Fair LTL Synthesis for Non-Deterministic Systems using Strong Cyclic Planners

Elective in Artificial Intelligence

Reasoning Agents

Prof. Fabio Patrizi

Lorenzo Nicoletti - 1797464

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

- Introduction
- Overview of Nondeterministic Planning, LTL and Büchi Automata
- The Problem
- Reduction to Strong Cyclic Planning
- Implementation
- Experiments and Results
- Conclusions

Introduction

The pipeline of the paper can be summarized as follows:

Fully-observable non-deterministic planning problem with **LTL goals**

→ Mapping LTL goals to deterministic **Büchi automata**

→ Reduction to **strong cyclic planning** task

→ Transformation to **a classical planing** task

Overview of Nondeterministic Planning, LTL and Büchi Automata

- A nondeterministic planning domain is a tuple $\boldsymbol{D = \langle Act,\ Prop,\ S,\ s_0,\ f \rangle}$ where:
  - Act is the finite set of domain actions;
  - $Prop$ is the set of domain propositions;
  - $S \subseteq 2^{Prop}$ is the set of domain states;
  - $s_0 \in S$ is the (single) initial state;
  - $f : S \times Act \to 2^S$ is the state-transition function.
- We will assume that the action non-determinism is ***fair***.

Fair =	infinite executions of a non-deterministic action in the same state yield each possible successor state an infinite number of times.
	whenever an action is executed infinitely often, all of its effects take place infinitely often

Overview of Nondeterministic Planning, LTL and Büchi Automata

There is a tight relation between LTL and Büchi automata on infinite words.

- A Büchi Automata is a tuple $\boldsymbol{\mathcal{A} = \langle \Sigma, Q, Q_0, \rho, F \rangle}$ where:
  - Σ is the finite input alphabet of the automaton;
  - Q is the finite set of automaton states;
  - $Q_0 \subseteq Q$ is the set of initial states of the automaton;
  - $\rho : Q \times \Sigma \rightarrow 2^Q$ is the automaton transition function;
  - $F \subseteq Q$ is the set of accepting states.

Theorem:
For every LTL formula φ, one can effectively construct a Büchi automaton $\mathcal{A}_\varphi$ whose number of states is at most exponential in the length of φ and such that $L(\mathcal{A}_\varphi) = M(\varphi)$, where $L(\mathcal{A}_\varphi)$ is the language accepted by the automaton and $M(\varphi)$ is the set of all models of φ.

The Problem: LTL fair realization

Given a non-deterministic planning domain D and a deterministic LTL goal φ, build a closed **finite-state controller (FSC) Π** that fairly realizes φ on D.

A FSC can be seen as a machine whose input and output alphabets are the states and the actions of D.

- We can facilitate the search for Π that achieves φ by finding a **policy π** mapping state-pairs $\langle q, s \rangle$ into actions, where q is a Büchi automaton state and s is a domain state.

Reduction to Strong Cyclic Planning

A ***strong cyclic plan*** over a non-deterministic planning domain is a policy π such that if s is a non-goal state that is potentially reachable from the initial state by following π, then a goal state must be potentially reachable from s by following π.

Theorem:
If there exists a solution to the fair realization problem defined by a non-deterministic planning domain D and a deterministic LTL formula φ, then a non-deterministic planning problem D' can be constructed such that the strong cyclic solutions to D' yield a **FSC that fairly realizes $\boldsymbol{\varphi}$ on $\boldsymbol{D}$**.

Implementation

The derivation of the strong cyclic planning problem P_φ from the non-deterministic domain P and an LTL goal φ (mapped to a Büchi automaton A_φ) was done in two ways:

- Sequential encoding, domain actions are followed by actions that progress the state of the automaton;
- Parallel encoding, a new action is created for representing each possible sequence of a domain action followed by an automaton transition.

Then, the strong cyclic planning problem was solved using the PRP planner, that maps it into a classical planning problem.

Such configuration was tested against the existing symbolic synthesis tool called TLV.

Experiments and Results

Three tested scenarios:

Lift

- Goal: build a lift controller for a n-floor building.
- Fluents: at_i and req_i
- Actions: $push\_f_i$, $move\_up\_from\_f_i$ and $move\_down\_from\_f_i$

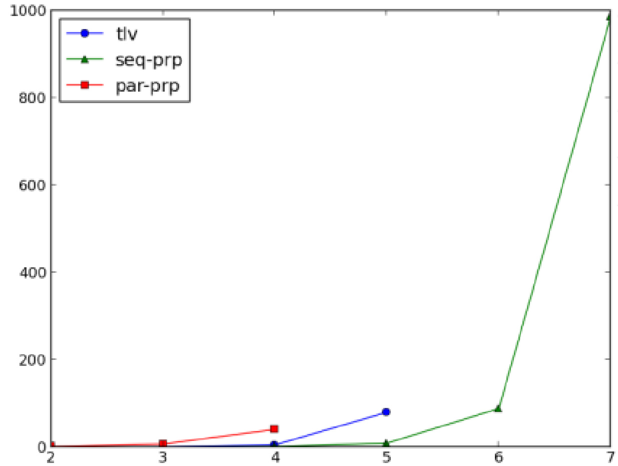
Waldo

- Goal: move a robot between locations until Waldo is found.
- Waldo can appear non-deterministically in room i or $i/2$. The robot must thus “patrol” rooms i and $i/2$ until Waldo appears.

