An Effective Approach to Realizing Planning Programs

Alfonso Gerevini[†], Fabio Patrizi[*] and Alessandro Saetti[†]

[†]University of Brescia, Italy
{gerevini,saetti}@ing.unibs.it

[*]Imperial College London, UK
fpatrizi@imperial.ac.uk

Introduction

- *Planning programs (p-programs)*: high-level, declarative representation of the behavior of agents acting in a domain [De Giacomo et al. AAMAS-2010]
- State transition systems labelled by domain goals and states representing decision points about which goal is next
- *Realizing a p-program*: finding and combining a collection of plans for the transition goals making the *p*-program *executable*
- The existing method for realizing a *p*-program is inefficient
- *We propose a planning-based approach for deterministic domains that is considerable faster*

Talk Outline

1. Planning program definition
2. Planning program realization
3. A planning-based algorithm
4. Experimental results
5. Conclusions and future work

Planning Programs (*P*-Programs)

Informally through an example

P-program: High-level, declarative representation of the behavior of an agent acting in a domain described by an automaton

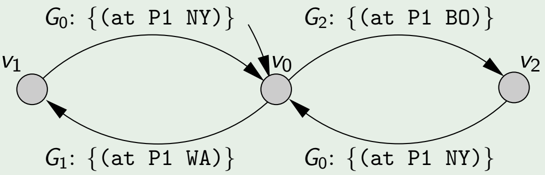
Planning Programs (*P*-Programs)

Informally through an example

P-program: High-level, declarative representation of the behavior of an agent acting in a domain described by an automaton

Example (Sale representative)

- On customer request, fly to WA (G_1) or BO (G_2)
- From BO and WA, required to return to NY (G_0)
- After returning, serve next (goal) request



Planning Programs

More formally

Consider a (deterministic) *planning domain* $\mathcal{D} = \langle P, A, \tau \rangle$, where:

- P: set of domain propositions
- A: set of domain actions
- $\tau : S \times A \longrightarrow S$: state transition function

(Call $S = 2^P$ the set of $\mathcal{D}$-states)

Planning Programs

More formally

Consider a (deterministic) *planning domain* $\mathcal{D} = \langle P, A, \tau \rangle$, where:

- P: set of domain propositions
- A: set of domain actions
- $\tau : S \times A \to S$: state transition function

(Call $S = 2^P$ the set of $\mathcal{D}$-states)

Definition (Planning Program for $\mathcal{D}$)

A *Planning Program* for $\mathcal{D}$ is a tuple $\mathcal{P} = \langle V, v_0, \Gamma, \delta \rangle$, where:

- V: (finite) set of $\mathcal{P}$-states
- $v_0 \in V$: initial $\mathcal{P}$-state
- Γ: set of possible goals in $\mathcal{D}$
- $\delta : V \times \Gamma \to V$: p-program transition function

P-Program Realization

Informally

The *execution of a p-program* $\mathcal{P}$ for $\mathcal{D}$ works as follows:

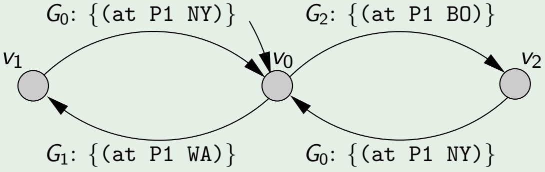
1. Initially, $\mathcal{D}$ and $\mathcal{P}$ are in the joint state $\langle s_0, v_0 \rangle$
2. When $\mathcal{D}$ and $\mathcal{P}$ are in joint state $\langle s, v \rangle$, a v-outgoing transition $\langle v, G, v' \rangle$ is selected from the p-program (if any)
3. A plan π achieving G from s is executed, leading $\mathcal{D}$ to s'
4. $\langle s', v' \rangle$ becomes the current joint state; a new iteration starts

Realizing a p-program (intuitively): building a plan for every transition selectable during the p-program execution in the current domain state

Remark: the transitions from joint states $\langle s', v \rangle$ and $\langle s'', v \rangle$ may require different plans if $s' \neq s''$

P-Program Realization
Example

Example (Sale representative, cont.)



