

### Automatic Planning

Jörg Hoffmann

University of Innsbruck  
DERI

Masterstudium Informatik Vertiefungsmodul (WM 9)

- ▶ **Parallel STRIPS**
- ▶ Parallel Regression
- ▶  $k$ -Reachability
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ Implementing 2-Planning Graphs
- ▶ Language Extensions



### Interfering Actions

#### Definition

Let  $a_1 \neq a_2$  be STRIPS actions.  $a_1$  and  $a_2$  **interfere** if  $del(a_1) \cap (pre(a_2) \cup add(a_2)) \neq \emptyset$ , or  $del(a_2) \cap (pre(a_1) \cup add(a_1)) \neq \emptyset$ .

- ▶ Some interfering actions:
  - ▶ Drive(A,B) and Drive(A,C):  $\{at(A)\} \cap \{at(A), at(C)\}$
  - ▶ Watch(Football) and Watch(Vom-Winde-Verweht)
- ▶ Some **non-interfering** actions:
  - ▶ Drive(Truck1,A,B) and Drive(Truck2,A,C)
  - ▶ Load(Truck1,Package1,A) and Load(Truck1,Package2,A):  $\{at(Package1, A)\} \cap \{at(Truck1, A), at(Package2, A), in(Package2, Truck1)\}$



### Parallel STRIPS, Formalities I

#### Definition

Let  $P$  be a set of facts,  $s \subseteq P$  a world state, and  $A$  a set of pairwise non-interfering STRIPS actions. The **result**  $Result(s, \langle A \rangle)$  of applying  $A$  to  $s$  is

$$Result(s, \langle A \rangle) = \begin{cases} (s \cup \bigcup_{a \in A} add(a)) \setminus \bigcup_{a \in A} del(a) & \bigcup_{a \in A} pre(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$

In the first case,  $A$  is said to be **applicable** in  $s$ . The result of applying a sequence  $\langle A_1, \dots, A_n \rangle$  is defined by

$$result(s, \langle A_1, \dots, A_n \rangle) = result(result(s, \langle A_1, \dots, A_{n-1} \rangle), \langle A_n \rangle)$$

and  $Result(s, \langle \rangle) = s$ .

- ▶  $A$  is called a **meta-action**



**Definition**

Let  $(P, A, I, G)$  be a STRIPS task. A meta-action sequence  $\langle A_1, \dots, A_n \rangle$  is a **parallel plan** for the task iff  $G \subseteq \text{Result}(I, \langle A_1, \dots, A_n \rangle)$ . The plan is called **step-optimal** if there is no parallel plan with strictly less meta-actions in it.

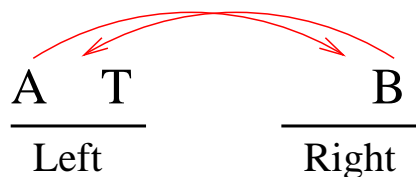
**Proposition**

Let  $(P, A, I, G)$  be a STRIPS task,  $s \subseteq P$  a state, and  $A_1 \subseteq A$  a set of pairwise non-interfering actions applicable in  $s$ . Let  $\langle a_1, \dots, a_n \rangle$  be an arbitrary ordering of the actions in  $A_1$ . Then  $\text{result}(s, \langle a_1, \dots, a_n \rangle)$  is defined, and equal to  $\text{Result}(s, \langle A_1 \rangle)$ .

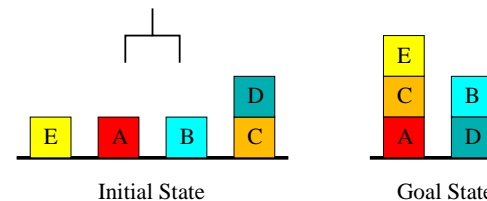
**Proof.**

Exercise. □

- ▶ Parallel STRIPS as opposed to **sequential** STRIPS
- ▶ The optimal parallel plan is as most as long as the optimal sequential plan: *why?*
- ▶ Most **optimal planning systems** are step-optimal
- ▶ Often, the optimal parallel plans are (much) shorter than the optimal sequential ones
- ▶ It is typically easier to prove optimality of shorter plans



- ▶ Optimal sequential plan:  $\langle \text{load}(A, T, \text{Left}), \text{drive}(T, \text{Left}, \text{Right}), \text{unload}(A, T, \text{Right}), \text{load}(B, T, \text{Right}), \text{drive}(T, \text{Right}, \text{Left}), \text{unload}(B, T, \text{Left}) \rangle$
- ▶ Optimal parallel plan: ???



