

# Principles of AI Planning

November 8th, 2006 — Planning by state-space search

## Normal form for effects

STRIPS operators

## Planning by state-space search

Ideas  
Progression  
Regression  
Complexity  
Branching

# Principles of AI Planning

Planning by state-space search

Malte Helmert    Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

November 8th, 2006

Normal form

## Normal form for effects

1. Similarly to normal forms in propositional logic (DNF, CNF, NNF, ...) we can define a **normal form** for effects.
2. Nesting of conditionals, as in  $a \triangleright (b \triangleright c)$ , can be eliminated.
3. Effects  $e$  within a conditional effect  $\phi \triangleright e$  can be restricted to atomic effects ( $a$  or  $\neg a$ ).
4. Only a small polynomial increase in size by transformation to normal form.  
Compare: transformation to CNF or DNF may increase formula size exponentially.

Normal form

## Equivalences on effects

$$c \triangleright (e_1 \wedge \dots \wedge e_n) \equiv (c \triangleright e_1) \wedge \dots \wedge (c \triangleright e_n) \quad (1)$$

$$c_1 \triangleright (c_2 \triangleright e) \equiv (c_1 \wedge c_2) \triangleright e \quad (2)$$

$$(c_1 \triangleright e) \wedge (c_2 \triangleright e) \equiv (c_1 \vee c_2) \triangleright e \quad (3)$$

$$e \wedge (c \triangleright e) \equiv e \quad (4)$$

$$e \equiv \top \triangleright e \quad (5)$$

$$e \equiv \top \wedge e \quad (6)$$

$$e_1 \wedge e_2 \equiv e_2 \wedge e_1 \quad (7)$$

$$(e_1 \wedge e_2) \wedge e_3 \equiv e_1 \wedge (e_2 \wedge e_3) \quad (8)$$

## Normal form for operators and effects

### Definition

An operator  $\langle c, e \rangle$  is in **normal form** if for all occurrences of  $c' \supset e'$  in  $e$  the effect  $e'$  is either  $a$  or  $\neg a$  for some  $a \in A$ , and there is at most one occurrence of any atomic effect in  $e$ .

### Theorem

*For every operator there is an equivalent one in normal form.*

Proof is constructive: we can transform any operator into normal form by using the equivalences from the previous slide.

## Normal form for effects

### Example

### Example

$$(a \supset (b \wedge (c \supset (\neg d \wedge e)))) \wedge (\neg b \supset e)$$

transformed to normal form is

$$(a \supset b) \wedge ((a \wedge c) \supset \neg d) \wedge ((\neg b \vee (a \wedge c)) \supset e)$$

## STRIPS operators

### Definition

An operator  $\langle c, e \rangle$  is a **STRIPS operator** if

1.  $c$  is a conjunction of literals, and
2.  $e$  is a conjunction of atomic effects.

Hence every STRIPS operator is of the form

$$\langle l_1 \wedge \dots \wedge l_n, l'_1 \wedge \dots \wedge l'_m \rangle$$

where  $l_i$  are literals and  $l'_j$  are atomic effects.

### STRIPS

STanford Research Institute Planning System, Fikes & Nilsson, 1971.

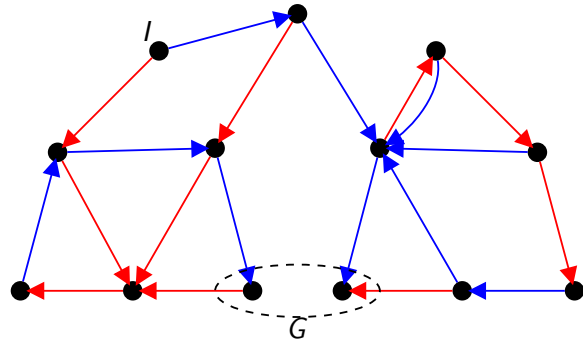
## Planning by state-space search

There are many alternative ways of doing planning by state-space search.

