Nondeterministic effects: restriction

Let $p \in P$ be a state variable.

Let $e_1 \wedge \cdots \wedge e_n$ be an effect. If $e_1, \ldots, e_n$ are not all deterministic, then p or $\neg p$ may occur as an atomic effect in at most one of $e_1, \ldots, e_n$.

EXAMPLE: $(0.5(a \rhd b)|0.5c) \wedge (0.5(\neg a \rhd \neg b)|0.5d)$ is not allowed.

Translation of ND effects into PL

The set of state variables that are possibly changed:

$$\begin{aligned}
\textit{changes}(a) &= \{a\} \\
\textit{changes}(\neg a) &= \{a\} \\
\textit{changes}(c \rhd e) &= \textit{changes}(e) \\
\textit{changes}(e_1 \wedge \cdots \wedge e_n) &= \textit{changes}(e_1) \cup \cdots \cup \textit{changes}(e_n) \\
\textit{changes}(p_1 e_1 | \cdots | p_n e_n) &= \textit{changes}(e_1) \cup \cdots \cup \textit{changes}(e_n)
\end{aligned}$$

Translation of ND effects into PL

For effects e in normal form I and sets B of state variables define

$$\begin{aligned}
\mathrm{PL}_B(e) &= \text{when } e \text{ is deterministic} \\
&\quad\ \text{translated like in Lecture 6, but restricted} \\
&\quad\ \text{to state variables in the set } B \\
\mathrm{PL}_B(p_1 e_1 | \cdots | p_n e_n) &= \mathrm{PL}_B(e_1) \vee \cdots \vee \mathrm{PL}_B(e_n) \\
\mathrm{PL}_B(e_1 \wedge \cdots \wedge e_n) &= \mathrm{PL}_{B \backslash (B_2 \cup \cdots \cup B_n)}(e_1) \wedge \mathrm{PL}_{B_2}(e_2) \wedge \cdots \wedge \mathrm{PL}_{B_n}(e \\
&\quad\ \text{where } B_i = \textit{changes}(e_i) \text{ for all } i \in \{2, \ldots, n\}
\end{aligned}$$

Translation of ND effects into PL: example

$$\begin{aligned}
& \mathrm{PL}_{\{A,B,C,D\}}((0.5A|0.5(C \rhd A)) \wedge (0.5B|0.5C)) \\
= \ & \mathrm{PL}_{\{A,D\}}(0.5A|0.5(C \rhd A)) \wedge \mathrm{PL}_{\{B,C\}}(0.5B|0.5C) \\
= \ & (\mathrm{PL}_{\{A,D\}}(A) \vee \mathrm{PL}_{\{A,D\}}(C \rhd A)) \wedge \\
& (\mathrm{PL}_{\{B,C\}}(B) \vee \mathrm{PL}_{\{B,C\}}(C)) \\
= \ & ((A' \wedge (D \leftrightarrow D')) \vee (((A \vee C) \leftrightarrow A') \wedge (D \leftrightarrow D'))) \wedge \\
& ((B' \wedge (C \leftrightarrow C')) \vee ((B \leftrightarrow B') \wedge C'))
\end{aligned}$$

Translation of ND operators into PL

The translation of an operator $o = \langle c, e \rangle$ in normal form I is

$$\tau_o = c \wedge \mathsf{PL}_P(e)$$

where P is the set of all state variables.

Nondeterministic operators: translation into propositional logic

$$\mathcal{R}_1(P, P') = \tau_{o_1} \vee \cdots \vee \tau_{o_n} \vee ((p_1 \leftrightarrow p'_1) \wedge \cdots \wedge (p_k \leftrightarrow p'_k))$$

where $P = \{p_1, \ldots, p_k\}$ is the set of state variables.

Preimages: weak preimage, strong preimage

image	$\mathit{img}_R(S) = \{s' \mid s \in S, \langle s, s' \rangle \in R\}$
preimage	$\mathit{wpreimg}_R(S) = \{s \mid s' \in S, \langle s, s' \rangle \in R\}$
strong preimage	$\mathit{spreimg}_R(S) = \{s \mid s' \in S, \langle s, s' \rangle \in R, \mathit{img}_R(s) \subseteq S\}$

Strong preimage = the set of states from which a state in S is always reached (whatever nondeterministic choice is made.)