- ▶  $\text{stack}(x, y), \text{unstack}(x, y), \text{pickup}(x), \text{putdown}(x)$
- ▶ This is a “sequential domain”: all pairs of actions either interfere, or are never applicable together in a reachable world state
- ▶ Example interfere: ???
- ▶ Example never applicable together: ???



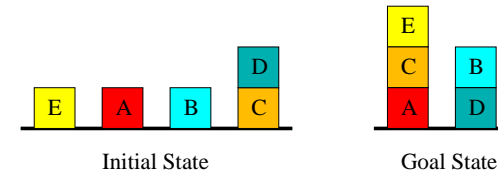
Whether or not there are any non-trivial meta-actions is a *property of the domain*.

Some **sequential domains**:

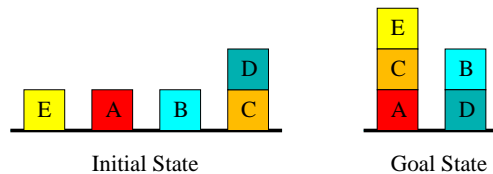
- ▶ Blocksworld, Towers-of-Hanoi, 8-Puzzle, Rubic's Cube, Miconic with a single elevator

Some **parallel domains**:

- ▶ Logistics, Freecell, Airport Ground Traffic Control, Miconic with several elevators,  
*Parallel Blocksworld*:



- ▶  $move(x, from, to) : (\{on(x, from), clear(x), clear(to)\}, \{on(x, to), clear(from)\}, \{on(x, from), clear(to)\})$
- ▶  $movetotable(x, from) : (\{on(x, from), clear(x)\}, \{on(x, table), clear(from)\}, \{on(x, from)\})$
- ▶  $movefromtable(x, to) : (\{on(x, table), clear(x), clear(to)\}, \{on(x, to)\}, \{on(x, table), clear(to)\})$
- ▶ No "Robot Hand", parallel moves possible!



- ▶ *Optimal sequential plan?*
- ▶ *Optimal parallel plan?*

- ▶ Parallel STRIPS
- ▶ **Parallel Regression**
- ▶ *k*-Reachability
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ Implementing 2-Planning Graphs
- ▶ Language Extensions

## Definition

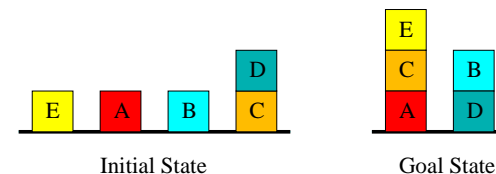
Let  $P$  be a set of facts,  $s \subseteq P$ , and  $A$  a set of pairwise non-interfering STRIPS actions. The **parallel regression**  $\text{Regress}(s, A)$  of  $s$  through  $A$  is

$$\text{Regress}(s, A) = (s \setminus \bigcup_{a \in A} \text{add}(a)) \cup \bigcup_{a \in A} \text{pre}(a)$$

given that

1.  $(\bigcup_{a \in A} \text{add}(a)) \cap s \neq \emptyset$ , and
2.  $(\bigcup_{a \in A} \text{del}(a)) \cap s = \emptyset$ ;

otherwise  $\text{Regress}(s, A)$  is undefined.



