Generalized Planning with Loops under Strong Fairness Constraints

Giuseppe De Giacomo[1] Fabio Patrizi[1] Sebastian Sardiña[2]

[1]Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma
Rome, Italy
degiacomo@dis.uniroma1.it
patrizi@dis.uniroma1.it

[2]School of Computer Science and IT
RMIT University
Melbourne, Australia
sebastian.sardina@rmit.edu.au

KR 2010 - Toronto, May 12, 2010

Nondeterministic Planning Domains

Nondeterministic Planning Domain $\mathcal{D}$:

- A finite set P of *propositions* –whose subsets are called *states*–, capturing all domain's relevant features
- A finite set A of *actions*, to be executed in the domain
- A transition *relation* $\longrightarrow \subseteq 2^P \times A \times 2^P$, representing the (possibly non-deterministic) domain dynamics, subject to action executions

Example (Coin Tossing)

- $P = \{head, tail, holding\}$, $A = \{grab, toss, turn\}$
- $\rho = \{\emptyset \xrightarrow{grab} \{holding\}, \{holding\} \xrightarrow{toss} \{head\}, \{holding\} \xrightarrow{toss} \{tail\}, \{head\} \xrightarrow{turn} \{tail\}, \{tail\} \xrightarrow{turn} \{head\}, \{tail\} \xrightarrow{grab} \{holding\}, \{head\} \xrightarrow{grab} \{holding\}\}$

Conditional Planning Problems

Conditional Planning Under Full Observability

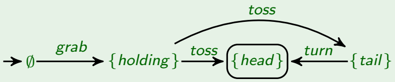
(For now, w/o loops)

- INPUT:
  - a nondeterministic domain $\mathcal{D} = \langle P, A, \rho \rangle$
  - an initial state $S_0 \subseteq P$
  - a propositional goal formula γ over P
- SOLUTION: a *conditional* plan π s.t. all *executions* achieve γ
- COMPLEXITY: EXPTIME-complete (also w/loops)

Example

$\langle S_0 = \emptyset, \gamma = head \rangle$ on *Coin Tossing* solved by plan:

1. *grab*
2. *toss*
3. if($\neg head$) then *turn*

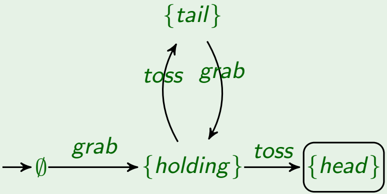


Conditional Planning with Loops

Loops allowed in plans

Example

π = while $(\neg head)$ $\{grab; toss\}$



- Clearly, π does not solve $\langle S_0 = \emptyset, \gamma = head \rangle$...
- ...however, in the real world, everyone would bet it *eventually* does!
- We want to assert non-local constraints!

Conditional Planning with Loops (2)

- Previous work on *Strong Cyclic Planning* [CPRT03] assumes *fair* action executions:
  - All action effects eventually occur
  - Cannot distinguish between fair and unfair action executions (either all or none!)
  - Thus, cannot make decisions based on this

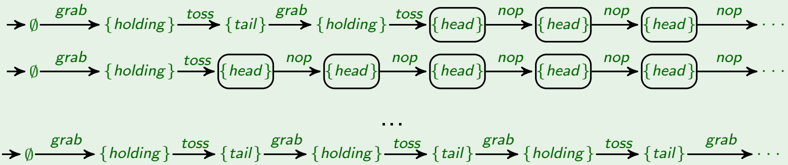
In this work

- We *explicitly* characterize *relevant runs*, through *constraints*
- Capture two different flavours of nondeterminism:
  - Uncertainty: the effect will occurr, but don't know exactly when (e.g., rolling a die)
  - Partial Knowledge: may or may not occur (e.g.: picking cards from a deck with a possibly missing Ace)

Constraints on Runs

Runs: possible evolutions of a domain, generated by executing plans

Example (π executions)



Constraints: LTL formulas built from propositions in $P \cup \bigcup_{a \in A}\{(act = a)\}$

Example (Constraints on $\mathcal{D}$ runs)

Tossing a coin infinitely often yields *head* to occur infinitely often

$$\Box\Diamond(act = toss) \rightarrow \Box\Diamond(head)$$

Constraints on Runs (2)

- *Constraints*: LTL formulas to be evaluated on domain runs
- We use run constraints to *rule out* irrelevant runs
- Only runs satisfying *all* constraints are significant

Semantics of constraints on domain runs

Given:

- a planning domain $\mathcal{D}$ with a finite set $\mathcal{C}$ of constraints on runs
- initial state S_0 and a goal formula γ

A conditional plan π with loops *achieves* γ ($\pi \models \top \mathcal{U} \gamma$) if all of its executions *satisfying* all $\mathcal{C}$ constraints reach a state S s.t. $S \models \gamma$

- LTL is very natural: Conditional Planning focuses on single executions (run-by-run)

Constraints on Runs (3)

We use run constraints to assert non-local domain properties

Example (More realistic Coin Tossing domain)

If we assert the following constraints on Coin Tossing:

- $\Box\Diamond(act = toss) \rightarrow \Box\Diamond(head)$
- $\Box\Diamond(act = toss) \rightarrow \Box\Diamond(tail)$

Then plan $\pi =$ while $(\neg head)$ $\{grab; toss\}$
solves $\langle S_0 = \emptyset, \gamma = head\rangle$

