Planning by state-space search (April 18, 2005)

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 $a \rhd (b \rhd c)$, can be eliminated.
3. Restriction to atomic effects e in conditional effects $\phi \rhd e$ can be made.
4. Only a small polynomial increase in size by transformation to normal form.
   Compare: transformation to CNF or DNF may increase formula size exponentially.

Equivalences on effects

$$
\begin{aligned}
c \rhd (e_1 \wedge \cdots \wedge e_n) &\equiv (c \rhd e_1) \wedge \cdots \wedge (c \rhd e_n) && (1)\\
c_1 \rhd (c_2 \rhd e) &\equiv (c_1 \wedge c_2) \rhd e && (2)\\
(c_1 \rhd e) \wedge (c_2 \rhd e) &\equiv (c_1 \vee c_2) \rhd e && (3)\\
e \wedge (c \rhd e) &\equiv e && (4)\\
e &\equiv \top \rhd e && (5)\\
e &\equiv \top \wedge e && (6)\\
e_1 \wedge e_2 &\equiv e_2 \wedge e_1 && (7)\\
(e_1 \wedge e_2) \wedge e_3 &\equiv e_1 \wedge (e_2 \wedge e_3) && (8)
\end{aligned}
$$

Normal form for operators and effects

Definition
An operator $\langle c, e\rangle$ is in normal form if for all occurrences of $c' \rhd e'$ 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

$$
\begin{aligned}
(a \;\rhd\; (b \wedge&\\
&(c \rhd (\neg d \wedge e)))) \wedge\\
(\neg b \;\rhd\; e)&
\end{aligned}
$$

transformed to normal form is

$$
\begin{aligned}
(a &\;\rhd\; b) \wedge\\
((a \wedge c) &\;\rhd\; \neg d) \wedge\\
((\neg b \vee (a \wedge c)) &\;\rhd\; e)
\end{aligned}
$$

STRIPS operators

Definition
An operator $\langle c, e\rangle$ is a STRIPS operator if
1. c is a conjunction of literals, and
2. e does not contain $\rhd$.

Hence every STRIPS operator is of the form

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

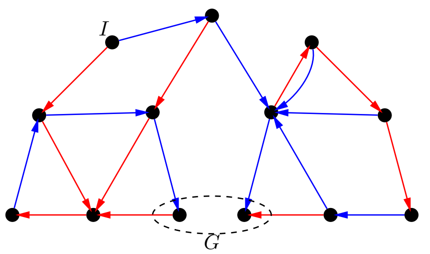
where l_i are literals and l'_j are atomic effects.

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



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 $regr_o(\phi)$ represents the states from which a state represented by ϕ is reached by operator o.
- Used in backward search: from the goal states toward the initial states.
- Regression is powerful because it allows handling sets of states (progression: only one state at a time.)
- Handling 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 $l_1 \wedge \cdots \wedge l_n$.
- First step: Choose an operator that makes some of $l_1, \ldots, l_n$ 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 $regr_o^{str}(\phi)$ of $\phi = l''_1 \wedge \cdots \wedge l''_{m'}$ with respect to

$$o = \langle l_1 \wedge \cdots \wedge l_n,\ \ l'_1 \wedge \cdots \wedge l'_m \rangle$$

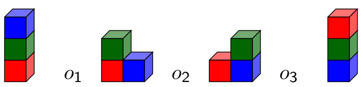
is the conjunction of literals

$$\bigwedge ((\{l''_1, \ldots, l''_{m'}\} \backslash \{l'_1, \ldots, l'_m\}) \cup \{l_1, \cdots, l_n\})$$

provided that $\{l', \ldots, l'_m\} \cap \{\overline{l''_1}, \ldots, \overline{l''_{m'}}\} = \emptyset$.

Regression for STRIPS operators

Example



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

$$\begin{aligned}
o_1 &= \langle {\color{blue}\blacksquare}\text{on}{\color{green}\blacksquare} \wedge {\color{blue}\blacksquare}\text{clr},\ \neg{\color{blue}\blacksquare}\text{on}{\color{green}\blacksquare} \wedge {\color{blue}\blacksquare}\text{onT} \wedge {\color{green}\blacksquare}\text{clr}\rangle \\
o_2 &= \langle {\color{green}\blacksquare}\text{on}{\color{red}\blacksquare} \wedge {\color{green}\blacksquare}\text{clr} \wedge {\color{blue}\blacksquare}\text{clr},\ \neg{\color{blue}\blacksquare}\text{clr} \wedge \neg{\color{green}\blacksquare}\text{on}{\color{red}\blacksquare} \wedge {\color{green}\blacksquare}\text{on}{\color{blue}\blacksquare} \wedge {\color{red}\blacksquare}\text{clr}\rangle \\
o_3 &= \langle {\color{red}\blacksquare}\text{onT} \wedge {\color{red}\blacksquare}\text{clr} \wedge {\color{green}\blacksquare}\text{clr},\ \neg{\color{green}\blacksquare}\text{clr} \wedge \neg{\color{red}\blacksquare}\text{onT} \wedge {\color{red}\blacksquare}\text{on}{\color{green}\blacksquare}\rangle
\end{aligned}$$

