Nondeterministic Planning via Symbolic Model Checking

Angelo Montanari

Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine

Table of Contents

Automated Planning: the main issues

Planning

Planning as model checking

Planning via symbolic model checking

The main issues

Action-based planning vs. timeline-based planning

The main issues (partial list)

- Expressiveness
- Decidability and complexity
- Planning as satisfiability checking
- **Planning as model checking**
- Synthesis of plan controllers
- Monitoring of plan execution

The Planning Problem

The planning problem: finding a sequence of actions that bring a system from an initial state to a goal state, given a set of possible actions (**action-based planning**).

Elements of the problem:

- Context/Domain:
  - Fluents/variables that describe the considered domain (a.k.a. environment/world)
    - **inertial fluents**, whose value can be changed by the execution of some plan action only
    - **non-inertial fluents**, whose value can vary independently of the execution of plan actions
  - States (the set of possible configurations)
  - Actions that cause the transition from a state to the next one
- Initial states
- Final/goal states

The Planning Problem: the simple case

The **standard (restricted) setting**

- Fluents/variables take value over finite/discrete domains
- Complete observability
- No external events that may affect plan execution
- Deterministic actions: no uncertainty about the effects of the execution of an action

Nondeterminism and extended goals

Nondeterminism

- External events that may cause plan-independent changes in the system
- Partial specification of the initial states
- Uncertainty about the effects of actions (nondeterminism)

Extended goals

- properties that the system must satisfy during the execution of the plan, e.g., integrity constraints

Classification of plans: situated, universal, and conformant

- **Situated plans**
  The evolution of the plan may be influenced by the environment. Environmental parameters are repeatedly measured, and the selection of the action to execute takes into account such measurements (**trial and error strategies**).
- **Universal plans** (a special case of situated plans)
  Universal plans associate at least one action with any possible state of the domain. In any system configuration, it is thus possible to find a planned action to execute.
- **Conformant plans**.
  In conformant plans, there is no need to collect any piece of information during the execution of the plan in order to achieve the goal. They are useful when observations of the environment are not possible (**null observability**).

Planning as model checking

Planning as model checking.

The **domain** of the planning problem and its **goal** can be represented by a formal **model** and a **formula** of a (temporal) logic, respectively.

The **(nondeterministic) planning domain** (P, S, A, R):

- P: a finite set of proposition letters P of the form: $fluent_i = value_j$
- $S \subseteq 2^P$: a finite set of states (a state is defined by the collection of the proposition letters that hold at it)
- A: a finite set of actions
- $R \subseteq S \times A \times S$: the transition relation (transition function in case of deterministic planning)

We say that an action a is **enabled** in a state s if there exists a state s' such that $R(s, a, s')$.

A running example

Consider a **parcel** that must transferred from the train station to the airport. The position of the parcel defines both the **initial condition** ("at the **train station**") and the **goal** ("at the **airport**").

On the path there is a **traffic light** where the vehicle that transports the parcel has to possibly stop waiting for the green light.

The **color** (red or green) of the **traffic light** introduces some form of **nondeterminism** of the domain (it will be modeled by means of a non-inertial variable), while the position of the parcel (train station, traffic light, or airport) may only change as the effect of the transportation action (inertial variable).

The only **possible actions** to execute to achieve the goal are thus moving the parcel from one position to the next one and waiting for the green light.

