# Automated Planning & Artificial Intelligence

## Planning graphs & GRAPHPLAN algorithm

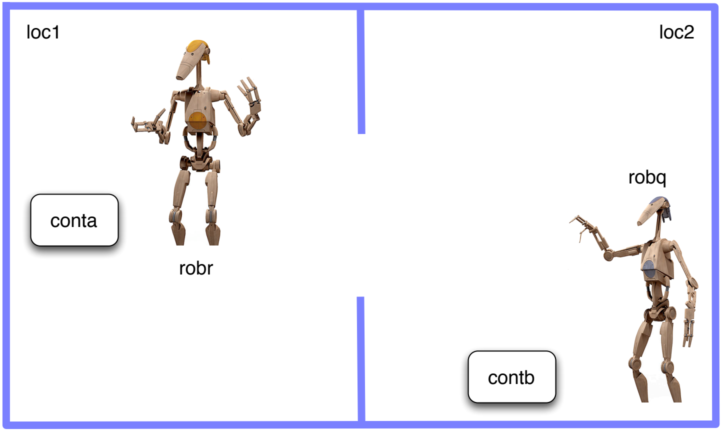
Humbert Fiorino

Humbert.Fiorino@imag.fr
http://membres-lig.imag.fr/fiorino

Laboratory of Informatics of Grenoble – MAGMA team

March 2011

## A simplified Dock-Worker Robot problem



## DWR operators

```
(operator MOVE
    (params (<r> ROBOT) (<l> LOCATION) (<l'>
LOCATION))
    (preconds
        (at <r> <l>)(adjacent <l> <l'>))
    (effects
        (at <r> <l'>)
        (del at <r> <l>)))
```

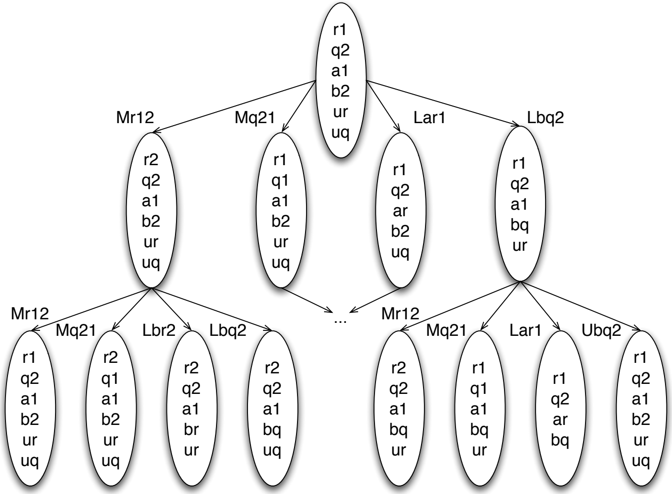
## DWR operators

```
(operator LOAD
    (params (<r> ROBOT) (<l> LOCATION) (<c>
CONTAINER))
    (preconds
        (at <r> <l>)(in <c> <l>)(unloaded <r>))
    (effects
        (loaded <r> <c>)
        (del in <c> <l>)(del unloaded <r>)))

(operator UNLOAD
    (params (<r> ROBOT) (<l> LOCATION) (<c>
CONTAINER))
    (preconds
        (at <r> <l>)(loaded <r> <c>))
    (effects
        (unloaded <r>)(in <c> <l>)
        (del loaded <r> <c>)))
```

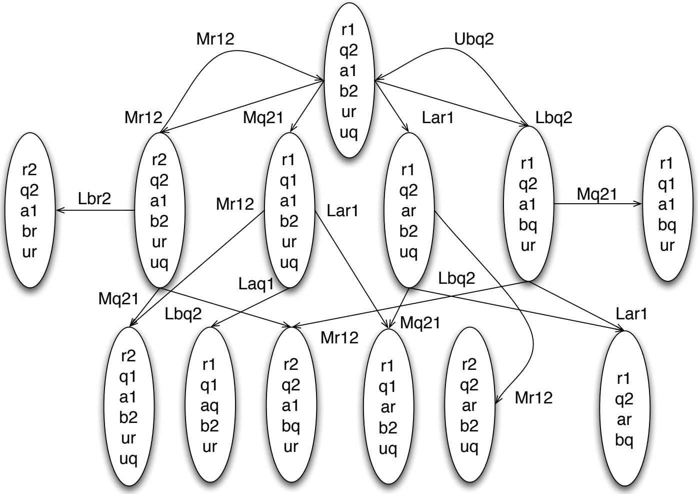
- To simplify representation, let us denote atoms by propositional symbols :
  - r1 and r2 stand for (at robr loc1) and (at robr loc2)
  - q1 and q2 stand for (at robq loc1) and (at robq loc2)
  - a1, a2, ar and aq stand for (int conta loc1), (in conta loc2), (loaded conta robr) and (loaded conta robq)
  - b1, b2, br and bq stand for (int contb loc1), (in contb loc2), (loaded contb robr) and (loaded contb robq)

- Let us also denote the 20 actions in A :
  - Mr12 is the action move(robr, loc1, loc2), Mr21 is the opposite, and Mq12 and Mq21 are the similar move action for robot robq
  - Lar1 is the action load(conta, robr, loc1), Lar2, Laq1 and Laq2 are the other load actions for conta in loc2 with contb. Lbr1, Lbr2, Lbq1 and Lbq2 are the load actions for contb
  - Uar1, Uar2, Uaq1, Uaq2, Ubr1, Ubr2, Ubq1, and Ubq2 are the unload actions