Program realization function

State	Transition	Plan
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, g_2, v_2\rangle$	$\langle a_1, a_2, a_3, a_4\rangle$
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, g_1, v_1\rangle$	$\langle a_1, a_2, a_5, a_6\rangle$
s_1 = {(at P1 Bo), (at A1 Bo)}	$\langle v_2, g_0, v_0\rangle$	$\langle a_7, a_1, a_8\rangle$
s_2 = {(at P1 Wa), (at A1 Wa)}	$\langle v_1, g_0, v_0\rangle$	$\langle a_9, a_{10}, a_8\rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, g_2, v_2\rangle$	$\langle a_2, a_3, a_4\rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, g_1, v_1\rangle$	$\langle a_2, a_5, a_6\rangle$

$\mathcal{D}$-actions:

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

P-Program Realization

More formally

For a planning domain $\mathcal{D} = \langle P, A, \tau \rangle$:

- Π: set of all plans executable from some $\mathcal{D}$-state
- $s_0 \in S$: an initial $\mathcal{D}$-state

Definition (P-program Realization)

Given $\mathcal{D}$ and $\mathcal{P} = \langle V, v_0, \Gamma, \delta \rangle$, a **realization** of P is a partial function $\rho : S \times \delta \rightarrow \Pi$, inductively defined as follows:

- for every $\mathcal{P}$-transition $\langle v_0, G, v \rangle \in \delta$, $\rho(s_0, \langle v_0, G, v \rangle)$ is defined;
- if $\rho(s, \langle v, G, v' \rangle)$ is defined then:
  - $\pi = \rho(s, \langle v, G, v' \rangle)$ is a plan achieving G from s
  - $\forall \langle v', G', v'' \rangle \in \delta$, if $Result(s, \pi) = s'$, then $\rho(s', \langle v', G', v'' \rangle)$ is defined.

P-Program Realization

How to Compute a Realization?

Two proposed approaches:

- **Reduction to LTL-synthesis** [DeGiacomo&al@AAMAS10]
  - Pros: tools available (TLV); easy to handle non-deterministic domains as well
  - Cons: computationally inefficient
- **Planning-based approach** [*in this paper*]
  - Pros: can exploit fast planning technology and the problem structure to efficiently solve the realization problem
  - Cons: need dedicated algorithm; current algorithm supports only deterministic domains

A Planning-based Algorithm

Informally

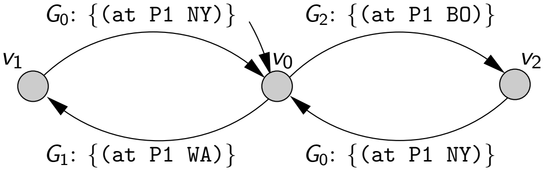
1. *Open* = set of joint (domain/program) states to process, initially set to $\langle s_0, v_0 \rangle$
2. **Repeat**
    3. Select a pair $\langle s, v \rangle$ from *Open*
    4. **Foreach** program transition d outgoing from v **do**
        5. Construct a plan π achieving the goals of d from s
        6. Update the realization function
        7. Progress the program and world states possibly generating a new joint state to process (pair added to *Open*)
8. **Until** *Open* is empty

Plans resulting in already generated domain states are *preferred*

(Preferred states are handled by soft goals compiled into PDDL2.1)

A planning-based algorithm

An example (part 1 of 4)



$Open = \{ \langle s_0, v_0 \rangle \}$

$State(v_0) = \{s_0\}$

$State(v_1) = \{\}$

$State(v_2) = \{\}$

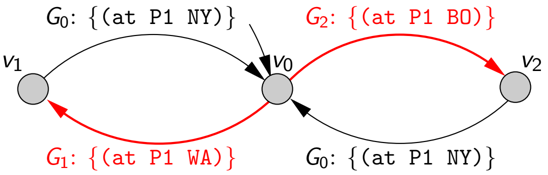
Program realization function under construction

State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	?
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

A planning-based algorithm

An example (part 1 of 4)



$Open = \{ \quad \}$
$State(v_0) = \{s_0\}$
$State(v_1) = \{\}$
$State(v_2) = \{\}$

Program realization function under construction

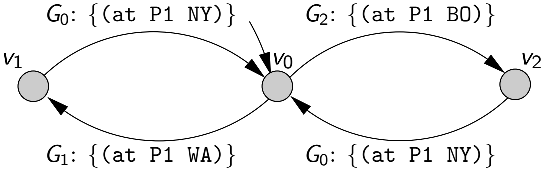
State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

Constructing plans for G_1 and G_2 from s_0

A planning-based algorithm

An example (part 1 of 4)



$Open = \{ \langle s_1, v_1 \rangle, \ \langle s_2, v_2 \rangle \}$

$State(v_0) = \{s_0\}$

$State(v_1) = \{s_1\}$

$State(v_2) = \{s_2\}$

Program realization function under construction

State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	?
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	?