The model of the example domain

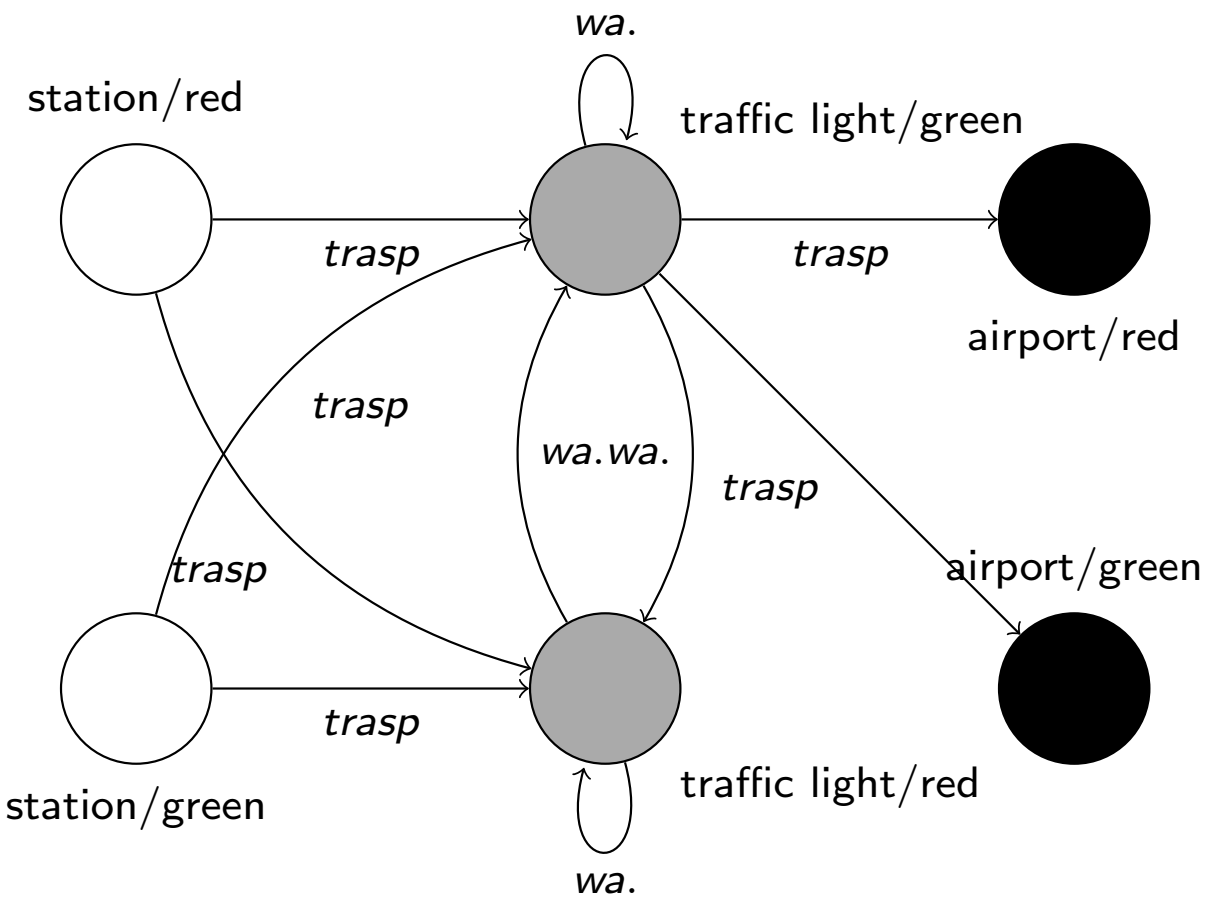

Figure: An example of *planning domain*

Initial states: white states; goal states: black states. Two variables: position (inertial) and color of the traffic light (non-inertial).

Plans as state-action tables

How do we **represent a plan**?

- A **plan** can be represented by means of a **state-action** table SA, that is, a set of pairs (s, a), where $s \in S$, $a \in A$, and a is enabled in s.

  Special cases. Deterministic plans: for each $s \in S$, there is at most one pair $(s, a) \in SA$. Universal plans: for each $s \in S$, there is at least one pair $(s, a) \in SA$.

- An **alternative representation** of plans can be given that distinguishes between the states of the agent/system and the possible contexts.

  It makes use of the following constructs:

  - a function $\textsc{act}(q, c)$, that specifies the action to execute when the agent/system is in state q in context c
  - a function $\textsc{ctxt}(q, c, q')$, that associates a new context with the state q' which is reached by executing the action $\textsc{act}(q, c)$

Execution structures

The execution of a state-action table (a plan) in a planning domain can be described in terms of the **transitions induced by the state-action table** (execution structure). The resulting graph specifies all the possible executions of the plan.

Formally, let SA be a state-action table of a planning domain (P, S, A, R). The **execution structure** induced by SA from the set of initial states I is a Kripke structure $K = (Q, T)$, where $Q \subseteq S$ and $T \subseteq S \times S$ are the minimal sets satisfying the following conditions:

- if $s \in I$, then $s \in Q$
- if $s \in Q$ and there exists $(s, a) \in SA$ such that $R(s, a, s')$, then $s' \in Q$ and $T(s, s')$

A state $s \in Q$ is a **terminal state** of K if there exists no $s' \in Q$ such that $T(s, s')$.

Execution paths

An execution structure is a **directed graph**, whose nodes are all the states that can be reached by executing the actions in the state-action table and whose edges represent possible action executions. It is not required to be total, that is, it may include nodes (states) with no outgoing edges (terminal states). Terminal states are states where the execution stops.

Let $K = (Q, T)$ be the execution structure induced by a state-action table SA from I. An **execution path** of K from $s_0 \in I$ is a (possibly infinite) sequence of states of Q such that, for all states s_i in the sequence, (i) either s_i is the last state of the sequence (a terminal state of K) or (ii) $T(s_i, s_{i+1})$.