1. different ways of expressing planning as a search problem:
  - 1.1 **search direction**: forward, backward
  - 1.2 **representation** of search space: states, sets of states
2. different **search algorithms**: depth-first, breadth-first, informed (heuristic) search (**systematic**: A\*, IDA\*, ...; **local**: hill-climbing, simulated annealing, ...), ...
3. different ways of controlling search:
  - 3.1 **heuristics** for heuristic search algorithms
  - 3.2 **pruning techniques**: invariants, symmetry elimination, ...

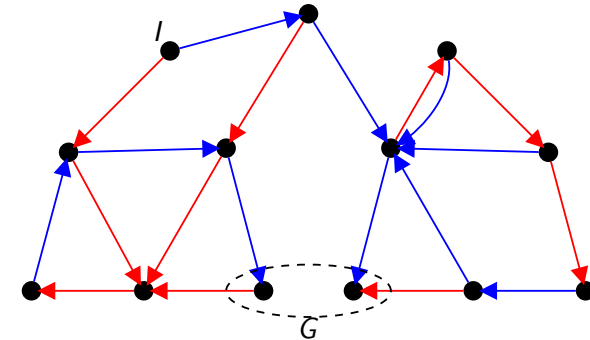
## Planning by forward search

with depth-first search



## Planning by backward search

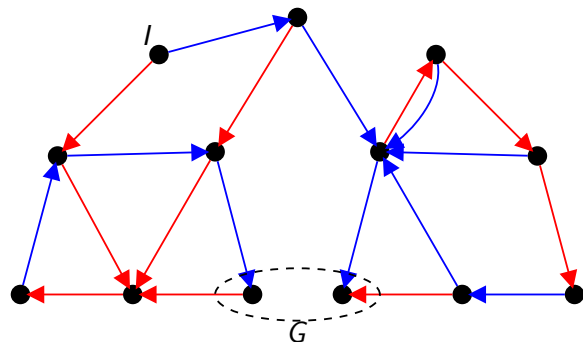
with depth-first search, one state at a time



## Planning by backward search

with depth-first search, for state sets (represented as formulae)

$$\begin{aligned}\phi_1 &= \text{regr}_{\rightarrow}(G) & \phi_3 &\xrightarrow{\text{red}} \phi_2 \xrightarrow{\text{blue}} \phi_1 \xrightarrow{\text{blue}} G \\ \phi_2 &= \text{regr}_{\rightarrow}(\phi_1) \\ \phi_3 &= \text{regr}_{\rightarrow}(\phi_2), I \models \phi_3\end{aligned}$$



## Progression

- **Progression** means computing the successor state  $app_o(s)$  of  $s$  with respect to  $o$ .
- Used in **forward search**: from the initial state toward the goal states.
- Very easy and efficient to implement.

## Regression

- **Regression** is computing the possible predecessor states of a set of states.
- The formula  $\text{regr}_o(\phi)$  represents the states from which a state represented by  $\phi$  is reached by operator  $o$ .
- Used in **backward search**: from the goal states toward the initial state.
- Regression is powerful because it allows handling sets of states (progression: only one state at a time.)
- Handling state sets (formulae) is more complicated than handling states: many questions about regression are **NP-hard**.

## Regression for STRIPS operators

- Regression for **STRIPS operators** is very simple.
- Goals are conjunctions of literals  $l_1 \wedge \dots \wedge l_n$ .
- **First step**: Choose an operator that makes some of  $l_1, \dots, l_n$  true and makes none of them false.
- **Second step**: Form a new goal by removing the fulfilled goal literals and adding the preconditions of the operator.