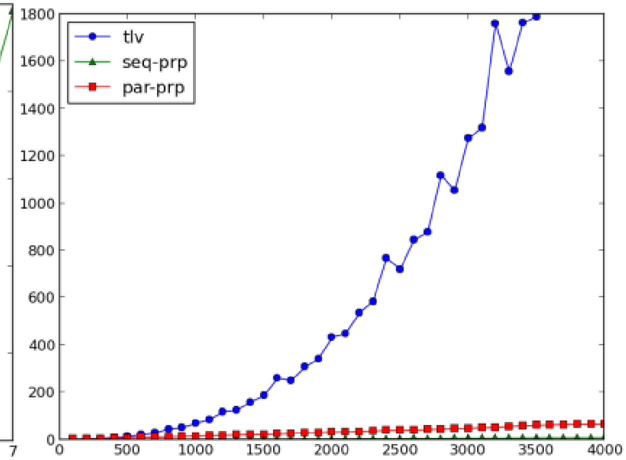
Clerk

- Goal: build a controller which guarantees that every client request is served.
- Fluents: $instore\_p_i$ and $want\_p_i$.
- Actions: $request$, $sell\_p_i$ and $buy\_supply\_p_i$. Extension with $pick$, $sell$ and $store$.

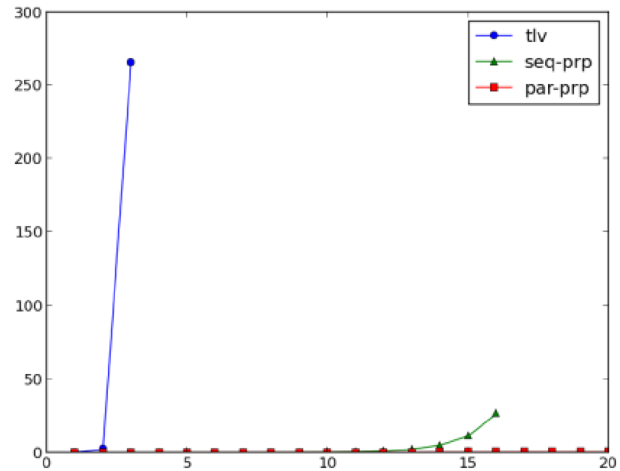
Experiments and Results



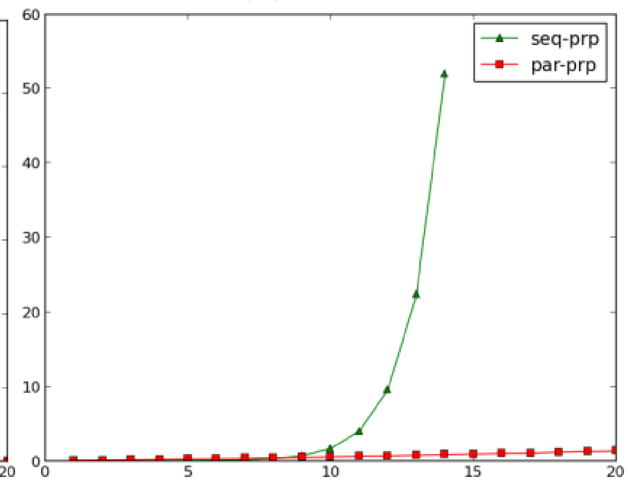
(a) Lift



(b) Waldo



(c) Clerk



(d) Clerk Extended

Conclusions

- The problem is a generalization of the fully-observable non-deterministic planning problem (FOND) where reachability goals are replaced by temporally extended LTL goals.
- The paper has shown that the problem can be compiled into a strong cyclic planning problem that can be solved by classical planners.
- The experiments show that the approach is computationally meaningful and has potential benefits in relation to standard symbolic synthesis methods.
- The proposed formulation extends the scope of current planning methods, which can thus be used effectively to generate controllers for systems that must operate continuously.

References

- **Fabio Patrizi, Nir Lipovetzky, Hector Geffner**
  “Fair LTL Synthesis for Non-Deterministic Systems using Strong Cyclic Planners",
  IJCAI, 2013.