- ▶  $G = \{on(E, C), on(C, A), on(B, D)\}$
- ▶  $movefromtable(B, D) : (\{on(B, table), clear(B), clear(D)\}, \{on(B, D)\}, \{on(B, table), clear(D)\})$
- ▶  $regress(G, \{movefromtable(B, D), movefromtable(E, C)\}) = \{on(C, A), on(B, table), clear(B), clear(D), on(E, table), clear(E), clear(C)\}$



# Parallel Regression, Formalities II

## Definition

Let  $(P, A, I, G)$  be a STRIPS task. **Parallel regression** is the tuple  $(S, s_0, succ, Sol)$  where

1.  $S = 2^P$  is the set of all subsets of  $P$ ,
2.  $s_0 = G$ ,
3.  $succ : S \mapsto 2^S$  is defined by  $succ(s) = \{s' \in S \mid \exists A_1 \subseteq A : (\bigcup_{a \in A_1} \text{add}(a)) \cap s \neq \emptyset, (\bigcup_{a \in A_1} \text{del}(a)) \cap s = \emptyset, s' = \text{Regress}(s, A_1)\}$
4.  $Sol = \{s \in S \mid s \subseteq I\}$

- ▶ Can safely restrict to non-redundant  $A_1$ :  $\forall a \in A_1 : \text{add}(a) \cap s \neq \emptyset$ . Why?
- ▶ How can this observation be generalized?



# Automatic Planning

## 5. The “k-Subsets” Relaxation

- ▶ Parallel STRIPS
- ▶ Parallel Regression
- ▶ **k-Reachability**
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ Implementing 2-Planning Graphs
- ▶ Language Extensions



Let  $s$  be a set (conjunction) of facts.

- ▶ Denote by  $r(s)$  the solution distance of  $s$  in sequential regression
- ▶ Denote by  $R(s)$  the solution distance of  $s$  in parallel regression
- ▶ We call  $r(s)/R(s)$  the sequential/parallel **reachability function**
- ▶  $r(s)$  is the number of actions required to reach  $s$  from ???
- ▶  $R(s)$  is the number of ??? required to reach  $s$  from ???

The reachability functions are characterized by:

$$r(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{a \in A} r(\text{regress}(s, a)) & \text{otherwise} \end{cases}$$

$$R(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{A' \subseteq A} R(\text{Regress}(s, A')) & \text{otherwise} \end{cases}$$

- ▶ Computing reachability == solving these equations
- ▶ Solving a shortest-path problem in state space
- ▶ Doing a breadth-first forward search

**Assume that, to achieve a set (conjunction) of facts  $s$ , it suffices to achieve the hardest  $k$ -subset of  $s$ .**

$$r^k(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{a \in A} r^k(\text{regress}(s, a)) & |s| \leq k \\ \max_{s' \subseteq s, |s'|=k} r^k(s') & |s| > k \end{cases}$$

$$R^k(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{A' \subseteq A} R^k(\text{Regress}(s, A')) & |s| \leq k \\ \max_{s' \subseteq s, |s'|=k} R^k(s') & |s| > k \end{cases}$$

These are *lower bounds* on  $r$  and  $R$ ! *Exercise.*

- ▶ It suffices to compute  $r^k / R^k$  for the  $\leq k$ -sets of facts
- ▶ We relax a shortest-path problem in state space to a shortest-path problem in the space of  $\leq k$ -sets of facts
- ▶ Breadth-first forward search in that space
- ▶ There are in the order of  $|P|^k$  such sets
- ▶ Easy in  $|P|$ , hard in  $k$
- ▶ The higher  $k$ , the more accurate  $r^k / R^k$ ;  $k = |P|$ : exact functions
- ▶ Trade-off between accuracy and computational costs

- ▶ Parallel STRIPS
- ▶ Parallel Regression
- ▶  $k$ -Reachability
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ Implementing 2-Planning Graphs
- ▶ Language Extensions

$$r^1(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{a \in A, p \in \text{add}(a)} r^1(\text{pre}(a)) & s = \{p\} \\ \max_{p \in s} r^1(\{p\}) & |s| > 1 \end{cases}$$

$$R^1(s) = \begin{cases} 0 & s \subseteq I \\ 1 + \min_{a \in A, p \in \text{add}(a)} R^1(\text{pre}(a)) & s = \{p\} \\ \max_{p \in s} R^1(\{p\}) & |s| > 1 \end{cases}$$

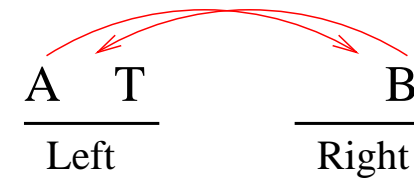
$r^1 == R^1$  (intuitively) because?