$\mathit{spreimg}_R(S) = \mathit{wpreimg}_R(S)$ whenever R is deterministic.

EXAMPLE: Consider $R = \{\langle s_1, s_2 \rangle, \langle s_1, s_3 \rangle\}$.
$\mathit{wpreimg}_R(\{s_2\}) = \{s_1\}$
$\mathit{spreimg}_R(\{s_2\}) = \emptyset$.

Computing strong preimages with formulas

As a propositional formula:

$$(\forall P'.(\mathcal{R}_o(P, P') \rightarrow (\phi[p'_1/p_1, \ldots, p'_n/p_n]))) \wedge (\exists P'.\mathcal{R}_o(P, P'))$$

Here $\forall p.\phi$ is *universal abstraction* that is defined analogously to existential abstraction as

$$\forall p.\phi = \phi[\top/p] \wedge \phi[\bot/p].$$

Computing strong preimages with formulas

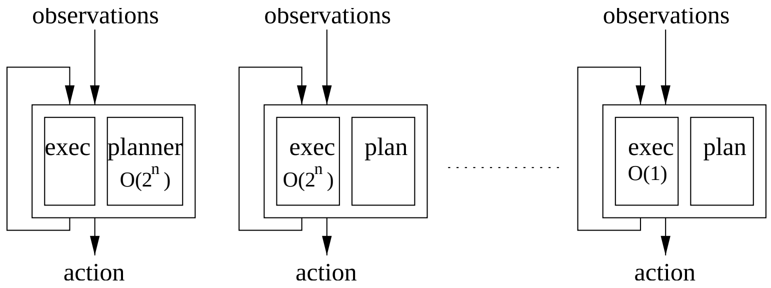
The formula

$$(\forall P'.(\mathcal{R}_o(P,P') \rightarrow (\phi[p'_1/p_1,\ldots,p'_n/p_n]))) \wedge (\exists P'.\mathcal{R}_o(P,P'))$$

determines the strong preimage as the set of states s such that (first and second conjunct, respectively):

1. for all states s', if $s\mathcal{R}_o s'$ then $s' \models \phi$, and
2. there is some state s' such that $s\mathcal{R}_o s'$ (to exclude states s without successors: they trivially satisfy the first conjunct.)

Conditional plans vs. planning interleaved with execution



Trade-off between plan size and how easy it is to execute!

Contingent plans vs. planning interleaved with execution

1. Choose only the action to be executed next: planning and execution are interleaved.
   - \+ No need to construct a (possibly very big) plan.
   - \- May be very very slow ($\geq$ the plan existence problem).
2. Construct a plan that handles all possible contingencies.
   - \+ Executing a plan can be very efficient.
   - \- A plan may be very big (if it is efficient to execute).

Conditional plans

- Next action is dependent on how nondeterminism worked.
- Many possibilities in representing these dependencies:
  1. Mappings from current states to an operator
  2. Mappings from sets of possible current states to an operator
  3. Programs

  The first two can be represented in terms of the third.

Conditional plans: definition

Program statements:

1. Apply an operator (and go to the next statement)
2. Choose the next statement based on values of state variables (case/switch/if-then-else)

Conditional plans: definition

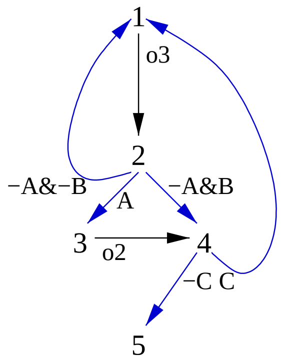
P = the state variables, O = the operators. A plan is $\langle N, b, l \rangle$ where

- N is a finite set of nodes,
- $b \in N$ is the initial node,
- $l : N \to (O \times N) \cup 2^{\mathcal{L} \times N}$ assigns each node
  - an operator and a successor node $\langle o, n \rangle \in O \times N$ or
  - a set of conditions and successor nodes $\langle \phi, n \rangle$ where $n \in N$ and ϕ is a formula over P.