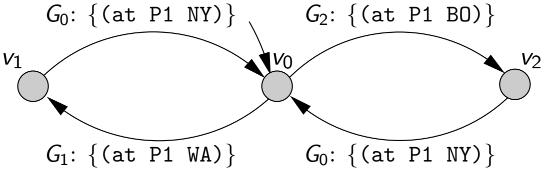
a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

The computed plans produce two new final states

s_1 for v_1 and s_2 for v_2

A planning-based algorithm

An example (part 2 of 4)



$Open = \{ \langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle \}$

$State(v_0) = \{s_0\}$

$State(v_1) = \{s_1\}$

$State(v_2) = \{s_2\}$

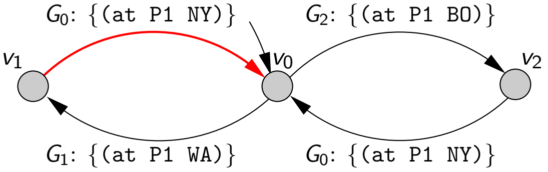
Program realization function under construction

State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	?
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

A planning-based algorithm

An example (part 2 of 4)



$Open = \{ \langle s_2, v_2 \rangle \}$
$State(v_0) = \{s_0\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

Program realization function under construction

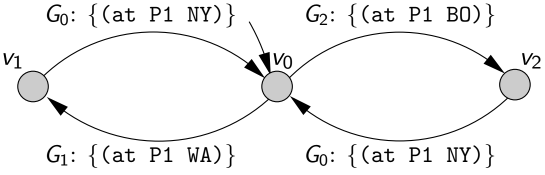
State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

Constructing a plan for G_0 *preferring* end state s_0

A planning-based algorithm

An example (part 2 of 4)



$Open = \{ \langle s_2, v_2 \rangle, \langle s_3, v_0 \rangle \}$

$State(v_0) = \{s_0, s_3\}$

$State(v_1) = \{s_1\}$

$State(v_2) = \{s_2\}$

Program realization function under construction

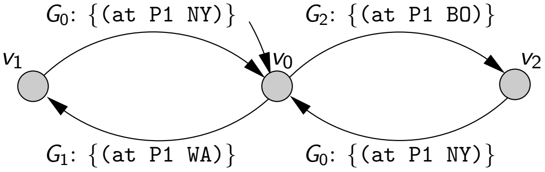
State	Transition	Plan
$s_0 =$ {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 =$ {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 =$ {(at P1 Bo), (at A1 Bo)}	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 =$ {(at P1 Wa), (at A1 Wa)}	$\langle v_1, G_0, v_0 \rangle$	?
$s_3 =$ {(at P1 NY), (at A1 NY)}	$\langle v_0, G_2, v_2 \rangle$	?
$s_3 =$ {(at P1 NY), (at A1 NY)}	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

The computed plan produces a new final states s_3 for v_0

A planning-based algorithm

An example (part 3 of 4)



$Open = \{ \langle s_2, v_2 \rangle, \langle s_3, v_0 \rangle \}$

$State(v_0) = \{s_0, s_3\}$

$State(v_1) = \{s_1\}$

$State(v_2) = \{s_2\}$

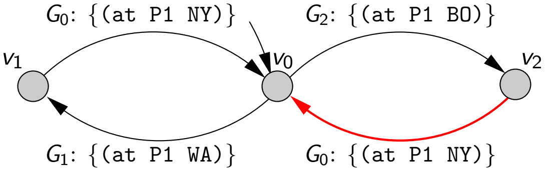
Program realization function under construction

State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	?
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_2, v_2 \rangle$	?
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

A planning-based algorithm

An example (part 3 of 4)



$Open = \{ \langle s_3, v_0 \rangle \}$
$State(v_0) = \{s_0, s_3\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