$$\begin{aligned}
G &= {\color{red}\blacksquare}\text{on}{\color{green}\blacksquare} \wedge {\color{green}\blacksquare}\text{on}{\color{blue}\blacksquare} \\
\phi_1 &= regr_{o_3}^{str}(G) = {\color{green}\blacksquare}\text{on}{\color{blue}\blacksquare} \wedge {\color{red}\blacksquare}\text{onT} \wedge {\color{red}\blacksquare}\text{clr} \wedge {\color{green}\blacksquare}\text{clr} \\
\phi_2 &= regr_{o_2}^{str}(\phi_1) = {\color{red}\blacksquare}\text{onT} \wedge {\color{green}\blacksquare}\text{clr} \wedge {\color{green}\blacksquare}\text{on}{\color{red}\blacksquare} \wedge {\color{blue}\blacksquare}\text{clr} \\
\phi_3 &= regr_{o_1}^{str}(\phi_2) = {\color{red}\blacksquare}\text{onT} \wedge {\color{green}\blacksquare}\text{on}{\color{red}\blacksquare} \wedge {\color{blue}\blacksquare}\text{clr} \wedge {\color{blue}\blacksquare}\text{on}{\color{green}\blacksquare}
\end{aligned}$$

Regression for general operators

- With disjunction and conditional effects, things become more tricky. How to regress $A \vee (B \wedge C)$ with respect to $\langle Q, D \rhd B \rangle$?
- 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.

$$\begin{aligned}
EPC_l(l) &= \top \\
EPC_l(l') &= \bot \text{ when } l \neq l' \quad \text{(for literals } l'\text{)} \\
EPC_l(\top) &= \bot \\
EPC_l(e_1 \wedge \cdots \wedge e_n) &= EPC_l(e_1) \vee \cdots \vee EPC_l(e_n) \\
EPC_l(c \rhd e) &= EPC_l(e) \wedge c
\end{aligned}$$

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

Example

Example

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

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

Connection to $[e]_s$

Lemma (B)

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 $s \not\models EPC_l(\top) = \bot$: 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 $s \not\models EPC_l(l') = \bot$ 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, $e = e_1 \wedge \cdots \wedge e_n$:

$l \in [e]_s$ iff $l \in [e_1]_s \cup \cdots \cup [e_n]_s$ (Def $[e_1 \wedge \cdots \wedge e_n]_s$)
iff $l \in [e']_s$ for some $e' \in \{e_1, \ldots, e_n\}$
iff $s \models EPC_l(e')$ for some $e' \in \{e_1, \ldots, e_n\}$ (IH)
iff $s \models EPC_l(e_1) \vee \cdots \vee EPC_l(e_n)$
iff $s \models EPC_l(e_1 \wedge \cdots \wedge e_n)$. (Def EPC)

Inductive case 2, $e = c \rhd e'$:

$l \in [c \rhd e']_s$ iff $l \in [e']_s$ and $s \models c$ (Def $[c \rhd e']_s$)
iff $s \models EPC_l(e')$ and $s \models c$ (IH)
iff $s \models EPC_l(e') \wedge c$
iff $s \models EPC_l(c \rhd e')$. (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

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

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

Regression: definition for state variables

Example

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

$variable$	$EPC_{\ldots}(e) \vee (\cdots \wedge \neg EPC_{\neg\ldots}(e))$
a	$b \vee (a \wedge \neg c)$
b	$\top \vee (b \wedge \neg\bot) \equiv \top$
c	$\bot \vee (c \wedge \neg\bot) \equiv c$
d	$\bot \vee (d \wedge \neg\top) \equiv \bot$

Regression: definition for state variables

Lemma (C)

Let a be a state variable, $o = \langle c, e\rangle \in O$ an operator, s a state and $s' = app_o(s)$. Then $s \models EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$ if and only if $s' \models a$.