## Regression for STRIPS operators

### Definition

### Definition

The **STRIPS-regression**  $\text{regr}_o^{str}(\phi)$  of  $\phi = l_1'' \wedge \dots \wedge l_k''$  with respect to

$$o = \langle l_1 \wedge \dots \wedge l_n, \quad l_1' \wedge \dots \wedge l_m' \rangle$$

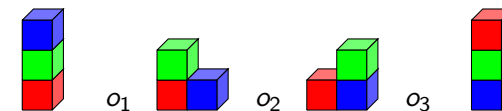
is the conjunction of literals

$$\bigwedge ((\{l_1'', \dots, l_k''\} \setminus \{l_1', \dots, l_m'\}) \cup \{l_1, \dots, l_n\})$$

provided that  $\{l_1', \dots, l_m'\} \cap \{\overline{l_1''}, \dots, \overline{l_k''}\} = \emptyset$ .

## Regression for STRIPS operators

### Example



**NOTE:** Predecessor states are in general not unique.  
This picture is just for illustration purposes.

$$o_1 = \langle \text{on} \text{green} \wedge \text{clr}, \neg \text{on} \text{green} \wedge \text{on} \text{red} \wedge \text{clr} \rangle$$

$$o_2 = \langle \text{on} \text{red} \wedge \text{clr} \wedge \text{on} \text{blue} \wedge \text{clr}, \neg \text{on} \text{red} \wedge \neg \text{on} \text{blue} \wedge \text{on} \text{red} \wedge \text{clr} \rangle$$

$$o_3 = \langle \text{on} \text{red} \wedge \text{clr} \wedge \text{on} \text{blue} \wedge \text{clr}, \neg \text{on} \text{red} \wedge \neg \text{on} \text{blue} \wedge \text{on} \text{red} \wedge \text{clr} \rangle$$

$$G = \text{on} \text{green} \wedge \text{on} \text{blue}$$

$$\phi_1 = \text{regr}_{o_3}^{str}(G) = \text{on} \text{blue} \wedge \text{on} \text{red} \wedge \text{clr} \wedge \text{clr}$$

$$\phi_2 = \text{regr}_{o_2}^{str}(\phi_1) = \text{on} \text{red} \wedge \text{on} \text{blue} \wedge \text{clr} \wedge \text{clr}$$

$$\phi_3 = \text{regr}_{o_1}^{str}(\phi_2) = \text{on} \text{red} \wedge \text{on} \text{blue} \wedge \text{clr} \wedge \text{clr}$$

## Regression for general operators

- ▶ With disjunction and conditional effects, things become more tricky. How to regress  $A \vee (B \wedge C)$  with respect to  $\langle Q, D \triangleright B \rangle$ ?
- ▶ The story about goals and subgoals and fulfilling subgoals, as in the STRIPS case, is no longer useful.
- ▶ We present a general method for doing regression for any formula and any operator.
- ▶ Now we extensively use the idea of *representing sets of states as formulae*.

Precondition for effect  $l$  to take place:  $EPC_l(e)$ 

Definition

## Definition

The condition  $EPC_l(e)$  for **literal  $l$  to become true under effect  $e$**  is defined as follows.

$$\begin{aligned}
 EPC_l(l) &= \top \\
 EPC_l(l') &= \perp \text{ when } l \neq l' \text{ (for literals } l') \\
 EPC_l(\top) &= \perp \\
 EPC_l(e_1 \wedge \dots \wedge e_n) &= EPC_l(e_1) \vee \dots \vee EPC_l(e_n) \\
 EPC_l(c \triangleright e) &= EPC_l(e) \wedge c
 \end{aligned}$$

Precondition for effect  $l$  to take place:  $EPC_l(e)$ 

Example

## Example

$$\begin{aligned}
 EPC_a(b \wedge c) &= \perp \vee \perp \equiv \perp \\
 EPC_a(a \wedge (b \triangleright a)) &= \top \vee (\top \wedge b) \equiv \top \\
 EPC_a((c \triangleright a) \wedge (b \triangleright a)) &= (\top \wedge c) \vee (\top \wedge b) \equiv c \vee b
 \end{aligned}$$

Precondition for effect  $l$  to take place:  $EPC_l(e)$ Connection to  $[e]_s$ 

## Lemma (A)

Let  $s$  be a state,  $l$  a literal and  $e$  an effect. Then  $l \in [e]_s$  if and only if  $s \models EPC_l(e)$ .