Forward computation of  $R^1(\{p\})$  for all  $p \in P$ :

```

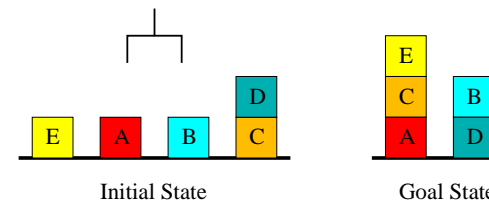
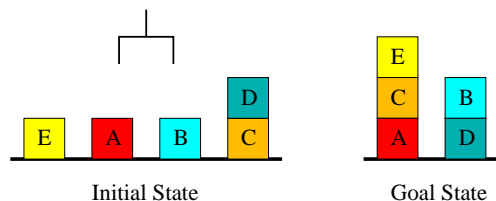
 $R^1(\{p\}) := 0$  for  $p \in I$ ,  $R^1(\{p\}) := \infty$  for  $p \notin I$ 
changes := TRUE, t := 0
while changes do
  changes := FALSE
  for all  $a \in A : R^1(\text{pre}(a)) = t$  do
    for all  $p \in \text{add}(a) : R^1(\{p\}) = \infty$  do
       $R^1(\{p\}) := t + 1$ , changes := TRUE
    endfor
  endfor
  t := t + 1
endwhile
    
```



Blackboard.

So  $r^1(G) = ?$  and  $R^1(G) = ?$





- 0.  $on(E, Table), clear(E), on(B, Table), clear(B), \dots, on(C, Table), on(D, C), clear(D), holding(NIL)$   
 $\dots pickup(E), pickup(B), unstack(D, C)$
- 1.  $holding(E), holding(B), clear(C), \dots$   
 $\dots stack(E, C), stack(B, D), pickup(C)$
- 2.  $on(E, C), on(B, D), holding(C), \dots$   
 $\dots stack(C, A)$
- 3.  $on(C, A), \dots$

- ▶  $G = \{on(E, C), on(C, A), on(B, D)\}$
- ▶  $r^1(G) = \max_{g \in G} r^1(\{g\})$
- ▶ So we have just seen that:  $r^1(G) = ?$
- ▶ But  $r(G) = ?$
- ▶ For parallel Blocksworld:  $R^1(G) = ?, R(G) = ?$



Some Supplementary Notations for  $k = 2$

Definition

Let  $p$  be a fact. By  $NOOP(p)$  we denote the STRIPS action that has  $p$  as its precondition, and whose only effect is  $p$ , i.e.,  $NOOP(p) = (\{p\}, \{p\}, \emptyset)$ . Given a set  $A$  of actions, and a set  $P$  of facts, by  $A^N$  we denote  $A \cup \{NOOP(p) \mid p \in P\}$ .

NOOPs are a shortcut for saying “leave this fact untouched”.

Definition

Let  $a$  and  $a'$  be STRIPS actions. We use  $a|a'$  to denote that  $a$  and  $a'$  are non-interfering.



(k = 2)-Reachability, Expanded Definition

$$R^2(s) = \begin{cases} 0 & s \subseteq I \\ 1+ \min_{a \in A, p \in add(a)} R^2(pre(a)) & s = \{p\} \\ 1+ \min_{a, a' \in A^N, a|a', \{p, q\} \subseteq add(a) \cup add(a')} R^2(pre(a) \cup pre(a')) & s = \{p, q\} \\ \max_{\{p, q\} \subseteq s} R^2(\{p, q\}) & |s| > 2 \end{cases}$$

In case 4, we allow  $p = q$ ; **actually this is unnecessary. Why?**  
 In case 3, we allow  $a = a'$ ; note that  $a$  or  $a'$  may be a NOOP.