Indeed, the only unsuccessful π execution

$$\rightarrow \emptyset \xrightarrow{grab} \{holding\} \xrightarrow{toss} \{tail\} \xrightarrow{grab} \{holding\} \xrightarrow{toss} \{tail\} \xrightarrow{grab} \{holding\} \xrightarrow{toss} \{tail\} \xrightarrow{grab} \cdots$$

is *discarded* by first constraint above

Strong Fairness Constraints

Strong Fairness

If something ϕ_s *happens infinitely often, then something else* ϕ_e *happens infinitely often*

$$\Box\Diamond\phi_s \longrightarrow \Box\Diamond\phi_e$$

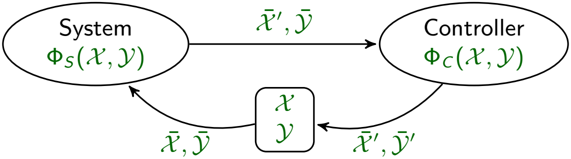
(ϕ_s and ϕ_e essentially propositional, $\bigcirc$ (Next) allowed)

- Strong Fairness captures also:
  - Weak fairness (something ϕ happens infinitely often): $\Box\Diamond\top \longrightarrow \Box\Diamond\phi$
  - Persistence (something ϕ holds from a point on): $\Box\Diamond\neg\phi \longrightarrow \Box\Diamond\bot$
- Not expressible in CTL [CGP99]

- Can phrase typical properties of our interest
- Good computational behavior (wrt Conditional Planning, see below)
- Existing results on LTL synthesis [PPS06, KPP05] apply

Synthesis in LTL

LTL Synthesis Problem:



- Uncontrolled ($\mathcal{X} = \{x_1, \ldots, x_n\}$) and controlled ($\mathcal{Y} = \{y_1, \ldots, y_m\}$) vars
- *System* assigns $\mathcal{X}$ vars (moves first), *Controller* assigns $\mathcal{Y}$ vars
- Both have their own *structural assumptions* (constraints on execution)

Objective:

- Find a controller strategy satisfying an LTL specification φ (Technically, φ combines Φ_C, Φ_S, and desired requirements. In particular, φ requires the strategy to meet Φ_C when interacting with Φ_S)

Synthesis in LTL (2)

Complexity:

- For arbitrary φ, the problem is 2EXPTIME-complete [PR89]
- GR(1) formulas yield an EXPTIME bound [PPS06, KPP05]

Generalized Reactivity (1) Specifications:

$$\varphi = \varphi_a \longrightarrow \varphi_r$$

- φ_a: *System structural assumptions* + possible (weak) fairness constraints
- φ_r: *Controller structural assumptions* + possible (weak) fairness constraints
- Express (desired) requirements of the form

$$\bigwedge_n \Box\Diamond\phi_i \longrightarrow \bigwedge_m \Box\Diamond\psi_j$$

- Expressive enough for our problem!

Conditional Planning w/ Loops under SFC as LTL Synthesis

$$\varphi = \varphi_a \longrightarrow \varphi_r$$

In our case:

1. $\varphi_a = \varphi_a^{init} \wedge \varphi_a^{trans} \wedge \varphi_a^{rc}$, where:
   - φ_a^{init}: initial condition ($\mathcal{D}$ state)
   - φ_a^{trans}: $\mathcal{D}$ transitions and goal achievement
   - φ_a^{rc}: constraints on $\mathcal{D}$ runs (phrased as weak fairness)

2. $\varphi_r = \varphi_r^{trans} \wedge \varphi_r^{goal}$, where:
   - φ_r^{trans}: one executable action at each point (plan structure)
   - φ_r^{goal}: achieve desired goal γ (phrased as weak fairness)

Main Results

Theorem (Soundness & Completeness)

Conditional Planning w/ Loops under Strong Fairness Constraints can be reduced to LTL synthesis for GR(1) formulas

Theorem (Complexity)

Building a conditional plan with loops under strong fairness constraints is EXPTIME-complete

Same complexity class as Conditional Planning w/ Full Observability!

Implementation

- Actual system available: TLV
- Based on (global) Model Checking techniques

Other Results

(See paper)

In general, approach shown effective for:

1. Goals of the form $\varphi = \psi \mathcal{U} \phi$ (achieve ϕ while maintaining ψ)
2. Planning Programs [DPS10], whose atomic instructions are goals $\phi \mathcal{U} \psi$, can be captured and realized
3. Component-based Planning: actions offered by (finite-state) devices, possibly subject to strong fairness constraints

Conclusions and Future Directions

1. Conditional Planning w/ loops, with non-local constraints explicitly asserted
2. More general but not computationally harder than Conditional Planning w/ out Loops
3. Tackled as an LTL synthesis problem, actual system available
4. Suitable for extended scenarios (Planning Programs, Component-based Planning)

1. Performance evaluation to be carried out
2. Plan quality: e.g., avoid loops when not required
3. Planning-oriented techniques and heuristics

References

Edmund M. Clarke, Orna Grumberg, and Doron A. Peled.
Model checking.
The MIT Press, Cambridge, MA, USA, 1999.

A. Cimatti, M. Pistore, M. Roveri, and P. Traverso.
Weak, strong, and strong cyclic planning via symbolic model checking.
Artificial Intelligence Journal, 147(1-2):35–84, 2003.

Giuseppe De Giacomo, Fabio Patrizi, and Sebastian Sardina.
Agent programming via planning programs.
In *Proc. of AAMAS'10*, Toronto, Canada, 2010.
To appear.

Yonit Kesten, Nir Piterman, and Amir Pnueli.
Bridging the gap between fair simulation and trace inclusion.
Information and Computation, 200(1):35 – 61, 2005.

Nir Piterman, Amir Pnueli, and Yaniv Sa'ar.
Synthesis of reactive(1) designs.
In *Proc. of VMCAI'06*, volume 3855 of *Lecture Notes in Computer Science (LNCS)*, pages 364–380. Springer, 2006.

Amir Pnueli and Roni Rosner.
On the Synthesis of a Reactive Module.
In *Proc. of POPL'89*, pages 179–190, 1989.