Program realization function under construction

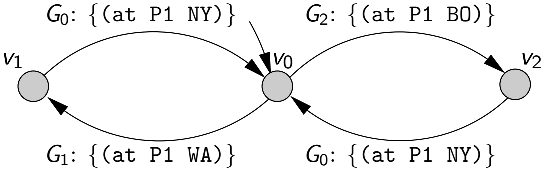
State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	$\langle a_9, a_{10}, a_8 \rangle$
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_2, v_2 \rangle$	?
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

Constructing a plan for G_0 *preferring* end states s_0 or s_3

A planning-based algorithm

An example (part 3 of 4)



$Open = \{ \langle s_3, v_0 \rangle \}$
$State(v_0) = \{s_0, s_3\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

Program realization function under construction

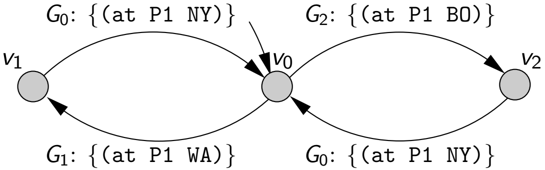
State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	$\langle a_9, a_{10}, a_8 \rangle$
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_2, v_2 \rangle$	?
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

The computed plan produces the preferred final state $s_3 \in State(v_0)$

A planning-based algorithm

An example (part 4 of 4)



$Open = \{ \langle s_3, v_0 \rangle \}$
$State(v_0) = \{s_0, s_3\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

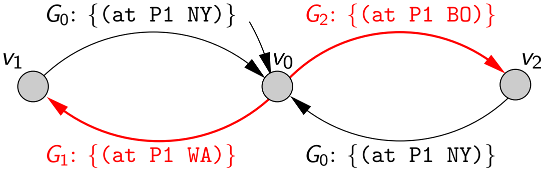
Program realization function under construction

State	Transition	Plan
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
$s_0 = \{$(at P1 NY), (at A1 Bo)$\}$	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
$s_1 = \{$(at P1 Bo), (at A1 Bo)$\}$	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
$s_2 = \{$(at P1 Wa), (at A1 Wa)$\}$	$\langle v_1, G_0, v_0 \rangle$	$\langle a_9, a_{10}, a_8 \rangle$
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_2, v_2 \rangle$	?
$s_3 = \{$(at P1 NY), (at A1 NY)$\}$	$\langle v_0, G_1, v_1 \rangle$	?

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

A planning-based algorithm

An example (part 4 of 4)



$Open = \{\ \ \}$
$State(v_0) = \{s_0, s_3\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

Program realization function under construction

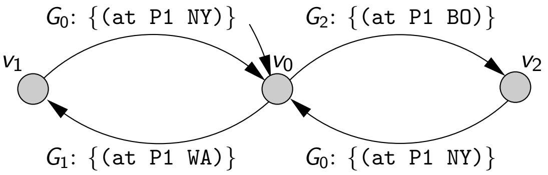
State	Transition	Plan
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_2, v_2\rangle$	$\langle a_1, a_2, a_3, a_4\rangle$
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_1, v_1\rangle$	$\langle a_1, a_2, a_5, a_6\rangle$
s_1 = {(at P1 Bo), (at A1 Bo)}	$\langle v_2, G_0, v_0\rangle$	$\langle a_7, a_1, a_8\rangle$
s_2 = {(at P1 Wa), (at A1 Wa)}	$\langle v_1, G_0, v_0\rangle$	$\langle a_9, a_{10}, a_8\rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, G_2, v_2\rangle$	$\langle a_2, a_3, a_4\rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, G_1, v_1\rangle$	$\langle a_2, a_5, a_6\rangle$

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

Constructing plans for G_1 and G_2 *preferring* end states s_1 and s_2

A planning-based algorithm

An example (part 4 of 4)



$Open = \{ \quad \}$
$State(v_0) = \{s_0, s_3\}$
$State(v_1) = \{s_1\}$
$State(v_2) = \{s_2\}$

