TD3 : STRIPS & Graphplan

EXERCICES

1. Define a simple planning problem, called SWAP, in which the objective is to swap the values of two parameters X and Y by exclusively assigning values to parameters.
2. Show that STRIPS algorithm is incomplete.

Algorithm 1: STRIPS$(\mathcal{O}, s, g)$

```
1  begin
2      π ← [];
3      repeat
4          relevant ← {a | a is relevant for g};
5          if relevant = ∅ then
6              return ⊥;
7          else
8              Nondeterministically choose a ∈ relevant;
9              π' ← STRIPS(O, s, precond(a));
10             if π' ≠ ⊥ then
11                 s ← γ(s, π' · a);
12                 π ← π · π' · a;
13             else
14                 return ⊥;
15     until s satisfies g ;
16     return π;
17 end
```

3. Suppose we run GRAPHPLAN on the SWAP problem described in Exercice1.
   (a) How many actions does it generate at level 1 of the planning graph ?
   (b) Expand the planning graph out of two levels and draw the result.
   (c) What is the first level at which GRAPHPLAN calls Extract ? Explain why it is not called before.
   (d) At what level will GRAPHPLAN find a solution ? What solution will it find ?

(e) What is the first level of the graph at which the number of actions reaches its maximum ?

Algorithm 2: Expand($[P_0, A_1, \mu A_1, P_1, \mu P_1, \ldots, A_{i-1}, \mu A_{i-1}, P_{i-1}, \mu P_{i-1}]$)

$A_i \leftarrow \{a \in A \mid \text{precond}(a) \subseteq P_{i-1} \wedge \text{precond}^2(a) \cap \mu P_{i-1} = \emptyset\}$;
$P_i \leftarrow \{p \mid \exists a \in A_i,\ p \in \text{effects}^+(a)\}$;
$\mu A_i \leftarrow \{(a,b) \in A_i^2 \mid \text{effects}^-(a) \cap [\text{precond}(b) \cup \text{effects}^+(b)] \neq \emptyset$
$\wedge\ \text{effects}^-(b) \cap [\text{precond}(a) \cup \text{effects}^+(a)] \neq \emptyset$
$\wedge\ \exists (p,q) \in \mu P_{i-1},\ p \in \text{precond}(a) \wedge q \in \text{precond}(b)\}$;
foreach $a \in A_i$ **do**
  link a with preconditions arcs to $\text{precond}(a) \in P_{i-1}$;
  link positive arcs to $\text{effects}^+(a)$ and negative arcs to $\text{effects}^-(a) \in P_i$;
return $[P_0, A_1, \mu A_1, P_1, \mu P_1, \ldots, A_i, \mu A_i, P_i, \mu P_i]$;

Algorithm 3: Extract(G, g, i)

if $i = 0$ **then return** [];
if $g \in \nabla(i)$ **then return** $\perp$;
$\pi_i \leftarrow \text{GP} - \text{Search}(G, g, \emptyset, i)$;
if $\pi_i \neq \perp$ **then return** *(π_i)*;
$\nabla(i) \leftarrow \nabla(i) \cup \{g\}$;
return $\perp$;

Algorithm 4: GP-search(G, g, π_i, i)

if $g = \emptyset$ **then**
  $\Pi \leftarrow$ Extract$(G, \bigcup\{\text{precond}(a) \mid \forall a \in \pi_i\}, i-1)$;
  if $\Pi = \perp$ **then return** $\perp$;
  return $\Pi.[\pi_i]$;
else
  select any $p \in g$;
  resolvers $\leftarrow \{a \in A_i \mid p \in \text{effects}^+(a) \wedge \forall b \in \pi_i,\ (a,b) \notin \mu A_i\}$;
  if resolvers $= \emptyset$ **then return** $\perp$;
  Nondeterministically choose $a \in$ resolvers;
  return GP-Search$(G, g - \text{effects}^+(a), \pi_i \cup \{a\}, i)$;

Algorithm 5: Graphplan$(\mathcal{O}, s_0, g)$

```
1  i ← 0 ; ∇ ← ∅ ; P_0 ← s_0 ; G ← [P_0];
2  isSolvable ← g ⊆ P_i ∧ g^2 ∩ μP_i = ∅;
3  isExpendable ← (isSolvable ∨ isFixedPoint) = false;
4  while IsExpandable do
5  |  i ← i + 1;
6  |  G ← Expand(G);
7  if isSolvable = false then ⊥;
8  Π ← Extract(G, g, i);
9  if isFixedPoint then
10 |  η ← |∇(κ)|
11 else
12 |  η ← 0
13 while Π = ⊥ do
14 |  i ← i + 1;
15 |  G ← Expand(G);
16 |  Π ← Extract(G, g, i);
17 |  if Π = ⊥ ∧ isFixedPoint then
18 |  |  if η = |∇(κ)| then return ⊥
19 |  η ← |∇(κ)|;
20 return Π;
```

4. A planning problems is defined as a tuple $(\mathcal{O}, s_0, g)$ but planning algorithms use *actions*. Explain how to generate a set of actions from planning problems. What is the corresponding mathematical problem ? Propose an algorithm.

