieve P.
   b. If $\mathcal{R} = \emptyset$ then
      i. $\mathcal{P} = \mathcal{P} - \{P\}$.
      ii. If $\mathcal{P} = \emptyset$ then fail (i.e., backtrack).
      iii. Go to step 6.
   c. Choose an operator O from $\mathcal{R}$ (*backtrack point*).
      - Choose the operator with minimum conspiracy number $= \mid pre(O) - \mathcal{C} \mid$.
   d. $\mathcal{O} = \mathcal{O} \cup \{O\}$.
   e. $\mathcal{G} = (\mathcal{G} - \{P\}) \cup pre(O)$.
   f. $c(O) = c(O) \cup \{P\}$.
   g. $\forall G \in pre(O).a(G) = a(G) \cup \{\{P\} \cup S \mid S \in a(P)\}$.
   h. Go to step 3.

7. Choose an operator A from $\mathcal{A}$ (*backtrack point for interactions*).
   - Use heuristic to find operators with fewer interactions – similar to the SABA heuristic.

   a. Apply A: $\mathcal{C} = (\mathcal{C} \cup add(A)) - del(A)$
   b. $\mathcal{O} = \mathcal{O} - \{A\}$.
   c. $\forall G \in pre(A).a(G) = a(G) - \{S \in a(G) \mid S \cap c(A) \neq \emptyset\}$.
   d. $\mathcal{G} = (\mathcal{G} \cup c(A)) - \{G \in pre(A) \mid a(G) = \emptyset\}$.
   e. $c(A) = \emptyset$.
   f. Go to step 2.

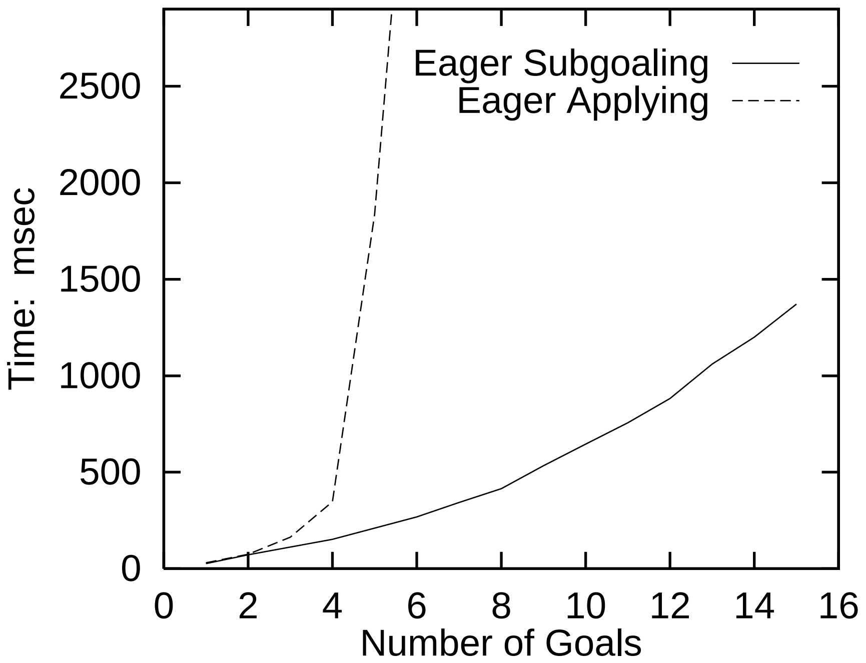
Using abstraction information to guide changes to *toggle*.