## Proof.

Induction on the structure of the effect  $e$ .

Base case 1,  $e = \top$ : By definition of  $[ \top ]_s$  we have  $l \notin [ \top ]_s = \emptyset$  and by definition of  $EPC_l(\top)$  we have  $s \not\models EPC_l(\top) = \perp$ : Both sides of the equivalence are false.

Base case 2,  $e = l$ :  $l \in [l]_s = \{l\}$  by definition, and  $s \models EPC_l(l) = \top$  by definition. Both sides are true.

Base case 3,  $e = l'$  for some literal  $l' \neq l$ :  $l \notin [l']_s = \{l'\}$  by definition, and  $s \not\models EPC_l(l') = \perp$  by definition. Both sides are false.

Precondition for effect  $l$  to take place:  $EPC_l(e)$ Connection to  $[e]_s$ 

proof continues...

Inductive case 1,  $e = e_1 \wedge \dots \wedge e_n$ : $l \in [e]_s$  iff  $l \in [e_1]_s \cup \dots \cup [e_n]_s$  (Def  $[e_1 \wedge \dots \wedge e_n]_s$ )iff  $l \in [e']_s$  for some  $e' \in \{e_1, \dots, e_n\}$ iff  $s \models EPC_l(e')$  for some  $e' \in \{e_1, \dots, e_n\}$  (IH)iff  $s \models EPC_l(e_1) \vee \dots \vee EPC_l(e_n)$ iff  $s \models EPC_l(e_1 \wedge \dots \wedge e_n)$ . (Def EPC)Inductive case 2,  $e = c \triangleright e'$ : $l \in [c \triangleright e']_s$  iff  $l \in [e']_s$  and  $s \models c$  (Def  $[c \triangleright e']_s$ )iff  $s \models EPC_l(e')$  and  $s \models c$  (IH)iff  $s \models EPC_l(e') \wedge c$ iff  $s \models EPC_l(c \triangleright e')$ . (Def EPC)

□

Precondition for effect  $l$  to take place:  $EPC_l(e)$ 

Connection to the normal form

## Remark

Notice that in terms of  $EPC_a(e)$  any operator  $\langle c, e \rangle$  can be expressed in normal form as

$$\left\langle c, \bigwedge_{a \in A} (EPC_a(e) \triangleright a) \wedge (EPC_{\neg a}(e) \triangleright \neg a) \right\rangle.$$

## Regression: definition for state variables

## Regressing a state variable

The formula  $EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$  expresses the value of  $a \in A$  after applying  $o$  in terms of values of state variables before applying  $o$ : Either

- ▶  $a$  became true, or
- ▶  $a$  was true before and it did not become false.

## Regression: definition for state variables

## Example

Let  $e = (b \triangleright a) \wedge (c \triangleright \neg a) \wedge b \wedge \neg d$ .

variable	$EPC_{\dots}(e) \vee (\dots \wedge \neg EPC_{\neg \dots}(e))$
$a$	$b \vee (a \wedge \neg c)$
$b$	$\top \vee (b \wedge \neg \perp) \equiv \top$
$c$	$\perp \vee (c \wedge \neg \perp) \equiv c$
$d$	$\perp \vee (d \wedge \neg \top) \equiv \perp$

## Regression: definition for state variables

## Lemma (B)

Let  $a$  be a state variable,  $o = \langle c, e \rangle \in O$  an operator,  $s$  a state and  $s' = \text{app}_o(s)$ . Then  $s \models \text{EPC}_a(e) \vee (a \wedge \neg \text{EPC}_{\neg a}(e))$  if and only if  $s' \models a$ .

## Proof.

First prove the implication from left to right.

Assume  $s \models \text{EPC}_a(e) \vee (a \wedge \neg \text{EPC}_{\neg a}(e))$ . Do a case analysis on the two disjuncts.