Program realization function under construction

State	Transition	Plan
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_2, v_2 \rangle$	$\langle a_1, a_2, a_3, a_4 \rangle$
s_0 = {(at P1 NY), (at A1 Bo)}	$\langle v_0, G_1, v_1 \rangle$	$\langle a_1, a_2, a_5, a_6 \rangle$
s_1 = {(at P1 Bo), (at A1 Bo)}	$\langle v_2, G_0, v_0 \rangle$	$\langle a_7, a_1, a_8 \rangle$
s_2 = {(at P1 Wa), (at A1 Wa)}	$\langle v_1, G_0, v_0 \rangle$	$\langle a_9, a_{10}, a_8 \rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, G_2, v_2 \rangle$	$\langle a_2, a_3, a_4 \rangle$
s_3 = {(at P1 NY), (at A1 NY)}	$\langle v_0, G_1, v_1 \rangle$	$\langle a_2, a_5, a_6 \rangle$

a_1 : (fly A1 Bo NY)
a_2 : (board P1 A1 NY)
a_3 : (fly A1 NY Bo)
a_4 : (debark P1 A1 Bo)
a_5 : (fly A1 NY Wa)
a_6 : (debark P1 A1 Wa)
a_7 : (board P1 A1 Bo)
a_8 : (debark P1 A1 NY)
a_9 : (board P1 A1 Wa)
a_{10} : (fly A1 Wa NY)

The constructed plans produce the preferred final states s_1 and s_2 that are already in $State(v_1)$ and $State(v_2)$, respectively

A planning-based algorithm

Backtracking

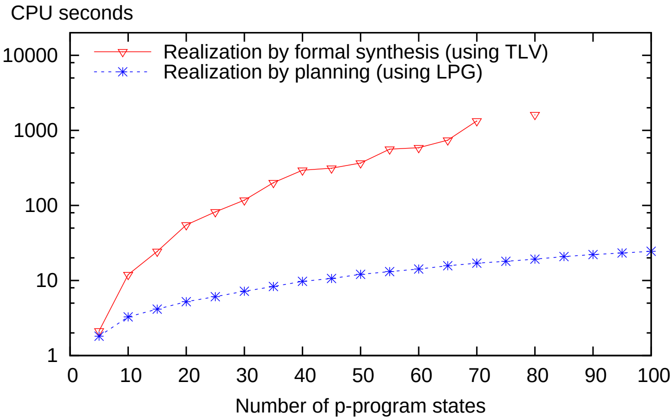
If for a pair $\langle s, v\rangle$ and a transition outgoing from v no realizing plan can be computed from s:

- State s is added to $Tabu(v) \Rightarrow$ s cannot be re-generated for v
- State s is removed from $State(v)$
- The realization function is updated (all plans generating s for the p-transitions ending in v are removed)
- *Open* is updated with the new frontier of the realization function

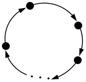
$\forall s \in Tabu(v)$ plans realizing a transition to v cannot end in s anymore
(compiled into a revised planning problem – new dummy goals and actions)

Experimental Results

Realizing planning programs by planning and formal synthesis



P-program structure

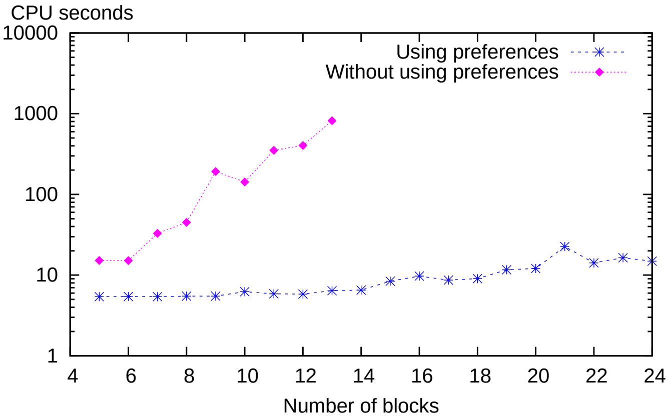


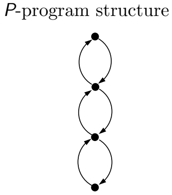
Planning programs with 5–100 program states forming a single cycle in `Blocksworld` with 2 blocks. CPU-time limit: 30 minutes

Planner: LPG

Experimental Results

Realizing planning program by planning with and without using preferences





Planning programs with 4 program states forming a sequence of multiple binary cycles in `Blocksworld` with 5–24 blocks.