We say that a state s' is **reachable** from a state s if there is path from s to s'. We say that K is an **acyclic** execution structure if all its execution paths are finite.

An example of execution structure

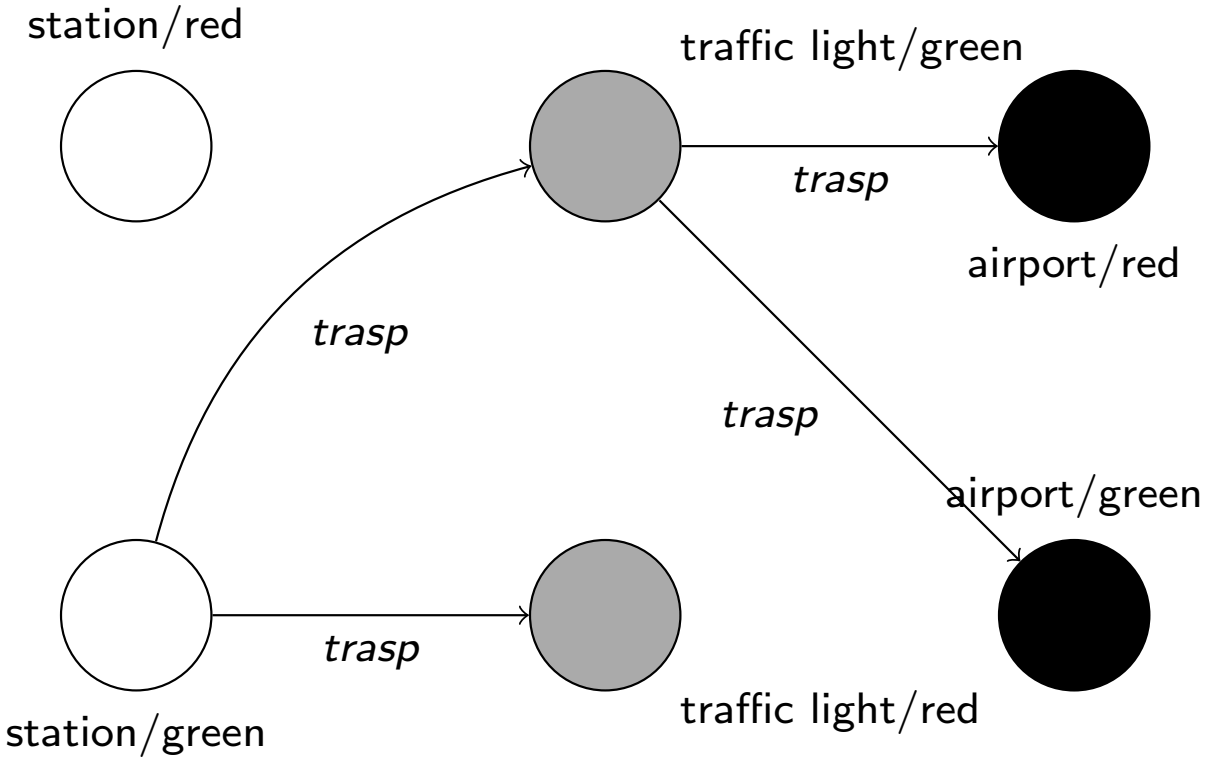

Figure: An example of *execution structure*

Notice that such a plan does not guarantee to always reach the goal. Basically, it executes the action of **transportation** whenever the color of the traffic light is **green**.

The planning problem: a definition

Let $D = (P, S, A, R)$ be a planning domain. A **planning problem** for D is a triple (D, I, G), where $I \subseteq Q$ is the set of initial states and $G \subseteq S$ is the set of goal states.

Notice that there are two forms of nondeterminism:

- set of initial states (uncertainty about initial conditions);
- transition relation (nondeterministic action executions).

The solutions of a planning problem satisfy a **reachability requirement**: a solution plan is a state-action table that, starting at any state in I, allows one to reach a state in G.

In fact, the reachability requirement can be met in different ways.

Solving the planning problem: the deterministic case

Let $D = (P, S, A, R)$ be a planning domain, (D, I, G) be a planning problem for D, SA be a **deterministic state-action table** for D, and $K = (Q, T)$ be the execution structure induced by SA from I.

We distinguish among weak, strong, and strong cyclic deterministic solutions to the planning problem.