ANSWERS TO EXERCICES

1.

```
1  (preconds ASSIGN
2     (params
3        (<x> VAR)(<xval> VALUE)(<y> VAR)(<yval> VALUE))
4     (preconds
5        (eq <x> <xval>)(eq <y> <yval>))
6     (effects
7        (eq <x> <yval>)(del eq <x> <xval>)))
```

```
1  (a VALUE)
2  (b VALUE)
3  (c VALUE)
4  (X VAR)
5  (Y VAR)
6  (Z VAR)
7  (preconds
8     (eq X a)(eq Y b)(eq Z c))
9  (effects
10    (eq X b)(eq Y a))
```

2. Xa denotes that the value of X is a. Let $XaYb$ denote the assignement of Y value to X.
There are four solution plans : $[ZcXa; XaYb; YbZa]$, $[ZcYb; YbXa; XaZb]$ etc. None of them is found by STRIPS. Hence, STRIPS is incomplete.

   Indeed, there are only two actions relevant for the goal $(eq\ X\ b)$ belonging to a solution plan :$XaYb$ and $XaZb$. Let see what happens if $XaZb$ is chosen (reasoning with $XaYb$ is similar) : STRIPS has to find a solution for $XaZb$ preconditions, that is to say $(eq\ X\ a)$ and $(eq\ Z\ b)$ (line 9 in algorithm 1) : the returned plan is $\pi' = [ZcYb]$. Line 11 sets s to $\{(eq\ X\ b), (eq\ Y\ b), (eq\ Z\ b)\}$ and $\pi = [ZcYb; XaZb]$ at line 12. Therefore, the remaining goal $(eq\ Y\ a)$ becomes unachievable. Symetrically, if we try to achieve $(eq\ Y\ a)$ first, the only two relevant actions included in solution plans are $YbZa$ and $YbXa$. Suppose that $YbZa$ is chosen ($YbXa$ leads to similar conclusions) : STRIPS has to find a solution for $YbZa$ preconditions, that is to say $(eq\ Y\ b)$ and $(eq\ Z\ a)$ (line 9 in algorithm 1) : the returned plan is $\pi' = [ZcXa]$. Line 11 sets s to $\{(eq\ X\ a), (eq\ Y\ a), (eq\ Z\ a)\}$ and $\pi = [ZcXa; YbZa]$ at line 12. Therefore, the remaining goal $(eq\ X\ b)$ becomes unachievable. As a consequence, whatever the chosen goal, STRIPS is unable to find any solution plans.

3. The planning graph produced by GRAPHPLAN with the SWAP problem is showed in figure 1.
   (a) At level 1, 6 actions are generated.

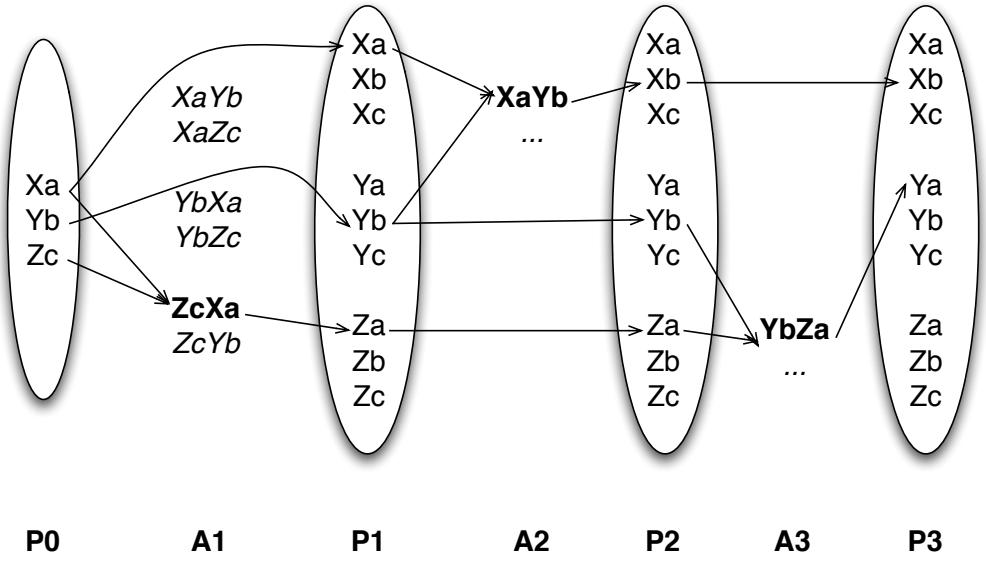

FIG. 1 – Planning graph generated by GRAPHPLAN

1

(b) See figure 1.