Similar results with other domains and program structures

Conclusions

- We have proposed a new planning-based method for realizing *p*-programs over deterministic domains
- The approach is parametric wrt the planner realizing the transitions (much better if soft goals are supported)
- Experimental results show
  - Dramatic performance improvement wrt the previous technique based on LTL synthesis
  - The use of preferred states is very effective to deal with (possibly undirected) cycles in the planning program

Future Work

- Performance comparison of other algorithm versions obtained using different planners
- Additional experiments with other domains and planning program structures
- Integration of plan-adaptation techniques when (re)planning for a program transition
- Plan-based method handling non-deterministic domains

Encoding a preferred state into PDDL2.1

- Two new dummy literals `(dummy-fact)` and `(dummy-goal)`:
  - `(dummy-fact)` is added to the initial state and the domain action preconditions
  - `(dummy-goal)` is added to the problem goals
- A new numerical fluent `(utility)` that initially is 0
- A new dummy action `nopref` with `(dummy-fact)` as precondition and negative effect, and with `(dummy-goal)` as positive effect
- For every preferred state s, a new dummy action `pref-s` similar to `nopref` but with
  - the set of literals in s as additional preconditions, and
  - a numerical effect increasing fluent `(utility)` by a positive value
- A plan metric function maximizing `(utility)`

Experimental Results

Realizing planning program by planning with and without using preferences

Planning program		IPC6 score (#solved)		Average #open pairs	
Structure	$\|\delta\|$	+pref.	−pref.	+pref.	−pref.
Blocksworld					
1C[50]	50	20 (20)	4.81 (20)	51.2	164
SC[26]	50	20 (20)	0.29 (2)	92.0	695
CD[8]	56	20 (20)	1.0 (4)	245	627
Storage					
1C[50]	50	20 (20)	6.13 (20)	51.4	107
SC[26]	50	20 (20)	0.17 (2)	81.7	2934
CD[8]	56	20 (20)	0.80 (4)	228.5	3081
Zenotravel					
1C[50]	50	20 (20)	6.81 (20)	51.3	63.7
SC[26]	50	20 (20)	1.31 (7)	88.5	2454
CD[8]	56	20 (20)	0.0 (0)	281	3040

1C: one cycle; **SC**: chain of binary cycles; **CD**: complete directed graph

[n]: n p-program states

Related Work

A number of previous works address related issues:

1. (Baier&McIlraith@ICAPS06): heuristic search to build *finite* plans that satisfy temporally extended goals
   - plan finiteness does not allow to capture cycles in p-programs
2. (Kabanza&Thiebaux@ICAPS05): deals with temporally extended goals over infinite, deterministic (cyclic) *linear* plans
   - we need some form of non-determinism to capture the arbitrary selection of *p*-program transitions
3. (Kuter&al@ICAPS08): use of classical planners to find strong-cyclic reachability solutions to conditional planning problems

None can be straightforwardly used to compute a p-program realization ($\neq$ goal reachability!)

The General Setting

Features

(DeGiacomo&al@ICAPS10) propose a more general setting, where:

- the planning domain $\mathcal{D} = \{P, A, \rho\}$ is nondeterministic:
  - State transition $\tau \subseteq S \times A \times S$ is a *relation* instead of a function
- the transitions $\langle v, \varphi, G, v' \rangle \in \delta$ of a *p*-program $\mathcal{P} = \langle V, v_0, \Gamma, \delta \rangle$ may include also a *maintenance* goal:
  - φ represents a condition to be satisfied during plan execution, until G is achieved
  - this allows for capturing persistent goal requirements

The General Setting

Realization

The notion of realization needs to account for the facts that:

- plans are conditional
- maintenance goals must be satisfied

Definition (*P*-program Realization, general)

Given $\mathcal{D}$ and $\mathcal{P} = \langle V, v_0, \Gamma, \delta \rangle$, a *realization* of P is a partial function $\rho : S \times \delta \longrightarrow \Pi$, inductively defined as follows:

- for every $\mathcal{P}$-transition $\langle v_0, \varphi, G, v \rangle \in \delta$, $\rho(s_0, \langle v_0, \varphi, G, v \rangle)$ is defined;
- if $\rho(s, \langle v, \varphi, G, v' \rangle)$ is defined then:
  - $\pi = \rho(s, \langle v, \varphi, G, v' \rangle)$ is a conditional plan achieving G from s
  - all states reachable by executing π from s satisfy φ
  - for every $\langle v', \varphi, G', v'' \rangle \in \delta$, and for all s' s.t. $Result(s, \pi) = s'$, $\rho(s', \langle v', \varphi, G', v'' \rangle)$ is defined.

The General Setting

Complexity

Theorem

Building a p-program *realization over a nondeterministic planning domain is an EXPTIME-complete problem.*