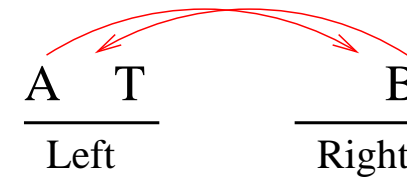
The expanded definition of  $r^2$  differs only by ???



```

R2({p}) := 0 for p ∈ I, R2({p}) := ∞ for p ∉ I
R2({p, q}) := 0 for {p, q} ⊆ I, R2({p, q}) := ∞ for {p, q} ⊄ I
changes := TRUE, t := 0
while changes do
  changes := FALSE
  for all a ∈ A : R2(pre(a)) = t do
    for all p ∈ add(a) : R2({p}) = ∞ do
      R2({p}) := t + 1, changes := TRUE
    endfor
    for all {p, q} ⊆ add(a) : R2({p, q}) = ∞ do
      R2({p, q}) := t + 1, changes := TRUE
    endfor
  endfor
  for all a, a' ∈ AN : a|a', R2(pre(a) ∪ pre(a')) = t do
    for all {p, q} ⊆ add(a) ∪ add(a') : R2({p, q}) = ∞ do
      R2({p, q}) := t + 1, changes := TRUE
    endfor
  endfor
  t := t + 1
endwhile

```



Without B: Blackboard.

How to modify this for r<sup>2</sup>?



With B:

- 0. {at(A, Left), at(T, Left)}  
load(A, T, Left), NOOP(at(T, Left))
- 1. {in(A, T), at(T, Left)}  
NOOP(in(A, T)), drive(T, Left, Right)
- 2. {in(A, T), at(T, Right)}  
unload(A, T, Right), NOOP(at(T, Right))
- 3. {at(A, Right), at(T, Right)}  
NOOP(at(A, Right)), drive(T, Right, Left)
- 4. {at(A, Right), at(T, Left), in(B, T)}  
NOOP(at(A, Right)), unload(B, T, Left)
- 5. {at(A, Right), at(B, Left)}

► For the 2-subsets of {at(A, Right), at(T, Left), in(B, T)}, R<sup>2</sup> and r<sup>2</sup> are exact respective to the corresponding real distances:

- 1. {at(A, Right), at(T, Left)}: as seen, R<sup>2</sup> = r<sup>2</sup> = ?
- 2. {at(T, Left), in(B, T)}: R<sup>2</sup> = r<sup>2</sup> = ?
- 3. {at(A, Right), in(B, T)}: R<sup>2</sup> = ?, r<sup>2</sup> = ?

⇒ in both cases, 2-subset nr. 1 is (one of the) hardest

- R<sup>2</sup>(G) = 5 is exact, for parallel STRIPS
- r<sup>2</sup>(G) = 5 is not, for sequential STRIPS!
- Remember that R<sup>1</sup>(G) = r<sup>1</sup>(G) = ?

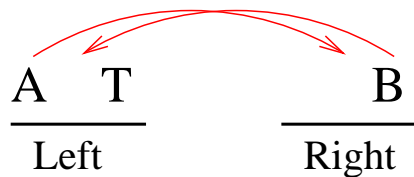


- ▶ Parallel STRIPS
- ▶ Parallel Regression
- ▶  $k$ -Reachability
- ▶  $k = 1, k = 2$
- ▶ **Planning Graphs**
- ▶ Implementing 2-Planning Graphs
- ▶ Language Extensions

Alternating fact- and action-layers  $F_0, A_0, F_1, A_1, \dots$

```

 $F_0 := I, t := 0$ 
while TRUE do
   $A_t := \{a \in A^N \mid pre(a) \subseteq F_t\}$ 
   $F_{t+1} := \bigcup_{a \in A_t} add(a)$ 
  if  $F_{t+1} = F_t$  then stop; endif
   $t := t + 1$ 
endwhile
    
```



Blackboard.

**Definition**

Let  $s$  be a set of facts, and  $F_0, \dots, F_m$  be the fact layers computed by a 1-Planning Graph. Then  $PG^1(s) := \min\{t \mid s \subseteq F_t\}$ , where the minimum over an empty set is  $\infty$ .