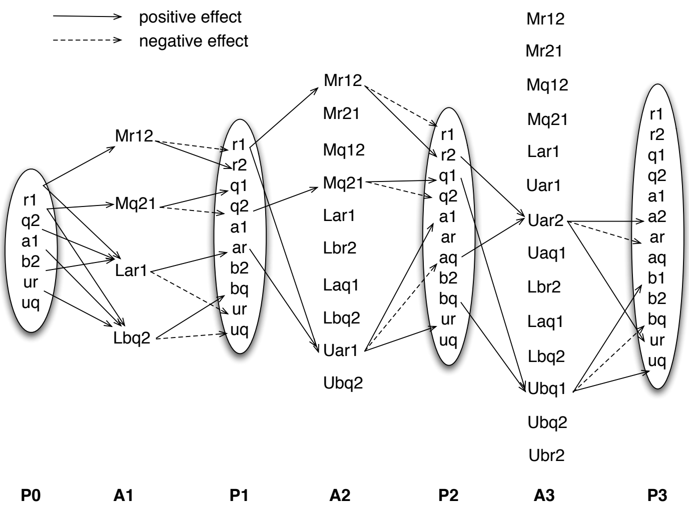
## Reachability tree



# Reachabilty graph



## Planning graph



## Planning graph

- A planning graph is a directed layered graph : arcs are permitted only from one layer to the next
- Nodes in level 0 correspond to the set $P_0$ of propositions denoting the initial state $s_0$ of the planning problem
- Level 1 contains two layers :
  1. an action level $A_1$ that is the set of actions (ground instance of operators) whose preconditions are nodes in $P_0$
  2. a proposition level $P_1$ that is defined as the union of $P_0$ and the sets of effects of actions in $A_1$

## Planning graph

- An action node in $A_1$ is connected with :
  - an incoming precondition arcs from its preconditions in $P_0$
  - outgoing arcs to its positive effects and to its negative effects in $P_1$
- Outgoing arcs are labeled positive or negative. Note that negative effects are not deleted from $P_1$, thus $P_0 \subseteq P_1$
- This process is iterated from one level to the next

## Persistence actions

- Inclusion of proposition levels is a way to represent inaction as well. $A_i$ levels contain actions that *could* occur and "dummy" actions – No-op – representing the fact that some world properties (propositions) persist in different situations
- To each proposition $p$ corresponds a no-op $\alpha_p$ :

$$\alpha_p = \begin{cases} \text{precond}(\alpha_p) = \{p\}, \\ \text{effect}^+(\alpha_p) = \{p\}, \\ \text{effect}^-(\alpha_p) = \{\}. \end{cases}$$

## Reachability analysis with planning graph

- A major contribution of Graphplan is a relaxation of the reachabiity analysis $\longrightarrow$ incomplete condition of reachability
- A goal is reachable from $s_0$ *only if* it appears in some node of the planning graph
- Weak reachability condition is compensated for by a low complexity : the planning graph is of polynomial size and can be built in polynomial time in the size of the input

## Layered plan

- A plan associated with a planning graph is no longer a sequence of actions : a plan is a **sequence of set of actions**

$$\Pi = [\pi_1, \dots, \pi_k]$$

  i.e. a *layered plan*
- $\pi_i \subseteq A_i$. $\pi_1$ is a subset of *independent actions* in $A_1$ that can be applied in **any** order to the initial state and can lead to a state that is a subset of $P_1$ etc.
- $\pi_k$ actions lead to a state meeting the goal.

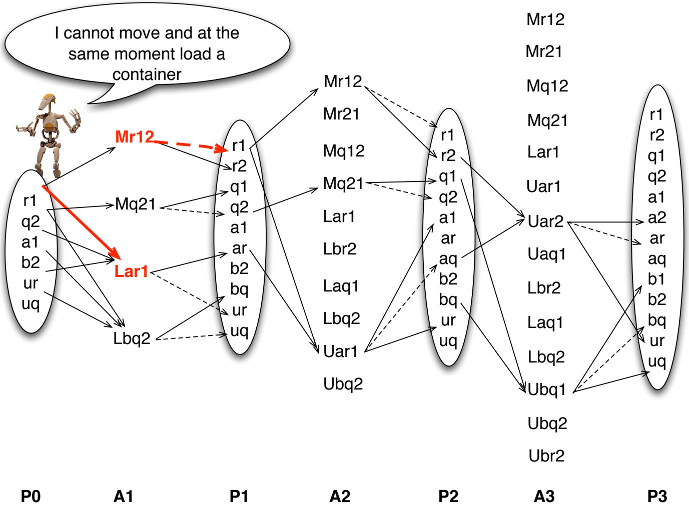
## Independent actions

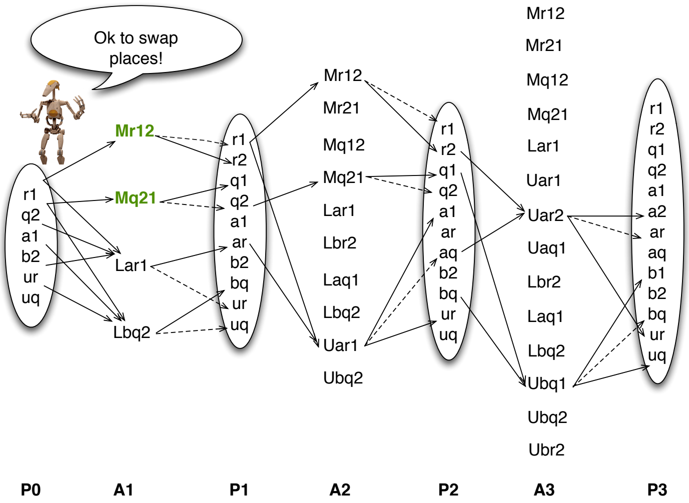
**Definition**

Two actions $(a, b)$ are independent iff :

- $\text{effects}^{-}(a) \cap [\text{precond}(b) \cup \text{effects}^{+}(b)] = \emptyset$
- $\text{effects}^{-}(b) \cap [\text{precond}(a) \cup \text{effects}^{+}(b)] = \emptyset$

A set of actions $\pi$ is independent when every pair of $\pi$ is independent.