Conditional plans: example

$$
\begin{aligned}
N &= \{1, 2, 3, 4, 5\} \\
b &= 1 \\
l(1) &= \langle o_3, 2 \rangle \\
l(2) &= \{\langle \phi_1, 1 \rangle, \langle \phi_2, 3 \rangle, \langle \phi_3, 4 \rangle\} \\
l(3) &= \langle o_2, 4 \rangle \\
l(4) &= \{\langle \phi_4, 1 \rangle, \langle \phi_5, 5 \rangle\} \\
l(5) &= \emptyset
\end{aligned}
$$

Conditional plans: example, cont'd



Conditional plans: example, cont'd

1: o_3
2: CASE
ϕ_1: GOTO 1
ϕ_2: GOTO 3
ϕ_3: GOTO 4
3: o_2
4: CASE
ϕ_4: GOTO 1
ϕ_5: GOTO 5
5:

Problem definition

A 4-tuple $\langle P, I, O, G\rangle$ consisting of

- a set P of state variables,
- a propositional formula I over P,
- a set O of operators, and
- a propositional formula G over P.

is *a problem instance in nondeterministic planning with full observability*.

A *solution* is a plan that always reaches goals when execution starts in an initial state. *We give 2 alternative formal definitions...*

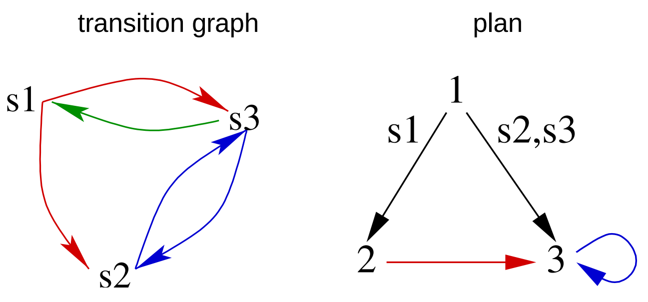
Execution graph of a plan

Let $\langle P, I, O, G\rangle$ be a problem instance and $\pi = \langle N, b, l\rangle$ be a plan. Then *the execution graph* of π is $\langle M, E\rangle$ where

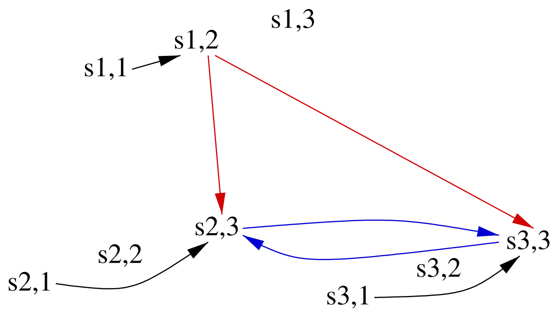
1. $M = S \times N$, where S is the set of Boolean valuations of P,
2. $E \subseteq M \times M$ has an edge from $\langle s, n\rangle$ to $\langle s', n'\rangle$ if and only if

(a) $l(n) = \langle o, n'\rangle$ and $s' \in img_o(s)$, or (operator node)
(b) $\langle \phi, n'\rangle \in l(n)$ and $s' = s$ and $s \models \phi$. (branch node)

Execution graph of a plan: example



Execution graph of a plan: example



Valid plans I

DEFINITION 1 (Plans without loops)

For all states s such that $s \models I$, every path that starts from (s, b)

1. *has finite length* and
2. *ends in a terminal node* (s', n) *such that* $s' \models G$.

Necessity of loops

Goal: Toss a coin until it lands on *heads*.

1. Planning problem is solvable!
2. No finite upper bound on execution length.
3. Infinite execution possible, but its probability is 0.

Valid plans II

DEFINITION 2 (Plans with loops)

For all states s such that $s \models I$, for every (s', n) to which there is a path from (s, b) that does not visit (s''', n'') for any s''' such that $s''' \models G$ and terminal node n'' there is also a path from (s', n) to some (s'', n') such that $s'' \models G$ and n' is a terminal node.