$PG^1(s)$  is the index of the first fact layer that contains  $s$ .

**Proposition**

Let  $(P, A, I, G)$  be a STRIPS task, and let  $s \subseteq P$ . Then  $R^1(s) = PG^1(s)$ .

**Proof.**

Blackboard. □

Does the same hold for  $r^1$ ?



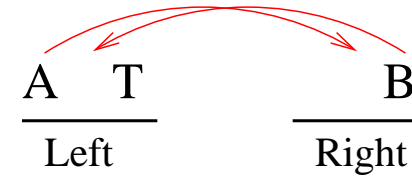
Alternating fact- and action-layers  $F_0, A_0, F_1, A_1, \dots$

With **mutex pairs**  $EF_0, EA_0, EF_1, EA_1, \dots$ ,

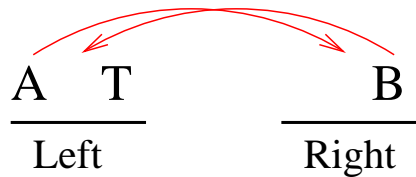
```

F0 := I, EF0 := ∅, t := 0
while TRUE do
  At := {a ∈ AN | pre(a) ⊆ Ft, ∄E ∈ EFt : E ⊆ pre(a)}
  EAt := {{a, a'} ⊆ At | a ≠ a' ∨ ∃E ∈ EFt : E ⊆ pre(a) ∪ pre(a')}
  Ft+1 := ∪a ∈ At add(a)
  EFt+1 := {{p, p'} ⊆ Ft+1 | ∃a, a' ∈ At : {p, p'} ⊆ add(a) ∪ add(a') →
    {a, a'} ∈ EAt}}
  if Ft+1 = Ft ∧ EFt+1 = EFt then stop; endif
  t := t + 1
endwhile
    
```

Note:  $1 < |E|$  for all  $E$ . Why?



Without B: Blackboard.



With B:

0.  $at(A, Left), at(B, Right), at(T, Left)$   
 $load(A, T, Left) \leftrightarrow drive(T, Left, Right)$
1.  $in(A, T) \leftrightarrow at(T, Right)$   
 $unload(A, T, Right), load(B, T, Right)$
2. ... and so on

Definition

Let  $s$  be a set of facts, and  $F_0, EF_0, \dots, F_m, EF_m$  be the fact and mutex layers computed by a 2-Planning Graph. Then  $PG^2(s) := \min\{t \mid s \subseteq F_t, \nexists E \in EF_t : E \subseteq s\}$ , where the minimum over an empty set is  $\infty$ .

$PG^2(s)$  is the index of the first fact layer that contains  $s$  without a mutex.

Proposition

Let  $(P, A, I, G)$  be a STRIPS task, and let  $s \subseteq P$ . Then  $R^2(s) = PG^2(s)$ .

Proof.

Blackboard. □

How to adapt the 2-planning graph to compute  $r^2$  instead?

```

F0 := I, EF0 := ∅, t := 0
while TRUE do
  At := {a ∈ AN | pre(a) ⊆ Ft, ∄E ∈ EFt : E ⊆ pre(a)}
  EAt := {E ⊆ At | |E| ≤ k, E minimal,
           ∄E' ∈ EFt : E' ⊆ ∪a∈E pre(a)}
  Ft+1 := ∪a∈At add(a)
  EFt+1 := {E ⊆ Pt+1 | |E| ≤ k, E minimal,
            ∄A' ⊆ At : E ⊆ ∪a∈A' add(a) →
            ∃E' ∈ EAt, E' ⊆ A'}
  if Ft+1 = Ft ∧ EFt+1 = EFt then stop; endif
  t := t + 1
endwhile

```

“Minimal”: no subset of  $E$  has the same property.