I cannot move and at the same moment load a container
Mr12
Mq21
Lar1
Lbq2
r1
q2
a1
b2
ur
uq
r1
r2
q1
q2
a1
ar
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Lbr2
Laq1
Lbq2
Uar1
Ubq2
r1
r2
q1
q2
a1
ar
aq
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Uar1
Uar2
Uaq1
Lbr2
Laq1
Lbq2
Ubq1
Ubq2
Ubr2
r1
r2
q1
q2
a1
a2
ar
aq
b1
b2
bq
ur
uq
P0
A1
P1
A2
P2
A3
P3


Ok to swap places!
P0: r1 q2 a1 b2 ur uq
A1: Mr12 Mq21 Lar1 Lbq2
P1: r1 r2 q1 q2 a1 ar b2 bq ur uq
A2: Mr12 Mr21 Mq12 Mq21 Lar1 Lbr2 Laq1 Lbq2 Uar1 Ubq2
P2: r1 r2 q1 q2 a1 ar aq b2 bq ur uq
A3: Mr12 Mr21 Mq12 Mq21 Lar1 Uar1 Uar2 Uaq1 Lbr2 Laq1 Lbq2 Ubq1 Ubq2 Ubr2
P3: r1 r2 q1 q2 a1 a2 ar aq b1 b2 bq ur uq
P0
A1
P1
A2
P2
A3
P3

## Applicable actions

**Definition**

A set $\pi$ of independent actions is applicable to a state $s$ iff $\text{precond}(\pi) \subseteq s$. The result of applying the set $\pi$ to $s$ is defined as :

$$\gamma(s, \pi) = (s - \text{effects}^{-}(\pi)) \cup \text{effects}^{+}(\pi)$$

where

- $\text{effects}^{-}(\pi) = \bigcup\{\text{effects}^{-}(a) | \forall a \in \pi\}$
- $\text{effects}^{+}(\pi) = \bigcup\{\text{effects}^{+}(a) | \forall a \in \pi\}$
- $\text{precond}(\pi) = \bigcup\{\text{precond}(a) | \forall a \in \pi\}$

**Theorem**

*If a set $\pi$ of independent actions is applicable to s then, for any permutation $\{a_1, \ldots, a_k\}$ of the elements of $\pi$, this permutation is applicable to s and the state resulting from the application of $\pi$ to s is such that* $\gamma(s, \pi) = \gamma(\ldots\gamma(\gamma(s, a_1), a_2)\ldots, a_k)$.

## Definition

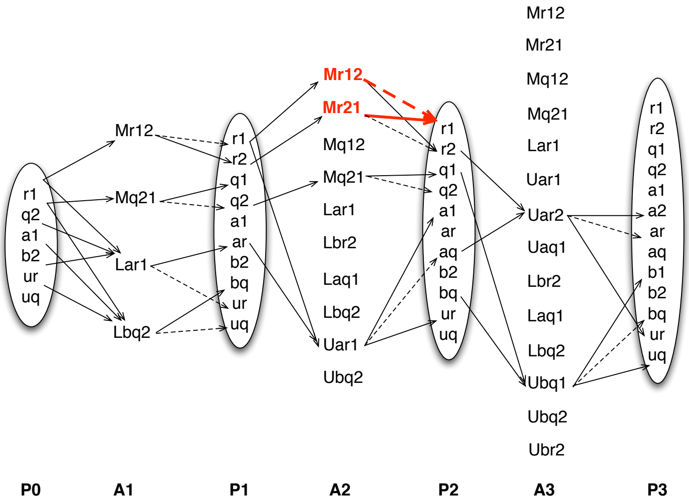
A layered plan is a sequence of action sets. The layered plan $\Pi = [\pi_1, \ldots, \pi_n]$ is a solution to a problem $(\mathcal{O}, s_0, g)$ iff :

1. each set $\pi_i \in \Pi$ is independent
2. the set $\pi_1$ is applicable to $s_0$, $\pi_2$ is applicable to $\gamma(s_0, \pi_1)$ etc.
3. $g \subseteq \gamma(\ldots\gamma(\gamma(s, \pi_1), \pi_2)\ldots, \pi_n)$.

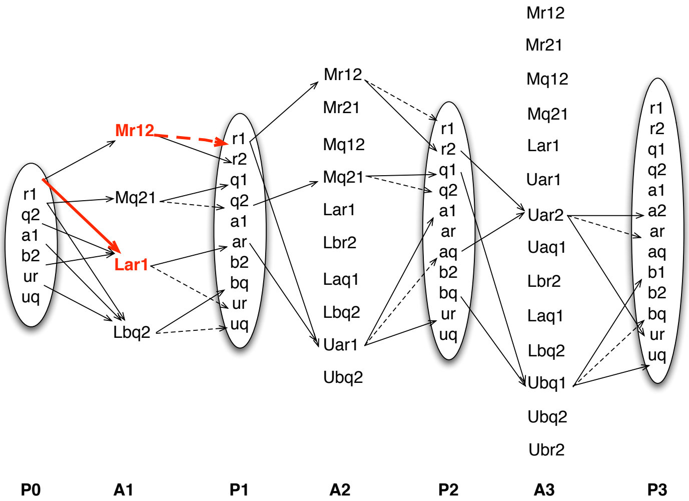
## Mutex

- The incompatibility relations between actions and between propositions in a planning graph, also called mutual exclusions or mutex relations are :
  - Interfering actions,
  - Inconsistent effects,
  - Actions having competing needs or inconsistent support.