1. Assume that  $s \models \text{EPC}_a(e)$ . By Lemma A  $a \in [e]_s$  and hence  $s' \models a$ .
2. Assume that  $s \models a \wedge \neg \text{EPC}_{\neg a}(e)$ . By Lemma A  $\neg a \notin [e]_s$ . Hence  $a$  remains true in  $s'$ .

## Regression: definition for state variables

proof continues...

We showed that if the formula is **true** in  $s$ , then  $a$  is **true** in  $s'$ .

For the second part we show that if the formula is **false** in  $s$ , then  $a$  is **false** in  $s'$ .

1. So assume  $s \not\models \text{EPC}_a(e) \vee (a \wedge \neg \text{EPC}_{\neg a}(e))$ .
2. Hence  $s \models \neg \text{EPC}_a(e) \wedge (\neg a \vee \text{EPC}_{\neg a}(e))$  (de Morgan).
3. Analyze the two cases:  $a$  is true or it is false in  $s$ .
  - 3.1 Assume that  $s \models a$ . Now  $s \models \text{EPC}_{\neg a}(e)$  because  $s \models \neg a \vee \text{EPC}_{\neg a}(e)$ . Hence by Lemma A  $\neg a \in [e]_s$  and we get  $s' \not\models a$ .
  - 3.2 Assume that  $s \not\models a$ . Because  $s \models \neg \text{EPC}_a(e)$ , by Lemma A  $a \notin [e]_s$  and hence  $s' \not\models a$ .

Therefore in both cases  $s' \not\models a$ .

□

## Regression: general definition

We base the definition of regression on formulae  $\text{EPC}_l(e)$ .

## Definition

Let  $\phi$  be a propositional formula and  $o = \langle c, e \rangle$  an operator.

The **regression of  $\phi$  with respect to  $o$**  is

$$\text{regr}_o(\phi) = c \wedge \phi_r \wedge f$$

where

1.  $\phi_r$  is obtained from  $\phi$  by replacing each  $a \in A$  by  $\text{EPC}_a(e) \vee (a \wedge \neg \text{EPC}_{\neg a}(e))$ , and
2.  $f = \bigwedge_{a \in A} \neg(\text{EPC}_a(e) \wedge \text{EPC}_{\neg a}(e))$ .

The formula  $f$  says that no state variable may become simultaneously true and false.

## Regression: examples

1.  $\text{regr}_{\langle a, b \rangle}(b) \equiv a \wedge (\top \vee (b \wedge \neg \perp)) \wedge \top \equiv a$
2.  $\text{regr}_{\langle a, b \rangle}(b \wedge c \wedge d) \equiv a \wedge (\top \vee (b \wedge \neg \perp)) \wedge (\perp \vee (c \wedge \neg \perp)) \wedge (\perp \vee (d \wedge \neg \perp)) \wedge \top \equiv a \wedge c \wedge d$
3.  $\text{regr}_{\langle a, c \triangleright b \rangle}(b) \equiv a \wedge (c \vee (b \wedge \neg \perp)) \wedge \top \equiv a \wedge (c \vee b)$
4.  $\text{regr}_{\langle a, (c \triangleright b) \wedge (b \triangleright \neg b) \rangle}(b) \equiv a \wedge (c \vee (b \wedge \neg b)) \wedge \neg(c \wedge b) \equiv a \wedge c \wedge \neg b$
5.  $\text{regr}_{\langle a, (c \triangleright b) \wedge (d \triangleright \neg b) \rangle}(b) \equiv a \wedge (c \vee (b \wedge \neg d)) \wedge \neg(c \wedge d) \equiv a \wedge (c \vee b) \wedge (c \vee \neg d) \wedge (\neg c \vee \neg d)$

## Regression: examples

Blocks World with conditional effects

Operators to move blocks A and B onto the table from the other block if they are clear:

$$\begin{aligned} o_1 &= \langle \top, (AonB \wedge Aclear) \triangleright (AonT \wedge Bclear \wedge \neg AonB) \rangle \\ o_2 &= \langle \top, (BonA \wedge Bclear) \triangleright (BonT \wedge Aclear \wedge \neg BonA) \rangle \end{aligned}$$