Note:  $1 < |E|$  for all  $E$ . Why?

k-Planning graphs are an alternative computation of k-reachability:

- ▶ Explicitly represent unreached tuples of facts
- ▶ Alternative: only remember reached tuples
- ▶ Explicitly represent all tuples of facts with same value
- ▶ Alternative: represent these fact tuples by the set containing all their members; mutexes say what possible tuples are *not* reached yet

- ▶ The mutex-based representation may or may not be beneficial (no mutex tuples/all tuples mutex)
- ▶ Typically, there are more reachable than unreachable tuples ... ???
- ▶ Fact layers increase monotonically over time – if  $p \in F_t$  then  $p \in F_{t+1}$ ; same for actions
- ▶ Mutexes decrease monotonically over time – if  $F_t \supseteq E \notin EF_t$  then  $E \notin EF_{t+1}$ ; same for actions
- ▶ Follows immediately from NOOPs

- ▶  $R^k/r^k$  depend on the initial state
- ▶ We want to use them as heuristic functions ...
- ▶ ... so, in forward search, the “initial state” changes!

*What can we do about this?*

- ▶ In practice, the effort to build planning graphs is dominated by the action mutexes
- ▶ (Without planning graphs: the action tuples possibly achieving the fact tuples)
- ▶ Thousands of actions,  $k = 2$  worst-case millions,  $k = 3 \dots$
- ▶ All implemented systems use  $k \leq 2$
- ▶ It is unclear if and how  $k > 2$  can be made effective
- ▶ Complicated domains like Rubik’s Cube or 15-Puzzle seem most likely

- ▶ Parallel STRIPS
- ▶ Parallel Regression
- ▶ k-Reachability
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ **Implementing 2-Planning Graphs**
- ▶ Language Extensions



Many of the relevant operations can be done in terms of bitvector operations:

- ▶ Assign UID to actions/facts (at layers)
- ▶ Create bitvectors  $PRE(a)$ ,  $ADD(a)$ ,  $DEL(a)$
- ▶ Create bitvectors  $F_t$ ,  $A_t$ ,  $MUTEX_t(a)$ ,  $ADD_t(p)$ ,  $MUTEX_t(p)$
- ▶ Bit-wise AND, OR, NOT operations:
  - ▶  $1010 \text{ AND } 0011 = ?$
  - ▶  $1010 \text{ OR } 0011 = ?$
  - ▶  $NOT(1010) = ?$
- ▶ Vector  $> 0$  :iff at least one bit is 1



- ▶ Can  $a$  be inserted into  $A_t$ ?

$$PRE(a) \text{ AND } NOT(F_t) == 0$$

“ $a$  has mutex preconditions at layer  $t$ ”?

- ▶ Are  $a$  and  $a'$  mutex at  $t$ ?

“ $a$  deletes an add or pre of  $a'$ ”?

“ $a'$  deletes an add or pre of  $a$ ”?

“ $a$  and  $a'$  have mutex preconditions at layer  $t$ ”: exercise.



- ▶  $F_0 = I$
- ▶  $F_{t+1} = OR_{a \in A_t} ADD(a)$
- ▶ “ $p$  and  $p'$  are mutex at layer  $t + 1$ ”: exercise.

#### Avoid Re-Computations and Double Storage:

- ▶ Keep only a single fact/action-layer, with a label indicating first appearance, and with dynamic structures storing time-dependent information, *namely?*
- ▶ Action interference needs to be computed at most once
- ▶ Don't re-generate action mutexes, propagate fact mutex changes to see if old mutexes disappear
- ▶ Don't re-generate fact mutexes, propagate action changes to see if old mutexes disappear

## Automatic Planning

### 5. The “ $k$ -Subsets” Relaxation

- ▶ Parallel STRIPS
- ▶ Parallel Regression
- ▶  $k$ -Reachability
- ▶  $k = 1, k = 2$
- ▶ Planning Graphs
- ▶ Implementing 2-Planning Graphs
- ▶ **Language Extensions**

## Some Historical Remarks

- ▶ Planning graphs were invented in 1995: “Graphplan” computes  $R^2$  via a 2-planning graph, then does an IDA\* (parallel) backward search
- ▶ At the time, the performance was spectacular
- ▶ Which lead to 1001 papers on “XY-Graphplan”
- ▶ We only outline the most important approaches (“important” is a very personalized notion here)
- ▶ The Graphplan interpretation/presentation here, in terms of heuristic search with  $k$ -reachability relaxation, was “invented” in [Haslum&Geffner, AIPS'00]; in the original Graphplan paper, the algorithms are not put into perspective