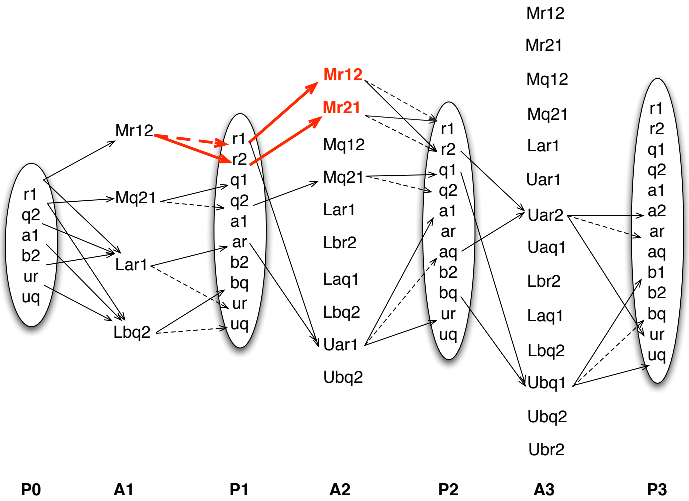
## Inconsistent effects



## Interference



## Inconsistent support & competing needs



**Definition**

- Two actions $a$ and $b$ in level $A_i$ are mutex if either $a$ and $b$ are dependent or if a precondition of $a$ is mutex with a precondition of $b$.
- Two propositions $p$ and $q$ in $P_i$ are mutex if every action in $A_i$ that has $p$ as a positive effect (including no-op actions) is mutex with every action that produces $q$, and there is no action in $A_i$ that produces both $p$ and $q$.

- Two dependent actions are necessarily mutex,
- However, two independent actions can be mutex !

- We will denote the set of mutex pairs in $A_i$ as $\mu A_i$, and the set of mutex pairs in $P_i$ as $\mu P_i$.

## Theorem

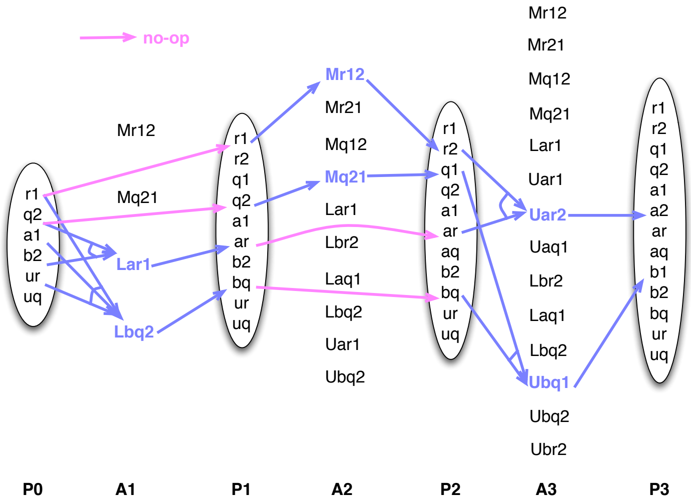
- *If two propositions p and q are in $P_{i-1}$ and $(p, q) \notin \mu P_{i-1}$ then $(p, q) \notin \mu P_i$*
- *If two actions a and b are in $A_{i-1}$ and $(a, b) \notin \mu A_{i-1}$ then $(a, b) \notin \mu A_i$*

- According to this result, propositions and actions monotonically increase from one level to the next, while mutex pairs monotonically decrease.
- These monotonicity properties are essential to the complexity and termination of the planning-graph techniques.

## Proof

- Every proposition $p$ in a level $P_i$ is supported by at least its no-op action $\alpha_p$. Two no-op actions are necessarily independent. If $p$ and $q$ in $P_i$ are such that $(p, q) \notin \mu P_{i-1}$ then $(\alpha_p, \alpha_q) \notin \mu A_i$. Hence, a nonmutex pair of propositions remains nonmutex in the following level.
- Similarly, if $(a, b) \notin \mu A_{i-1}$, then $a$ and $b$ are independent and their preconditions in $P_{i-1}$ are not mutex ; both properties remain valid at the following level.

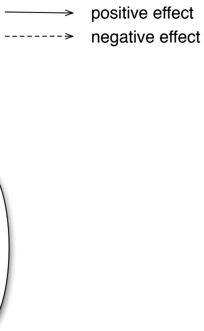
## A solution plan



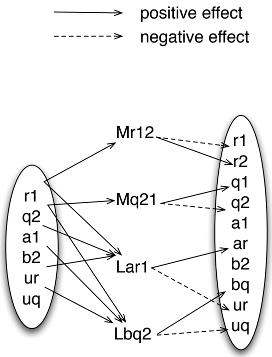
**Theorem**

*A set $g$ of propositions is reachable from $s_0$ only if there is in the corresponding planning graph a proposition layer $P_i$ such that $g \in P_i$ and no pair of propositions in $g$ are in $\mu P_i$.*

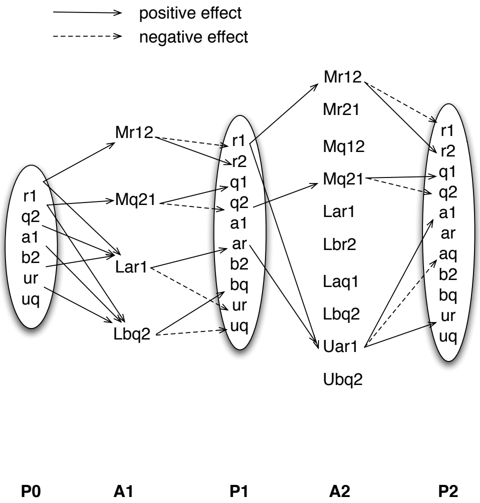
- The Graphplan performs a procedure close to iterative deepening, discovering a new part of the search space at each iteration ;
- It iteratively expands the planning graph by one level, then it searches backward from the last level of this graph for a solution ;
- The first expansion proceeds to a level $P_i$ in which all of the goal propositions are included and no pairs of them are mutex ;
- The iterative loop of graph expansion and search is pursued until either a plan is found or a failure termination condition is met.