Proof.

First prove the implication from left to right.

Assume $s \models EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$. Do a case analysis on the two disjuncts.

1. Assume that $s \models EPC_a(e)$. By Lemma B $a \in [e]_s$ and hence $s' \models a$.
2. Assume that $s \models a \wedge \neg EPC_{\neg a}(e)$. By Lemma B $\neg a \notin [e]_s$. Hence a remains true in s'.

Regression: definition for state variables

proof continues...

In the first part we showed that if the formula is true in s, then a is true in s'.

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

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

Definition

Let ϕ be a propositional formula and $o = \langle c, e\rangle$ an operator.
The regression of ϕ with respect to o is

$$regr_o(\phi) = \phi_r \wedge c \wedge f$$

where

1. ϕ_r is obtained from ϕ by replacing each $a \in A$ by $EPC_a(e) \vee (a \wedge \neg EPC_{\neg a}(e))$, and
2. $f = \bigwedge_{a \in A} \neg(EPC_a(e) \wedge EPC_{\neg a}(e))$.

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

Regression: examples

1. $\mathit{regr}_{\langle a,b\rangle}(b) = (a \wedge (\top \vee (b \wedge \neg\bot))) \equiv a$
2. $\mathit{regr}_{\langle a,b\rangle}(b \wedge c \wedge d) =$
   $(a \wedge (\top \vee (b \wedge \neg\bot)) \wedge (\vee\bot(c \wedge \neg\bot)) \wedge (\bot \vee (d \wedge \neg\bot))) \equiv a \wedge c \wedge d$
3. $\mathit{regr}_{\langle a,c\rhd b\rangle}(b) = (a \wedge (c \vee (b \wedge \neg\bot))) \equiv a \wedge (c \vee b)$
4. $\mathit{regr}_{\langle a,(c\rhd b)\wedge(b\rhd\neg b)\rangle}(b) = (a \wedge (c \vee (b \wedge \neg b)) \wedge \neg(c \wedge b)) \equiv a \wedge c \wedge \neg b$
5. $\mathit{regr}_{\langle a,(c\rhd b)\wedge(d\rhd\neg b)\rangle}(b) = (a \wedge (c \vee (b \wedge \neg d)) \wedge \neg(c \wedge d)) \equiv$
   $a \wedge (c \vee b) \wedge (c \vee \neg d) \wedge (\neg c \vee \neg d)$

Regression: examples

Blocks World with conditional effects

Moving blocks A and B onto the table from any location if they are clear.

$$o_1 = \langle \top, (\mathit{AonB} \wedge \mathit{Aclear}) \rhd (\mathit{AonT} \wedge \mathit{Bclear} \wedge \neg\mathit{AonB})\rangle$$
$$o_2 = \langle \top, (\mathit{BonA} \wedge \mathit{Bclear}) \rhd (\mathit{BonT} \wedge \mathit{Aclear} \wedge \neg\mathit{BonA})\rangle$$

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

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

All three 2-block states satisfy ϕ_2. Similar plans exist for any number of blocks.

Regression: examples

Incrementing a binary number

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

$$\begin{aligned} \mathit{EPC}_{b_2}(e) &= \neg b_2 \wedge b_1 \wedge b_0 & \mathit{EPC}_{\neg b_2}(e) &= \bot \\ \mathit{EPC}_{b_1}(e) &= \neg b_1 \wedge b_0 & \mathit{EPC}_{\neg b_1}(e) &= \neg b_2 \wedge b_1 \wedge b_0 \\ \mathit{EPC}_{b_0}(e) &= \neg b_0 & \mathit{EPC}_{\neg b_0}(e) &= (\neg b_1 \wedge b_0) \vee (\neg b_2 \wedge b_1 \wedge b_0) \\ & & &\equiv (\neg b_1 \vee \neg b_2) \wedge b_0 \end{aligned}$$

Regression replaces state variables as follows.

$$\begin{aligned} b_2 \text{ by } & (b_2 \wedge \neg\bot) \vee (\neg b_2 \wedge b_1 \wedge b_0) \equiv b_2 \vee (b_1 \wedge b_0) \\ b_1 \text{ by } & (b_1 \wedge \neg(\neg b_2 \wedge b_1 \wedge b_0)) \vee (\neg b_1 \wedge b_0) \\ & \equiv (b_1 \wedge (b_2 \vee \neg b_0)) \vee (\neg b_1 \wedge b_0) \\ b_0 \text{ by } & (b_0 \wedge \neg((\neg b_1 \vee \neg b_2) \wedge b_0)) \vee \neg b_0 \equiv (b_1 \wedge b_2) \vee \neg b_0 \end{aligned}$$

Regression: properties

Lemma (D)

Let ϕ be a formula, o an operator, s any state and $s' = \mathit{app}_o(s)$. Then $s \models \mathit{regr}_o(\phi)$ if and only if $s' \models \phi$.