(c) At level 1, Ya and Xb are mutex because $XaYb$ and $YbXa$ are the only actions producing them and they are clearly dependent. Hence, Extract is not called at level P1. Likewise, it is not called at P2. Indeed, any action producing Ya has necessarily Ya as precondition. In order to show that Ya and Xb are still mutex at P2, it suffices to show that any action having Xb as positive effect has a mutex precondition with Ya :
- α_{Xb} and α_{Ya} are mutex actions ;
- Actions satisfacing the pattern X_Yb produce Yb. All of them have Yb as precondition. Now, Yb and Ya are mutex at P1 ;
- Actions satisfacing the pattern X_Zb produce Yb. All of them have Zb as precondition. We must show that Zb and Ya are mutex at P1. This is true because the only action that makes Ya at P1 is $YbXa$. Likewise, there is only one action producing Zb at level P1 : $ZcYb$. It is easy to see that both are dependent.

(d) A solution is reached at level 3. One possible solution is

$$[ZcXa; XaYb; YbZa]$$

Indeed, Xa, Yb and Za are not mutex at level P1 (α_{Xa}, α_{Yb} and $ZcXa$ are mutually independent). Therefore, the only way to have

Xb, Yb and Za in the nogood table at level P2 is to generate them with mutually dependent actions, which is not the case (α_{Za}, α_{Yb} and $XaYb$). Finally, Xb and Ya are not mutex at level P3 because α_{Xb} and $YbZa$ are independent and their preconditions are nonmutex.

(e) The planning graph levels off at level 2. A2 contains 36 actions.

4. Generating actions out of a planning problem amounts to gather all the constant symbols. In the SWAP problem, these symbols are $\{X, Y, a, b, c\}$. The next step consists in generating all t-tuples where t is the arity of the considered operator. For instance, the arity of ASSIGN is 4. This procedure will make correct actions such that ASSIGN(X, a, Y, b) etc. For convenience, we can start by ignoring the parameter types. Then, we have to generate all possible 4-tuple out of $n = 5$ symbols. This is equivalent to counting from 0 up to n^t ($5^4 = 625$) in base 5. For instance, $0 = b_4 \times 5^4 + b_3 \times 5^3 + b_2 \times 5^2 + b_1 \times 5^1 + b_0 \times 5^0$ where each $b_i = 0$ ($4 \geq i \geq 0$). b_i coefficients can be interpreted as the index of the elements of $\{X, Y, a, b, c\}$. The corresponding 4-tuples are $\{X, X, X, X\}$, $\{X, X, X, Y\}$, $\ldots$, $\{c, c, c, c\}$.

   More generally, suppose that we need to generate all $(a_1, \ldots, a_n)$ in which each a_j is one of hte symbols corresponding to a_j's type :

$$1 \leq a_j < m_j$$

   for $1 \leq j \leq n$, where the upper limits m_j might be different in different components of $(a_1, \ldots, a_n)$. Each m_j can be interpreted as the size of each a_j's symbol list. For the ASSIGN(a_1, a_2, a_3, a_4) operator, we have : $m_1 = m_3 = 2$ (X and Y) and $m_2 = m_4 = 3$ (a, b and c). Now, we can use the *mixed-radix generation*[1] (cf. algorithm 6).

Algorithm 6: Mixed-radix generation

1 Set $a_j \leftarrow 0$ for $0 \leq j \leq n$, and set $m_0 \leftarrow 2$.
2 Visit the n-tuples.
3 Set $j \leftarrow n$.
4 If $a_j = m_j - 1$, set $a_j \leftarrow 0$, $j \leftarrow j - 1$, and repeat this step.
5 If $j = 0$, terminate the algorithm. Otherwise, set $a_j \leftarrow a_j + 1$ and go back to step 2.

[1] Donald E. Knuth, in "The Art of Computer Programming", Volume 4, Fascicle 2, *Generating All Tuples and Permutations*, Addison-Wesley Educational Publishers Inc, 2005.