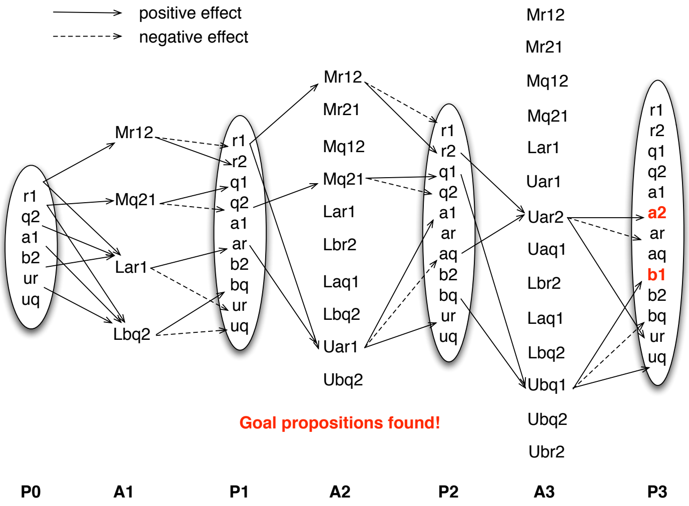


**P0**


positive effect
negative effect
Mr12
Mq21
Lar1
Lbq2
r1
q2
a1
b2
ur
uq
r1
r2
q1
q2
a1
ar
b2
bq
ur
uq

**P0** **A1** **P1**


positive effect
negative effect
r1
q2
a1
b2
ur
uq
Mr12
Mq21
Lar1
Lbq2
r1
r2
q1
q2
a1
ar
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Lbr2
Laq1
Lbq2
Uar1
Ubq2
r1
r2
q1
q2
a1
ar
aq
b2
bq
ur
uq
P0
A1
P1
A2
P2

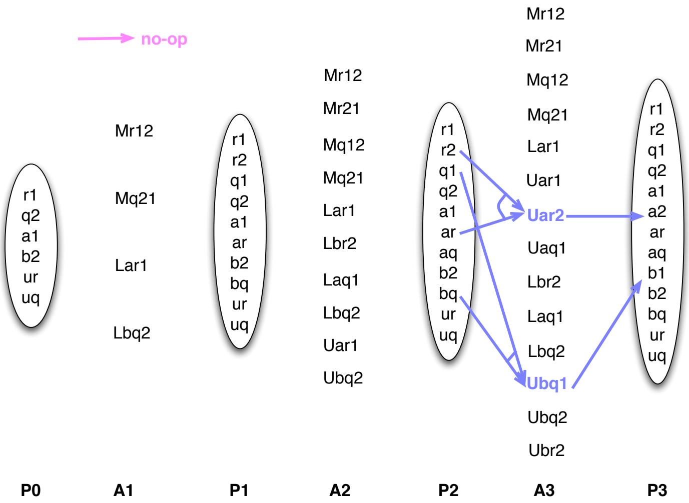


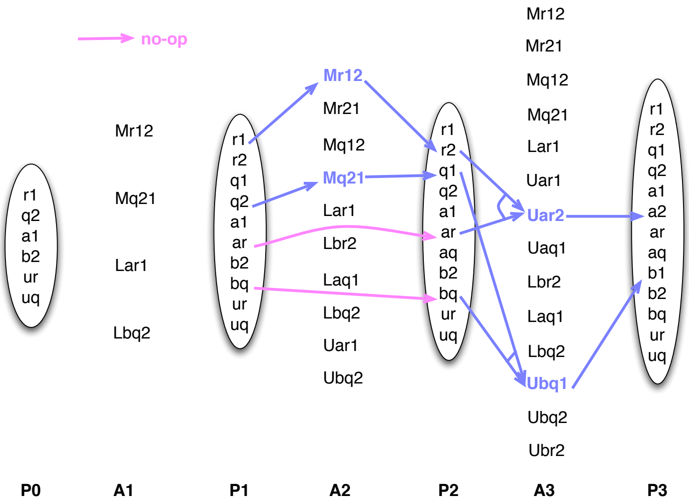
**Goal propositions found!**

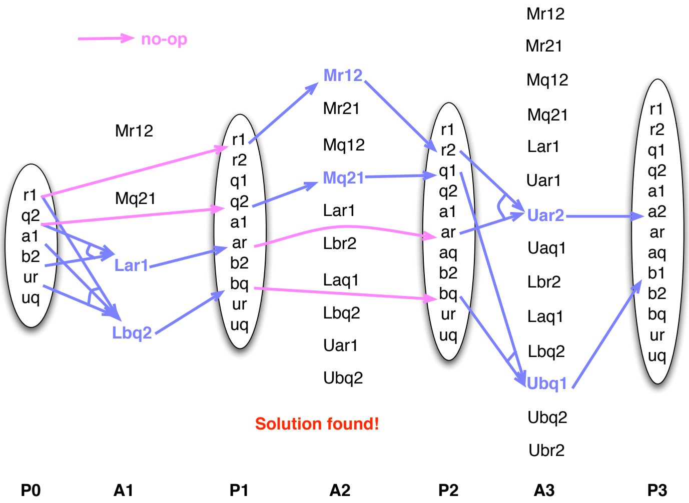
## Expand$([P_0, A_1, \mu A_1, P_1, \mu P_1, \ldots, A_{i-1}, \mu A_{i-1}, P_{i-1}, \mu P_{i-1}])$

```
1 $A_i \leftarrow \{a \in A \mid \text{precond}(a) \subseteq P_{i-1} \wedge \text{precond}^2(a) \cap \mu P_{i-1} = \emptyset\}$;
2 $P_i \leftarrow \{p \mid \exists a \in A_i,\ p \in \text{effects}^+(a)\}$;
3 $\mu A_i \leftarrow \{(a, b) \in A_i^2 \mid \text{effects}^-(a) \cap [\text{precond}(b) \cup \text{effects}^+(b)] \neq \emptyset$
  $\wedge\ \text{effects}^-(b) \cap [\text{precond}(a) \cup \text{effects}^+(b)] \neq \emptyset$
  $\wedge\ \exists (p, q) \in \mu P_{i-1},\ p \in \text{precond}(a) \wedge q \in \text{precond}(b)\}$;
4 foreach $a \in A_i$ do
5   link $a$ with preconditions arcs to $\text{precond}(a) \in P_{i-1}$;
6   link positive arcs to $\text{effects}^+(a)$ and negative arcs to $\text{effects}^-(a) \in P_i$;
7 return $[P_0, A_1, \mu A_1, P_1, \mu P_1, \ldots, A_i, \mu A_i, P_i, \mu P_i]$;
```