Eagerly Subgoaling Can Be Better

Operator:	**paint-white** <obj>	**paint-yellow** <obj>	...	**paint-black** <obj>
preconds:	(usable white)	(usable yellow)	...	(usable black)
adds:	(white <obj>)	(yellow <obj>)	...	(black <obj>)
deletes:		(usable white)	...	(usable white)
			...	(usable yellow)
			⋮	⋮
				(usable brown)

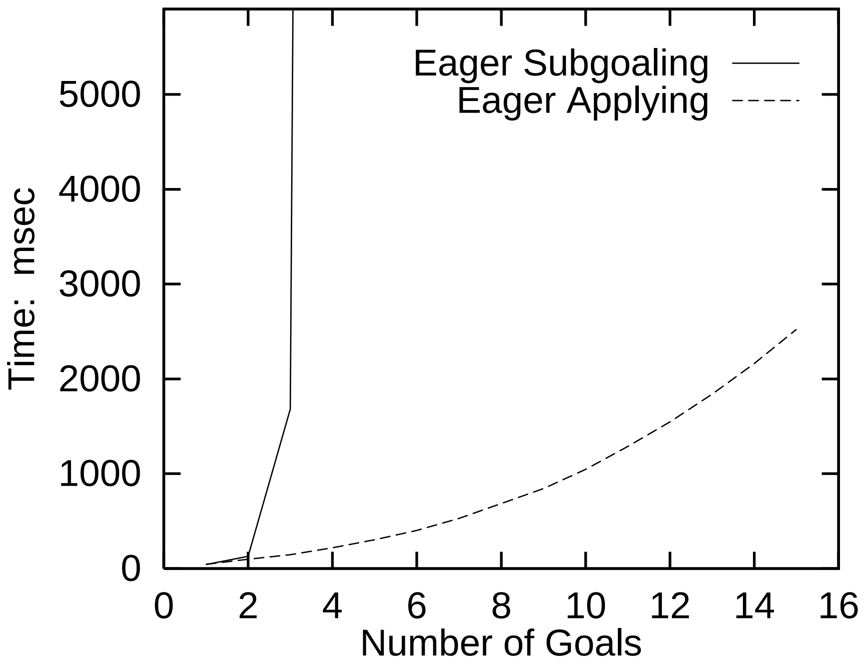
$\boxed{\text{Isomorphic to } D^m S^1}$



Eagerly Applying Can Be Better

Operator:	**paint-with-brush1** <parts> <color>	...	**paint-with-brush8** <parts> <color>
preconds:	(unused brush1)	...	(unused brush8)
adds:	(painted <parts> <color>)	...	(painted <parts> <color>)
deletes:	(unused brush1)	...	(unused brush8)

Isomorphic to D^1-use-once



An Intermediate Heuristic

Operator:	**designate-roller** <wall> <roller> <color>	**fill-roller** <roller> <color>	**paint-wall** <wall> <roller> <color>
preconds:	(clean <roller>) (needs-painting <wall>)	(clean <roller>) (chosen <roller> <color>)	(ready <wall> <roller> <color>) (filled-with-paint <roller> <color>)
adds:	(ready <wall> <roller> <color>) (chosen <roller> <color>)	(filled-with-paint <roller> <color>)	(painted <wall> <color>)
deletes:		(clean <roller>)	(ready <wall> <roller> <color>) (needs-painting <wall>)

Initial State

(needs-painting wallA)
(needs-painting wallB)
(needs-painting wallC)
(needs-painting wallD)
(needs-painting wallE)
(clean roller1)
(clean roller2)

Goal Statement

(painted wallA red)
(painted wallB red)
(painted wallC red)
(painted wallD green)
(painted wallE green)

An Optimal Solution

<Designate-Roller wallA roller1 red>
<Designate-Roller wallB roller1 red>
<Designate-Roller wallC roller1 red>
<Fill-Roller roller1 red>
<Paint-Wall wallA roller1 red>
<Paint-Wall wallB roller1 red>
<Paint-Wall wallC roller1 red>
<Designate-Roller wallD roller2 green>
<Designate-Roller wallE roller2 green>
<Fill-Roller roller2 green>
<Paint-Wall wallD roller2 green>
<Paint-Wall wallE roller2 green>

	time(sec)	solution
eager applying	500	no
eager subgoaling	500	no
variable strategy	4	yes

Take-home Lessons

- Commitment strategies in planning (eager vs. delayed) are orthogonal to plan-step ordering strategies (total vs. partial).
- There is no single domain-independent search heuristic that performs more efficiently than others for all problems or in all domains.
- FLECS adds the commitment strategy as an explicit choice point.