Plan for putting both blocks onto the table **from any blocks world state** with two blocks is  $o_2, o_1$ . Proof by regression:

$$\begin{aligned} G &= AonT \wedge BonT \\ \phi_1 &= regr_{o_1}(G) \equiv ((AonB \wedge Aclear) \vee AonT) \wedge BonT \\ \phi_2 &= regr_{o_2}(\phi_1) \\ &\equiv ((AonB \wedge ((BonA \wedge Bclear) \vee Aclear)) \vee AonT) \\ &\quad \wedge ((BonA \wedge Bclear) \vee BonT) \end{aligned}$$

All three 2-block states satisfy  $\phi_2$ . Similar plans exist for any number of blocks.

## Regression: examples

Incrementing a binary number

$$\begin{aligned} &(\neg b_0 \triangleright b_0) \wedge \\ &((\neg b_1 \wedge b_0) \triangleright (b_1 \wedge \neg b_0)) \wedge \\ &((\neg b_2 \wedge b_1 \wedge b_0) \triangleright (b_2 \wedge \neg b_1 \wedge \neg b_0)) \end{aligned}$$

$$EPC_{b_2}(e) = \neg b_2 \wedge b_1 \wedge b_0$$

$$EPC_{b_1}(e) = \neg b_1 \wedge b_0$$

$$EPC_{b_0}(e) = \neg b_0$$

$$EPC_{\neg b_2}(e) = \perp$$

$$EPC_{\neg b_1}(e) = \neg b_2 \wedge b_1 \wedge b_0$$

$$EPC_{\neg b_0}(e) = (\neg b_1 \wedge b_0) \vee (\neg b_2 \wedge b_1 \wedge b_0) \equiv (\neg b_1 \vee \neg b_2) \wedge b_0$$

Regression replaces state variables as follows:

$$b_2 \text{ by } (\neg b_2 \wedge b_1 \wedge b_0) \vee (b_2 \wedge \neg \perp) \equiv (b_1 \wedge b_0) \vee b_2$$

$$b_1 \text{ by } (\neg b_1 \wedge b_0) \vee (b_1 \wedge \neg(\neg b_2 \wedge b_1 \wedge b_0))$$

$$\equiv (\neg b_1 \wedge b_0) \vee (b_1 \wedge (b_2 \vee \neg b_0))$$

$$b_0 \text{ by } \neg b_0 \vee (b_0 \wedge \neg(\neg b_1 \vee \neg b_2) \wedge b_0) \equiv \neg b_0 \vee (b_1 \wedge b_2)$$

## Regression: properties

## Lemma (C)

Let  $\phi$  be a formula,  $o$  an operator,  $s$  any state and  $s' = app_o(s)$ . Then  $s \models regr_o(\phi)$  if and only if  $s' \models \phi$ .

## Proof.

Let  $e$  be the effect of  $o$ . We show by structural induction over subformulae  $\phi'$  of  $\phi$  that  $s \models \phi'_r$  iff  $s' \models \phi'$ , where  $\phi'_r$  is  $\phi'$  with every  $a \in A$  replaced by  $EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$ . Rest of  $regr_o(\phi)$  just states that  $o$  is applicable in  $s$ .

**Induction hypothesis**  $s \models \phi'_r$  if and only if  $s' \models \phi'$ .

**Base cases 1 & 2**  $\phi' = \top$  or  $\phi' = \perp$ : Trivial as  $\phi'_r = \phi'$ .

**Base case 3**  $\phi' = a$  for some  $a \in A$ : Now  $\phi'_r = EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$ .  
By Lemma B  $s \models \phi'_r$  iff  $s' \models \phi'$ .

## Regression: properties

proof continues...

**Inductive case 1**  $\phi' = \neg\psi$ : By the induction hypothesis  $s \models \psi_r$  iff  $s' \models \psi$ . Hence  $s \models \phi'_r$  iff  $s' \models \phi'$  by the truth-definition of  $\neg$ .