- **Weak** solutions: SA is a weak solution if, for any state in I, some terminal state of K belonging to G can be reached.
- **Strong** solutions: SA is a strong solution if K is acyclic and all terminal states of K are in G.
- **Strong cyclic** solutions: SA is a strong cyclic solution to the planning problem if from any state in Q some terminal state of K is reachable and all the terminal states of K are in G (they guarantee the achievement of the goal "under fairness").

About strong cyclic solutions

Strong cyclic solutions capture the idea of **"acceptable" trial-and-error strategies**: all their partial executions can be extended to obtain a finite execution path whose terminal state is a goal state.

However, they can produce executions that loop forever.

This happens when an infinite sequence of "bad choices" is done, that is, all the infinite execution paths are "**unfair**" because they eventually enter loops where some actions are executed infinitely often in some states where alternative actions leading to the goal were also enabled (and never executed).

Solving the planning problem: the nondeterministic case

Let SA be a nondeterministic state-action table SA. A **determinization** of SA is any deterministic state-action table $SA' \subseteq SA$ such that $\{s : (s, a) \in SA'\} = \{s : (s, a) \in SA\}$.

A **nondeterministic state-action** table SA is a weak (resp., strong, strong cyclic) solution to a planning problem if all the determinizations of SA are.

Strong solutions to a planning problem are a subset of strong cyclic solutions which are in turn a subset of weak solutions.

Direct algorithms for weak, strong, and strong cyclic planning have been proposed in the literature (Weak, strong, and strong cyclic planning via symbolic model checking, by A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, Artificial Intelligence, Volume 147(1–2), July 2003, pages 35-84).

The main technical ingredient: weak and strong preimages

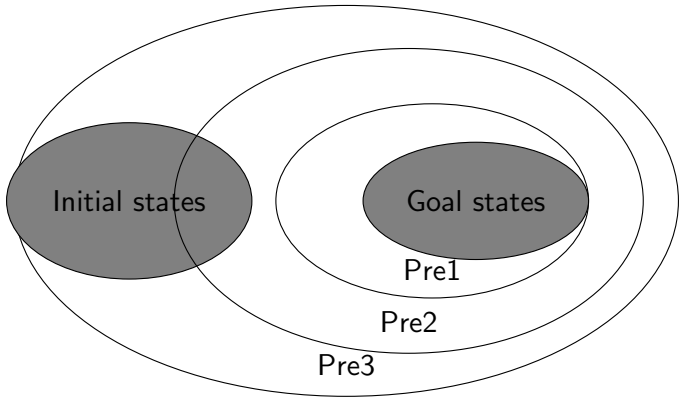

Figure: Backward breadth-first search

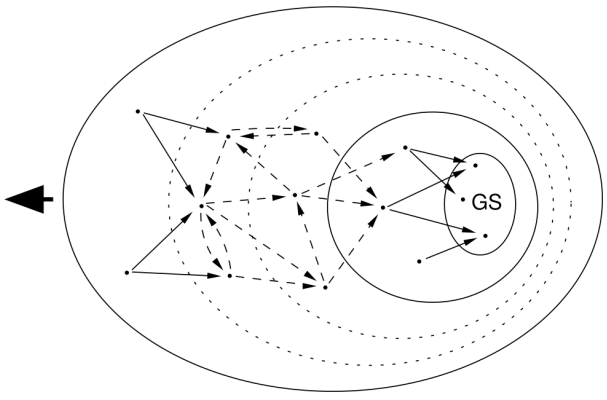

Figure: Strong preimage computation

A symbolic representation of a *planning domain*: states

The **symbolic representation** of a planning domain $D = (P, S, A, R)$ is as follows.

First, we associate a Boolean variable with each **element of** P.

The **set of states** S is represented by means of a vector of variables $\vec{x}$.

A **state** $s \in S$ is specified by a formula $\xi(s)$ consisting of the conjunction of the variables which are true in s and of the negation of those which are false in s.

A **set of states** $Q \subseteq S$ is thus represented by the formula:

$$\xi(Q) = \bigvee_{s \in Q} \xi(s)$$

A symbolic representation of a *planning domain*: actions

Actions are represented by means of another set of Boolean variables (action variables).

We can use a distinct action variable for each action in A. This allows for the representation of concurrent actions. In case no concurrent actions are allowed, a mutual exclusion constraint must be imposed.

In case concurrent actions are excluded, it is possible to use only $\lceil log|A| \rceil$ action variables $\vec{\alpha}$, where each assignment to the action variables denotes a specific action to be executed (the mutual exclusion constraint is not necessary).

A symbolic representation of a *planning domain*: transitions

Transitions are triples $(state, action, state)$. A third vector $\vec{x}'$ of propositional variables, called **next state variables**, is introduced. We require $\vec{x}$ and $\vec{x}'$ to have the same number of variables, and the variables in the same positions to correspond.

