Non-Deterministic Conformant Planning Using a Counterexample-Guided Incremental Compilation to Classical Planning

Enrico Scala, Alban Grastien
University of Brescia (Italy)Australian National University

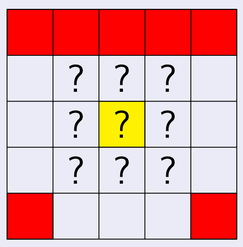
Abstract

We address the problem of non-deterministic conformant planning, i.e., finding a plan in a non-deterministic context where the environment is not observable. Our approach uses an unsound but complete reduction from non-deterministic conformant planning to classical planning to find a candidate plan; the validity of this plan is then verified by a SAT solver; if the plan is invalid, the reduction is revised to guarantee that the invalid plan will not be valid in the classical planning problem. This procedure is executed until a valid plan is found, or it is shown that there is no plan. Experiments show that this approach is competitive with the existing solvers, and is able to solve difficult instances.

Setting: Non-Deterministic Conformant Planning

- Class of problems
  - Uncertainty in the initial state. *I don't know exactly where I start from*
  - Uncertainty in the action effects. *Transitions are non-deterministic*
  - No observation allowed. *Plans are just sequence. Actions need to be applicable regardless uncertainty*
- Syntax
  - Problem is a tuple $\mathbb{P} : \langle I, G, A\rangle$ where
    - I is given as a formula, representing the **belief** of the agent about the initial situation. Each model is a possible world, also called a state.
    - A is a set of actions, An action $a \in A$ is a tuple $\langle \text{name}(a), \text{pre}(a), \text{eff}(a)\rangle$ where: (i) $\text{name}(a)$ is a label (ii) $\text{pre}(a)$ is precondition and (iii) $\text{eff}(a)$ is a set of conditional effects $\{CE_1, \ldots, CE_m\}$ such that each CE_i is a relation $\phi \rhd \{e_1, \ldots, e_k\}$ where ϕ is a formula.
- Semantics
  - Action applicability: a is applicable in a belief $\mathcal{B} \in 2^S$ iff all states in $\mathcal{B}$ satisfy $\text{pre}(a)$. The execution of an action a in a belief $\mathcal{B}$, $\mathcal{B}[a]$, is a new belief that is obtained as follows: $\mathcal{B}[a] = \{s[o] \mid s \in \mathcal{B} \wedge o \in O(a)\}$.
  - Plan $\pi = \langle \text{name}(a_1), \ldots, \text{name}(a_n)\rangle$ is *valid* iff each a_i is applicable in $\mathcal{B}_{i-1}$ and its last action is the goal action.

Faulty-Robot Example



- Actions
  - Can traverse only white cells
  - Can move Up (↑), Down (↓), Right (→), Left (←).
  - → **non deterministically** slides up, too
  - Agent remains put when moving towards one of the borders
- Initial State
  - Any cell marked with "?", eg. over x $\phi = oneof_{i\in\{1,\ldots,3\}} x(i)$
- Goal
  - Being at the center, yellow position

Solution?

$$\pi = \langle 3\times\downarrow, \uparrow, 3\times\leftarrow, 2\times(\rightarrow,\downarrow), \downarrow, 2\times\uparrow\rangle$$

State of The Art, Issues and Research Question

- Mainly two ways being studied
  - Search over Belief State
    - e.g., MBP by Cimatti et al. [2004]
    - Main Issue: not many heuristics in this search space
  - Compilation into a simpler problem, i.e. deterministic conformant planning
    - e.g., k1k0 by Albore et al. [2010]
    - Main Issue: difficult to make it complete, as it is difficult to predict bound on the number of action occurrences

Research Question

Is there a compilation that is complete and effective at the same time?
Can we do this by exploiting CEGAR?