- ▶ STRIPS, with conditional effects, and 1st order formulas as (pre-, effect-, goal) conditions
- ▶ Compile the formulas away in a pre-process [Gazen&Knoblock, ECP'97] (e.g.,  $\forall x \in C : \phi(x)$  becomes  $\bigwedge_{x \in C} : \phi(x)$ )
- ▶ This is worst-case exponential but works Ok in practice, with a few optimizations [Koehler&Hoffmann, ECAI-ws'00]
- ▶ Compiling conditional effects away provably implies growth in plan length or exponential blow-up [Nebel, JAIR'00]
- ▶ We have to make explicit all possible *combinations* of effects
- ▶ Extending planning graphs to treat them directly is easy (IPP, [Koehler et al, ECP'97])

- ▶ STRIPS, actions annotated with duration, execution in parallel (i.e., overlapping)
- ▶ Extension of  $R^k$  definition/computation relatively straightforward: maximize over (estimated)  $k$ -subset achievement times
- ▶ Proposed by [Haslum&Geffner, ECP'01]
- ▶ An earlier approach extended 2-planning graphs: one separate graph layer for every time point at which “something happens”
- ▶ Proposed by [Smith&Weld, IJCAI'99]

- ▶ For restrictive language subset (only “resource consumption”),  $R^k$  definition/computation: maximize over (estimated)  $k$ -subset resource consumption
- ▶ Proposed by [Haslum & Geffner, ECP'01]
- ▶ For general language, keep “reachable intervals” alongside planning graph layers: update interval borders as new actions come in
- ▶ Proposed by [Koehler, ECAI'98]: “R-IPP”, Resource-InterferenceProgressionPlanner (“RIP-P”: RestInPeace-Planner)

ADL: no one has treated formulas directly, but that does not seem necessary

Temporal: the results *seem* at the best level one can achieve this way

Numeric:

- ▶ The algorithms work Ok only in very restrictive subsets; for more general subsets, they are either undefined or produce very bad (un-informative) heuristics
- ▶ Can we learn something from Model Checking?
- ▶ Can we make planning graphs that “distinguish between some of the possible numeric values”?

- ▶ P. Haslum and H. Geffner, *Admissible Heuristics for Optimal Planning*, Proceedings AIPS 2000
- ▶ A. Blum and M. Furst, *Fast planning through planning graph analysis*, Proceedings IJCAI 1995, AI, vol. 90, 1997
- ▶ D. Long and M. Fox, *The Efficient Implementation of the Plan Graph in STAN*, JAIR, vol. 10, 1999
- ▶ R. Brafman, *Reachability, Relevance, Resolution, and the Planning as Satisfiability Approach*, Proceedings IJCAI 1999, JAIR, vol. 14, 2001
- ▶ B. Gazen and C. Knoblock, *Combining the Expressiveness of UCPOP with the Efficiency of Graphplan*, Proceedings ECP 1997

- ▶ J. Koehler, J. Hoffmann, *On the Instantiation of ADL Operators Involving Arbitrary First-Order Formulas*, Proceedings “Puk Workshop” at ECAI 2000
- ▶ B. Nebel, *On the Compilability and Expressive Power of Propositional Planning Formalisms*, JAIR, vol. 12, 2000
- ▶ J. Koehler, B. Nebel, J. Hoffmann and Y. Dimopoulos, *Extending Planning Graphs to an ADL Subset*, Proceedings ECP 1997
- ▶ P. Haslum and H. Geffner, *Heuristic Planning with Time and Resources*, Proceedings ECP 2001
- ▶ D. Smith and D. Weld, *Temporal Planning with Mutual Exclusion Reasoning*, Proceedings IJCAI 1999
- ▶ J. Koehler, *Planning under Resource Constraints*, Proceedings ECAI 1998