We make use of $\xi'(.)$ (the obvious adaptation of $\xi(.)$) to specify the next state.

A transition is represented by an assignment to $\vec{x}$, $\vec{\alpha}$, and $\vec{x}'$.

Formulas that represent the states of the domain, the transition relation, the initial states, and the goal states by $S(\vec{x})$, $R(\vec{x}, \vec{\alpha}, \vec{x}')$, $I(\vec{x})$, and $G(\vec{x})$, respectively.

A symbolic representation of a plan

The machinery for the symbolic representation of planning domains can be used to represent and manipulate symbolically the other elements of the planning algorithms, in particular the state-action tables (**plans**).

Let us denote by $SA(\vec{x}, \vec{\alpha})$ the formula corresponding to the state-action table SA.

- States of the state/action table:
  $\textsc{StatesOf}(SA) = \exists\vec{\alpha}(SA(\vec{x}, \vec{\alpha}))$
- Actions of the state/action table associated with a state s:
  $\textsc{ActionsOf}(SA) = \exists\vec{x}(SA(\vec{x}, \vec{\alpha}))$

A symbolic representation of planning algorithms

The **planning algorithms** can be expressed in terms of transformations over propositional formulas.

The basic steps of the algorithms are the following **preimage operations** that, rather than returning set of states, construct **state-action tables**.

- Weak Preimage:
  $\exists \vec{x}'(R(\vec{x}, \vec{\alpha}, \vec{x}') \wedge Q(\vec{x}'))$
- Strong Preimage:
  $\forall \vec{x}'(R(\vec{x}, \alpha, \vec{x}') \rightarrow Q(\vec{x}')) \wedge \textsc{appl}(\vec{x}, \vec{\alpha})$

where the actions enabled at a given state are expressed as follows:
$\textsc{appl}(\vec{x}, \vec{\alpha}) = \exists \vec{x}'(R(\vec{x}, \vec{\alpha}, \vec{x}'))$

The language $\mathcal{AR}$

One of the first **action description languages** proposed in the literature to model planning problems in nondeterministic domains is $\mathcal{AR}$.

- A CAUSES P IF Q: in any state where Q holds, once A has been executed, P holds
- A POSSIBLY CHANGES F IF Q: if A is executed in a state where Q holds, the value of F may change
- ALWAYS P: P holds in all states
- INITIALLY P: P holds in all initial states
- GOAL G: G holds in all final states

Formalization of the running example in $\mathcal{AR}$

Algorithm 1 A formalization of the running example

1: transportation CAUSES traffic light IF station
2: transportation CAUSES airport IF traffic light $\wedge$ light color $=$ green
3: transportation CAUSES $\bot$ IF $\neg$(station $\vee$ (traffic light $\wedge$ light color $=$ green))
4: wait CAUSES $\bot$ IF $\neg$ traffic light
5: ALWAYS station $\leftrightarrow \neg$(traffic light $\vee$ airport)
6: ALWAYS traffic light $\leftrightarrow \neg$(station $\vee$ airport)
7: ALWAYS airport $\leftrightarrow \neg$(traffic light $\vee$ station)
8: INITIALLY station
9: GOAL airport

A symbolic representation of states in $\mathcal{AR}$

States in $\mathcal{AR}$.

States in $\mathcal{AR}$ are represented by those valuations that satisfy ALWAYS P.

$$State = \bigwedge_{\text{ALWAYS } P} P$$

Initial and goal states must satisfy, in addition, INITIALLY P and GOAL G, respectively.

$$Init = State \wedge \bigwedge_{\text{INITIALLY } P} P$$

$$Goal = State \wedge \bigwedge_{\text{GOAL } G} G$$

A symbolic representation of actions in $\mathcal{AR}$

Actions in $\mathcal{AR}$. Actions are associated with a set of variables, called action variables, that are true if and only if the corresponding action is executed.

To model the effects of the execution of an action (the countepart of the $\mathcal{AR}$ statement A CAUSES P IF Q), we need to force the updated values of fluents to define valid states where P holds if Q holds in the current state.

$$Res^0 = State \wedge State[F_1/F_1', \ldots, F_n/F_n'] \wedge \bigwedge_{A \text{ CAUSES } P \text{ IF } Q} P[F_1/F_1', \ldots, F_n/F_n'] \subseteq (A \wedge Q) \quad (1)$$

In addition, we must constrain inertial fluents $F_1, \ldots, F_m$, with $m \leq n$, to be possibly affected only by the execution of the action or by a condition of the form A POSSIBLY CHANGES F IF Q.