Background

CEGAR in a nutshell

1. Start with some relaxation of the problem
2. Find a solution for such a relaxation
   1. If such a solution does not exist, the **problem** is **unsolvable**
3. Check whether the solution is valid and if it is not add some motivation to why this is the case
   1. This is what is called a **counterexample**
4. Refine the relaxation with the found counterexample if solution is not fvalid, otherwise **exit with a plan**
5. Go to step 1

Belief Tracking (Bonet and Geffner [2014])

The *context* $ctx(f)$ of a fact f is the smallest set that satisfies:

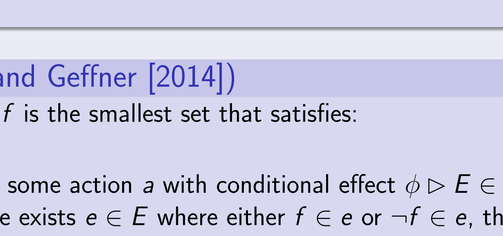
- $f \in ctx(f)$;
- For all $f' \neq f$, if there is some action a with conditional effect $\phi \rhd E \in \text{eff}(a)$, such that i) f' appears in ϕ and ii) there exists $e \in E$ where either $f \in e$ or $\neg f \in e$, then $f' \in ctx(f)$;
- If $f'' \in ctx(f')$ and $f' \in ctx(f)$ then $f'' \in ctx(f)$.

Ingredients for our setting

1. Counterexamples through "Adversarial" NFA
   1. So as to capture **invalid plans**
2. SAT encoding for searching counterexamples
3. Finding candidate plans by classical planning
4. Speed up through Adversarial Automaton Decomposition

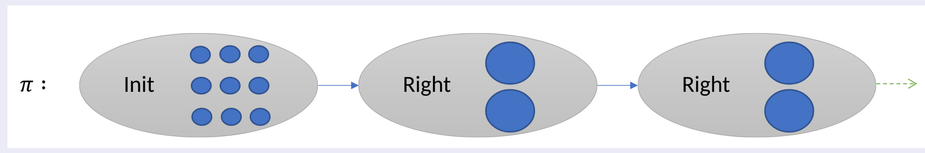
Counterexamples and Adversarial NFA

- A counterexample is a sequence of states actions that starts from the initial state, and ends with an inapplicable actions (pseudo actions init and goal). Initial uncertainty compiled into "init" action; Goal compiled into precondition of dummy goal action
  - e.g., $\tau_0 = q_0 \xrightarrow{\text{"Init"}} q_1 \xrightarrow{Left} q_3 \xrightarrow{Up} \bot$
  - e.g., $\tau_1 = q_0 \xrightarrow{\text{"Init"}} q_2 \xrightarrow{Right} q_3 \xrightarrow{\text{"Goal"}} \bot$
- All counterexamples stored in NFA $\mathcal{A}^{\{\tau_0,\tau_1\}} = \langle Q, \Sigma, T, q_I, F\rangle$. For instance:



- A counter-automaton so built accepts only invalid plans, and is therefore named adversarial. Note that, $\Pi(\mathbb{P}) \cap \mathcal{L}(\mathcal{A}) = \emptyset$.

Finding Counterexamples via SAT



SAT Encoding

$$IA = \bigwedge_{f\in I} f@0 \wedge \bigwedge_{f\in F\setminus I} \neg f@0$$

$$FA = \bigwedge_{\substack{i\in\{0,\ldots,n\}\\ f\in F}} \{(\neg f@(i) \wedge f@(i+1)) \rightarrow ach(f,i)\} \wedge \bigwedge_{\substack{i\in\{0,\ldots,n\}\\ f\in F}} \{(f@(i) \wedge \neg f@(i+1)) \rightarrow ach(\neg f,i)\}$$

$$EA = \bigwedge_{\substack{i\in\{0,\ldots,n\}\\ f\in F}} ach(f,i) \rightarrow f@(i+1) \wedge ach(\neg f,i) \rightarrow \neg f@(i+1)$$

$$UP = \bigvee_{i\in\{1,\ldots,n\}} \neg \text{pre}(a_i)@i$$

where

$$ach(\ell,i) = \bigvee_{\substack{c,e \text{ such that}\\ \exists E.\ (c,E)\in a_i \wedge e\in E \wedge \ell\in e}} \left(c@i \wedge d^{c,e}@i\right).$$

Intuition: Search for some allocation of the action effects such that there is at least a precondition that is not satisfied.