- The size of a planning graph down to level $k$ and the time required to expand it to that level are polynomial in the size of the planning problem.
- Every graph $G$ has a fixed-point level $\kappa$, which is the smallest $k$ such that $P_{k-1} = P_k$ and $\mu P_{k-1} = \mu P_k$.


no-op
r1
q2
a1
b2
ur
uq
Mr12
Mq21
Lar1
Lbq2
r1
r2
q1
q2
a1
ar
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Lbr2
Laq1
Lbq2
Uar1
Ubq2
r1
r2
q1
q2
a1
ar
aq
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Uar1
Uar2
Uaq1
Lbr2
Laq1
Lbq2
Ubq1
Ubq2
Ubr2
r1
r2
q1
q2
a1
a2
ar
aq
b1
b2
bq
ur
uq
P0
A1
P1
A2
P2
A3
P3


no-op
r1
q2
a1
b2
ur
uq
Mr12
Mq21
Lar1
Lbq2
r1
r2
q1
q2
a1
ar
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Lbr2
Laq1
Lbq2
Uar1
Ubq2
r1
r2
q1
q2
a1
ar
aq
b2
bq
ur
uq
Mr12
Mr21
Mq12
Mq21
Lar1
Uar1
Uar2
Uaq1
Lbr2
Laq1
Lbq2
Ubq1
Ubq2
Ubr2
r1
r2
q1
q2
a1
a2
ar
aq
b1
b2
bq
ur
uq
P0
A1
P1
A2
P2
A3
P3



Solution found!

- The search procedure looks for a set $\pi_i \in A_i$ of nonmutex actions that achieves all goal propositions.
- Preconditions of elements in $\pi_i$ become the new goal for level $i - 1$ and so on.
- A failure to meet the goal of some level $j$ leads to a backtrack over other subsets of $A_{j+1}$.
- If level 0 is successfully reached, then the corresponding sequence $[\pi_1, \ldots, \pi_i]$ is a solution plan.

- The extraction of a plan corresponds to a search in an AND/OR subgraph of the planning graph :
  - From a proposition in goal $g$, OR-branches are positive arcs from all actions in the preceding action level that support this proposition ;
  - From an action node, AND-branches are its preconditions arcs.

## Graphplan($\mathcal{O}, s_0, g$)

```
1  $i \leftarrow 0$ ; $\nabla \leftarrow \emptyset$ ; $P_0 \leftarrow s_0$ ; $G \leftarrow [P_0]$;
2  isSolvable $\leftarrow g \subseteq P_i \wedge g^2 \cap \mu P_i = \emptyset$ ; isExpandable $\leftarrow$ not (isSolvable $\vee$ isFixedPoint);
3  while IsExpandable do
4      i $\leftarrow$ i + 1;
5      G $\leftarrow$ Expand(G);
6      isExpandable $\leftarrow$ not (isSolvable $\vee$ isFixedPoint);
7  if isSolvable = false then $\perp$;
8  $\Pi \leftarrow$ Extract(G, g, i);
9  if isFixedPoint then
10     $\eta \leftarrow |\nabla(\kappa)|$
11 else
12     $\eta \leftarrow 0$
13 while $\Pi = \perp$ do
14     i $\leftarrow$ i + 1;
15     G $\leftarrow$ Expand(G);
16     $\Pi \leftarrow$ Extract(G, g, i);
17     if $\Pi = \perp \wedge$ isFixedPoint then
18         if $\eta = |\nabla(\kappa)|$ then return $\perp$
19     $\eta \leftarrow |\nabla(\kappa)|$;
20 return $\Pi$;
```

- The mutex relation between propositions provides only forbidden pairs, not tuples.
- The search may show that a tuple corresponding to an intermediate subgoal fails.
- Because of the backtracking and the iterative deepening, the search may have to analyze that same tuple more than once. Recording the tuples that failed may save time in future searches.
- This recording is performed by the Extract procedure into a nogood hash-table denoted $\nabla$.
- This hash table is indexed by the level of the failed goal because a goal $g$ can fail at level $i$ and succeed at $j > i$.

- Extract takes as input a planning graph $G$, a current set of goal propositions $g$ and a level $i$ ;
- It extracts a set of actions $\pi_i \subseteq A_i$ that achieves propositions of $g$ by recursively calling the GP-Search procedure ;
- If it succeeds in reaching level 0, then it returns an empty sequence, from which pending recursions successfully return a solution plan ;
- It records failed tuples into $\nabla$ table, and it chekcs each current goal with respect to recorded tuples ;
- a tuple $g$ is added to the nogood table at level $i$ only if the call to GP-Search fails to establish $g$ at this level from mutex and other nogoods found or established at the previous level.

## Extract$(G, g, i)$

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

## GP-search($G, g, \pi_i, i$)

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

- The GP-Search procedure selects each goal proposition $p$ at a time in some heuristic order ;
- Among the resolvers of $p$, it nondetermnistically chooses one action $a$ that tentatively extends the current subset $\pi_i$ through a recursive call at the same level ;
- This is performed on a subset of goals minus $p$ and minus all positive effects of $a$ in $g$ ;
- As usual, a failure for this nondeterministic choice is a backtrack point over other alternatives for achieving $p$, if any, or a backtracking further up if all resolvers of $p$ have been tried ;
- When $g$ is empty, then $\pi_i$ is complete ; the search recursively tries to extract a solution for level $i - 1$.

- Graphplan performs an initial graph expansion until either it reaches a level containing all goal propositions without mutex or it arrives at a fixed-point level ;
- If this fixed-point is achieved first, the goal is not achievable ;
- Otherwise, a search for a solution is performed. If no solution is found at this stage, the planning graph is iteratively expanded and explored ;
- This iterative deepening is pursued even **after** a fixed-point has been reached, until success or until the termination condition is met :
  - the number of nogood in $\nabla(\kappa)$ at the fixed-point level $\kappa$ stabilizes after two successive failures.