**Inductive case 2**  $\phi' = \psi \vee \psi'$ : By the induction hypothesis  $s \models \psi_r$  iff  $s' \models \psi$ , and  $s \models \psi'_r$  iff  $s' \models \psi'$ . Hence  $s \models \phi'_r$  iff  $s' \models \phi'$  by the truth-definition of  $\vee$ .

**Inductive case 3**  $\phi' = \psi \wedge \psi'$ : By the induction hypothesis  $s \models \psi_r$  iff  $s' \models \psi$ , and  $s \models \psi'_r$  iff  $s' \models \psi'$ . Hence  $s \models \phi'_r$  iff  $s' \models \phi'$  by the truth-definition of  $\wedge$ .

□



## Regression: complexity issues

The following two tests are useful when generating a search tree with regression.

1. Testing that a formula  $\text{regr}_o(\phi)$  does not represent the empty set (= search is in a blind alley).  
For example,  $\text{regr}_{\langle a, \neg p \rangle}(p) \equiv a \wedge \perp \equiv \perp$ .
2. Testing that a regression step does not make the set of states smaller (= more difficult to reach).  
For example,  $\text{regr}_{\langle b, c \rangle}(a) \equiv a \wedge b$ .

Both of these problems are **NP-hard**.

## Regression: complexity issues

The formula  $\text{regr}_{o_1}(\text{regr}_{o_2}(\dots \text{regr}_{o_{n-1}}(\text{regr}_{o_n}(\phi))))$  may have size  $\mathcal{O}(|\phi| |o_1| |o_2| \dots |o_{n-1}| |o_n|)$ , i.e. the product of the sizes of  $\phi$  and the operators.

The size in the worst case  $\mathcal{O}(m^n)$  is hence exponential in  $n$ .

### Logical simplifications

1.  $\perp \wedge \phi \equiv \perp$ ,  $\top \wedge \phi \equiv \phi$ ,  $\perp \vee \phi \equiv \phi$ ,  $\top \vee \phi \equiv \top$
2.  $a \vee \phi \equiv a \vee \phi[\perp/a]$ ,  $\neg a \vee \phi \equiv \neg a \vee \phi[\top/a]$ ,  $a \wedge \phi \equiv a \wedge \phi[\top/a]$ ,  
 $\neg a \wedge \phi \equiv \neg a \wedge \phi[\perp/a]$

To obtain the maximum benefit from the last equivalences, e.g. for  $(a \wedge b) \wedge \phi(a)$ , the equivalences for associativity and commutativity are useful:  $(\phi_1 \vee \phi_2) \vee \phi_3 \equiv \phi_1 \vee (\phi_2 \vee \phi_3)$ ,  $\phi_1 \vee \phi_2 \equiv \phi_2 \vee \phi_1$ ,  
 $(\phi_1 \wedge \phi_2) \wedge \phi_3 \equiv \phi_1 \wedge (\phi_2 \wedge \phi_3)$ ,  $\phi_1 \wedge \phi_2 \equiv \phi_2 \wedge \phi_1$ .

## Regression: generation of search trees

**Problem** Formulae obtained with regression may become very big.

**Cause** **Disjunctivity** in the formulae. Formulae **without disjunctions** easily convertible to small formulae  $l_1 \wedge \dots \wedge l_n$  where  $l_i$  are literals and  $n$  is at most the number of state variables.