Lemma

$IA \wedge FA \wedge EA \wedge UP$ is SAT iff there exists a counterexample

Computing Plans via Classical Planning

One-Outcome Determinisation

Arbitrarily pick only one effect from the non-deterministic effects. Example: assume action RIGHT always moves the robot up, too. Note that this is a relaxation of the main problem.

NFA Synchronisation

- $F' = F \cup \bigcup_{q\in Q} p_q$
- $A' = \{a^{\mathcal{A}} \mid a \in A\}$
- $I' = I \cup \{p_{q_I}\}$

where $a^{\mathcal{A}} = \langle \text{name}(a), \text{pre}(a) \wedge \text{pre}^{\mathcal{A}}(a), \text{eff}(a) \cup \text{eff}^{\mathcal{A}}(a)\rangle$ such that

- $\text{pre}^{\mathcal{A}}(a) = \bigwedge_{q'\in F} \bigwedge_{q\in pred_{\mathcal{A},a}(q')} \{\neg p_q\}$ ⇒ Prevent to apply invalid prefixes
- $\text{eff}^{\mathcal{A}}(a) = \bigcup_{q'\in Q} \left\{\bigvee_{q\in pred_{\mathcal{A},a}(q')} p_q \rhd p_{q'}\right\} \cup \bigcup_{q'\in Q} \left\{\neg \bigvee_{q\in pred_{\mathcal{A},a}(q')} p_q \rhd \neg p_{q'}\right\}$ ⇒ Advance the automaton

Adversarial Automaton and Contexts

- Maintain a counter-automaton for each context and add the projected counter-example onto the right counter-automaton.
- Example: assume plan down, up, then goal action. If we focus on the goal precondition $\varphi = x(3)$, the context of φ is $\{x(1), \ldots, x(5)\}$ since the y position never affects the x position of the robot The counter-example is therefore

$$\{x(2), y(2)\} \xrightarrow{Down} \{x(2), y(1)\} \xrightarrow{Up} \{x(2), y(2)\} \xrightarrow{\text{"Goal"}} \bot,$$

and its projection on $\{x(1), \ldots, x(5)\}$ is

$$\{x(2)\} \xrightarrow{Down} \{x(2)\} \xrightarrow{Up} \{x(2)\} \xrightarrow{\text{"Goal"}} \bot.$$

- Two main benefits:
  - more generalisation (can't do multiple Down or Up to avoid counterexample)
  - fewer additional facts (do not consider all combinations of states)

Non-Deterministic Conformant Planning Algorithm

```
1:  A := A^∅
2:  loop
3:      π := Plan(P^A_D)
4:      if π is ⊥ then
5:          return UNSAT                ▷ Plan can't be found
6:      end if
7:      τ := check(π, P)
8:      if τ = ⊥ then
9:          return π                    ▷ Plan found
10:     end if
11:     A := Ref(A, τ)
12: end loop
```

Properties

- Soundness
  - The algorithm only returns plan proved to be valid, plus classical planning call is a proper relaxation
- Completeness
  - The number of iterations is bounded by the size of the automaton, whose state space is bounded by number of states in $\mathbb{P}$

Experimental Results

Setting

Two non-deterministic planners from the literature:

- MBP (Complete, Belief Space), Cimatti et al. [2004]
- K1K0 (Incomplete, Compilation), Albore et al. [2010]

Test against two variants of our approach, VANILLA and DECO. DECO exploits contexts (Bonet and Geffner [2014])

Results

Domain	I	Contexts	Complete Methods VANILLA	DECO	MBP	k1k0
UTS	3	(9-15)	0	1	1	**3**
COINS	3	(21-45)	0	2	2	**3**
TRAIL	3	(2)	1	**3**	0	0
MOVE-PKG	4	(18-31)	3	**4**	**4**	0
BOMB-1-T	40	(2)	21	**40**	19	**40**
BOMB-N-T	40	(4)	14	**40**	17	**40**
MOUSE-CAT	3	(1925-7845)	**3**	**3**	0	2
FAULTY-ROBOT	24	(5)	4	12	**21**	0
Total	120		46	**105**	64	88

Conclusion and Future Work

- New compilation from non-deterministic to classical planning that is sound and complete
- Better One-Outcome determinisation?
- Better selection of counterexamples?