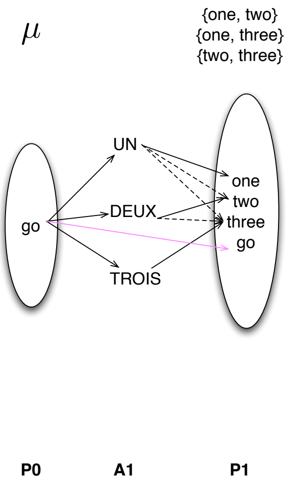
## Count operators

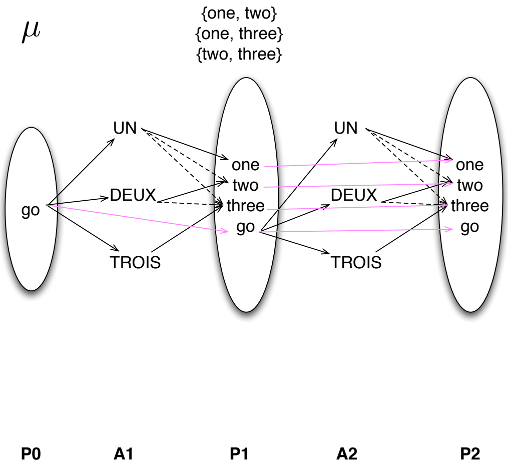
```
(operator UN
    (params )
    (preconds (go))
    (effects (one)(del two)(del three)))

(operator DEUX
    (params )
    (preconds (go))
    (effects (two)(del three)))

(operator TROIS
    (params )
    (preconds (go))
    (effects (three)))
```

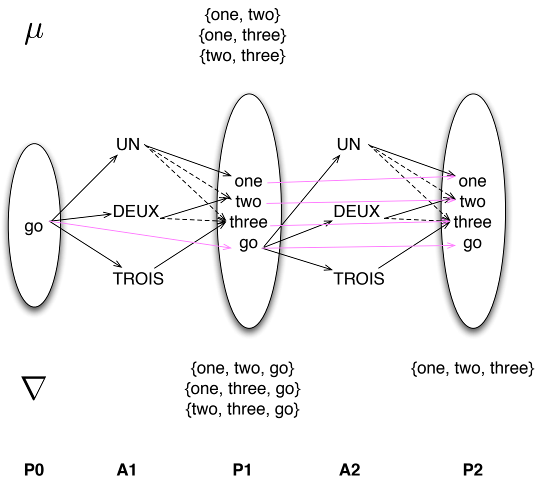
## Count Problem Definition

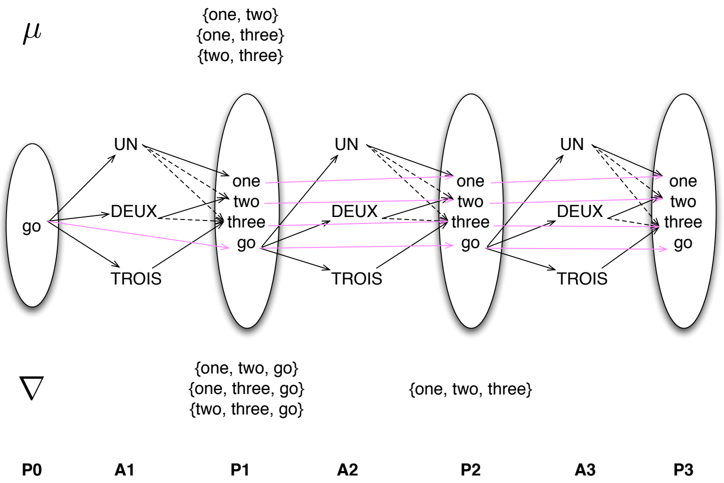
*(preconds*
*(go))*

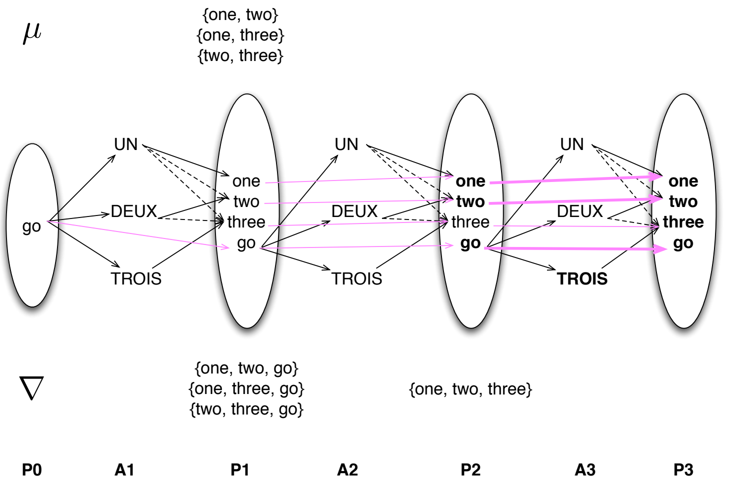
*(effect*
*(one)*
*(two)*
*(three))*

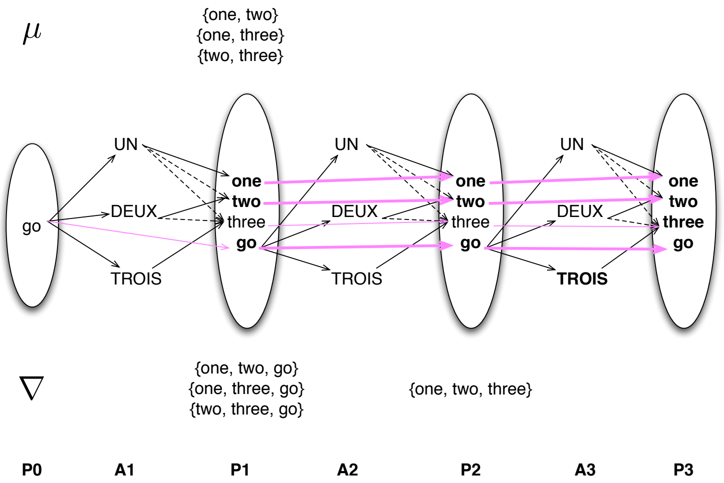

$\mu$
{one, two}
{one, three}
{two, three}
go
UN
DEUX
TROIS
one
two
three
go
P0
A1
P1