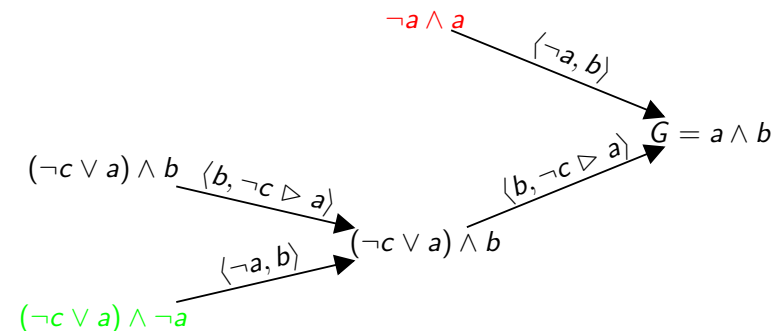
**Solution** Handle disjunctivity when generating search trees. Alternatives:

1. Do nothing. (May lead to very big formulae!!!)
2. Always eliminate all disjunctivity.
3. Reduce disjunctivity if formula becomes too big.

## Regression: generation of search trees

Unrestricted regression (= do nothing about formula size)

Reach goal  $a \wedge b$  from state  $I$  such that  $I \models \neg a \wedge \neg b \wedge \neg c$ .



## Regression: generation of search trees

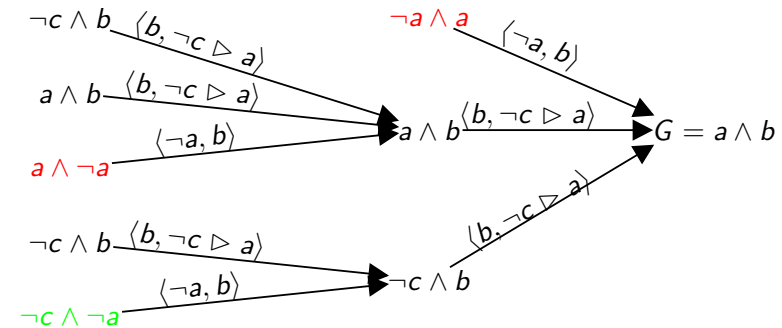
Full splitting (= eliminate all disjunctivity)

- Planners for STRIPS operators only need to use formulae  $l_1 \wedge \dots \wedge l_n$  where  $l_i$  are literals.
- Some PDDL planners also restrict to this class of formulae. This is done as follows.
  - $regr_o(\phi)$  is transformed to **disjunctive normal form (DNF)**:  $(l_1^1 \wedge \dots \wedge l_{n_1}^1) \vee \dots \vee (l_1^m \wedge \dots \wedge l_{n_m}^m)$ .
  - Each disjunct  $l_1^i \wedge \dots \wedge l_{n_i}^i$  is handled in its own subtree of the search tree.
  - The DNF formulae need not exist in its entirety explicitly: generate one disjunct at a time.
- Hence **branching** is both on the **choice of operator** and on the **choice of the disjunct** of the DNF formula.
- This leads to an **increased branching factor** and bigger search trees, but **avoids big formulae**.

## Regression: generation of search trees

Full splitting

Reach goal  $a \wedge b$  from state  $I$  such that  $I \models \neg a \wedge \neg b \wedge \neg c$ .  
 $(\neg c \vee a) \wedge b$  in DNF is  $(\neg c \wedge b) \vee (a \wedge b)$ .  
 It is split to  $\neg c \wedge b$  and  $a \wedge b$ .



## Regression: generation of search trees

Restricted splitting

- With **full splitting** search tree can be **exponentially bigger** than without splitting. (But it is not necessary to construct the DNF formulae explicitly!)
- Without **splitting** the formulae may have **size that is exponential** in the number of state variables.
- A compromise is to split formulae only when necessary: combine benefits of the two extremes.
- There are several ways to split a formula  $\phi$  to  $\phi_1, \dots, \phi_n$  such that  $\phi \equiv \phi_1 \vee \dots \vee \phi_n$ . For example:
  - Transform  $\phi$  to  $\phi_1 \vee \dots \vee \phi_n$  by equivalences like distributivity  $(\phi_1 \vee \phi_2) \wedge \phi_3 \equiv (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)$ .
  - Choose state variable  $a$ , set  $\phi_1 = a \wedge \phi$  and  $\phi_2 = \neg a \wedge \phi$ , and simplify with equivalences like  $a \wedge \psi \equiv a \wedge \psi[\top/a]$ .