Proof.

Let e be the effect of o. We show by structural induction over subformulae ϕ' of ϕ that $s \models \phi'_r$ iff $s' \models \phi'$, where ϕ'_r is ϕ' with every $a \in A$ replaced by $\mathit{EPC}_a(e) \vee (a \wedge \neg\mathit{EPC}_{\neg a}(e))$. Rest of $\mathit{regr}_o(\phi)$ 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 $\phi' = \bot$: Trivial as $\phi'_r = \phi'$.

Base case 3 $\phi' = a$ for some $a \in A$: Now $\phi'_r = \mathit{EPC}_a(e) \vee (a \wedge \neg\mathit{EPC}_{\neg a}(e))$. By Lemma C $s \models \phi'_r$ 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 $\mathit{regr}_o(\phi)$ does not represent the empty set (= search is in a blind alley).
   For example, $\mathit{regr}_{\langle a,\neg p\rangle}(p) = a \wedge \bot \equiv \bot$.
2. Testing that a regression step does not make the set of states smaller (= more difficult to reach).
   For example, $\mathit{regr}_{\langle b,c\rangle}(a) = a \wedge b$.

Both of these problems are NP-hard.

Regression: complexity issues

The formula $\mathit{regr}_{o_1}(\mathit{regr}_{o_2}(\cdots \mathit{regr}_{o_{n-1}}(\mathit{regr}_{o_n}(\phi))))$ may have size $\mathcal{O}(|\phi||o_1||o_2|\cdots|o_{n-1}||o_n|)$, i.e. the product of the sizes of ϕ and the operators.
The size in the worst case $\mathcal{O}(2^n)$ is hence exponential in n.

Logical simplifications

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

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 $l_1 \wedge \cdots \wedge l_n$ where l_i are literals and n is at most the number of state variables.

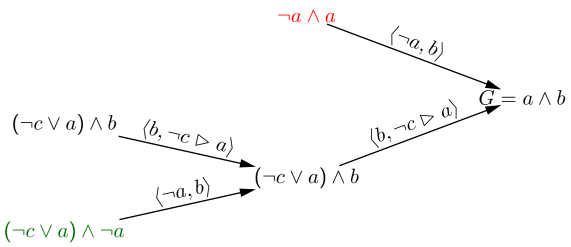
Solution Handle disjunctivity when generating search trees. Alternatives:

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

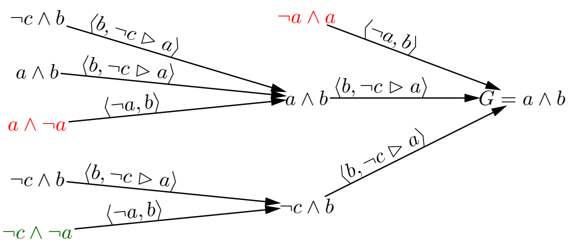
Full splitting (= eliminate all disjunctivity)

- Planners for STRIPS operators only need to use formulae $l_1 \wedge \cdots \wedge l_n$ where l_i are literals.
- Some PDDL planners also restrict to this class of formulae. This is done as follows.
  1. *regr*$_o(\phi)$ is transformed to disjunctive normal form (DNF): $(l_1^1 \wedge \cdots \wedge l_{n_1}^1) \vee \cdots \vee (l_1^n \wedge \cdots \wedge l_{n_n}^n)$.
  2. Each disjunct $l_1^i \wedge \cdots \wedge i_{n_1}^i$ is handled in its own subtree of the search tree.
  3. 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 ϕ to $\phi_1, \ldots, \phi_n$ such that $\phi \equiv \phi_1 \vee \cdots \vee \phi_n$. For example:
  1. Transform ϕ to $\phi_1 \vee \cdots \vee \phi_n$ by equivalences like distributivity $(\phi_1 \vee \phi_2) \wedge \phi_3 \equiv (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)$.
  2. Choose state variable a, set $\phi_1 = a \wedge \phi$ and $\phi_2 = \neg a \wedge \phi$, and simplify with equivalences like $a \wedge \psi \equiv a \wedge \psi[\top/a]$.