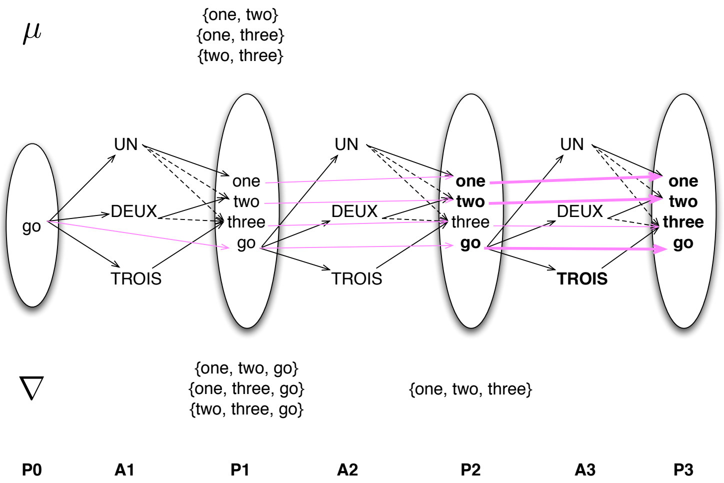

μ
{one, two}
{one, three}
{two, three}
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
P0
A1
P1
A2
P2

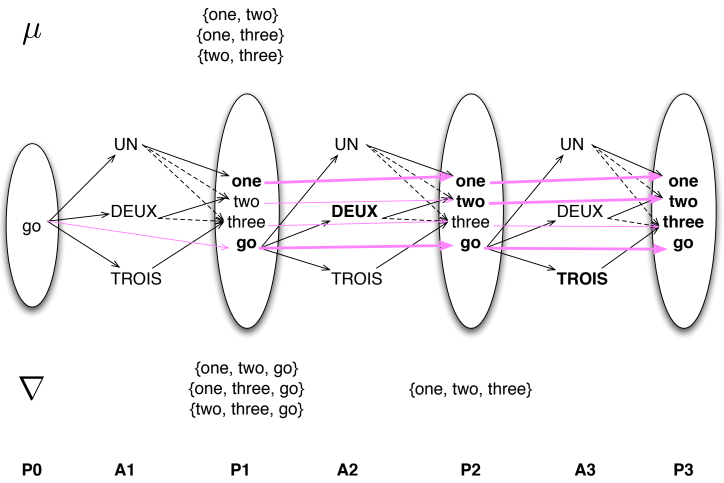

$\mu$
{one, two}
{one, three}
{two, three}
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
$\nabla$
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2

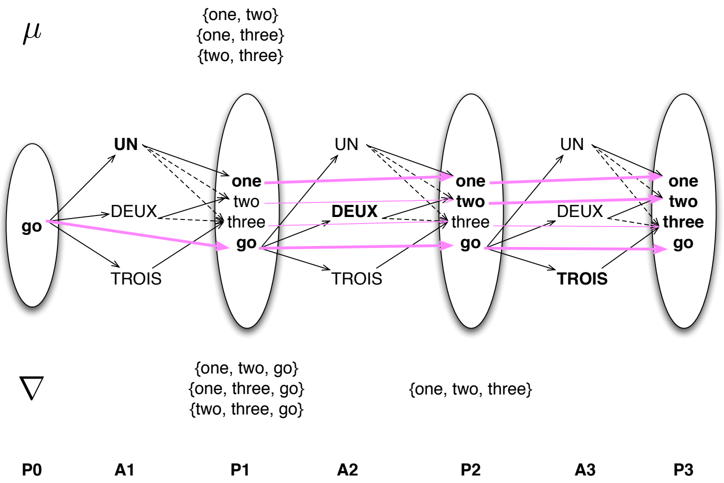

$\mu$
{one, two}
{one, three}
{two, three}
UN
DEUX
TROIS
go
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
$\nabla$
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3

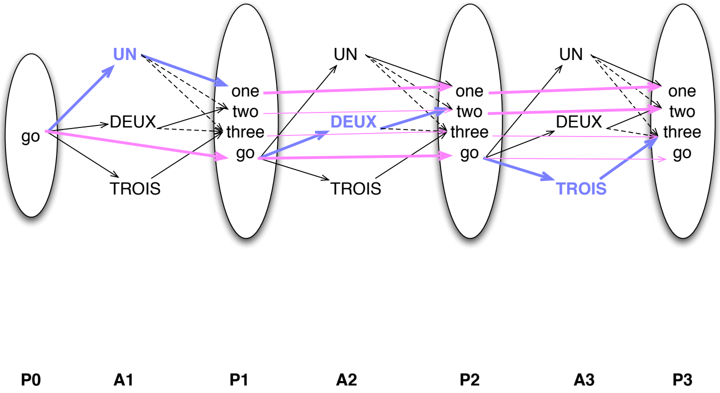

μ
{one, two}
{one, three}
{two, three}
UN
DEUX
TROIS
go
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
∇
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3


μ
{one, two}
{one, three}
{two, three}
UN
DEUX
TROIS
go
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
∇
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3


μ
{one, two}
{one, three}
{two, three}
UN
DEUX
TROIS
go
one
two
three
go
∇
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3


μ
{one, two}
{one, three}
{two, three}
UN
DEUX
TROIS
go
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
∇
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3


$\mu$
{one, two}
{one, three}
{two, three}
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
$\nabla$
{one, two, go}
{one, three, go}
{two, three, go}
{one, two, three}
P0
A1
P1
A2
P2
A3
P3


go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
UN
DEUX
TROIS
one
two
three
go
P0
A1
P1
A2
P2
A3
P3