Decidability and Complexity of Action-based Temporal Planning over Dense Time

Nicola Gigante[1] Andrea Micheli[2] Angelo Montanari[3] Enrico Scala[4]

[1]Free University of Bozen-Bolzano [2]Fondazione Bruno Kessler, Trento
[3]University of Udine [4]University of Brescia

TEMPORAL PLANNING

An action-based temporal planning problem (as in PDDL 2.1) $\mathcal{P} = \langle P, A, I, G \rangle$ is made of:

- a set of fluents P
- a set of durative actions A
- the initial state I
- the goal condition G

Finding a solution plan is EXPSPACE-complete on discrete time [1]. What about dense time?

PROBLEM INTERPRETATIONS

The PDDL 2.1 specification [2] can be ambiguous:

- self-overlap: are different instances of the same action allowed to overlap with themselves?
- non-zero/ε-separation: are mutex events required to be separated simply by a non-zero amount of time, or by a given ε value?

We find the computational complexity of the problem in all the four cases.

REFERENCES

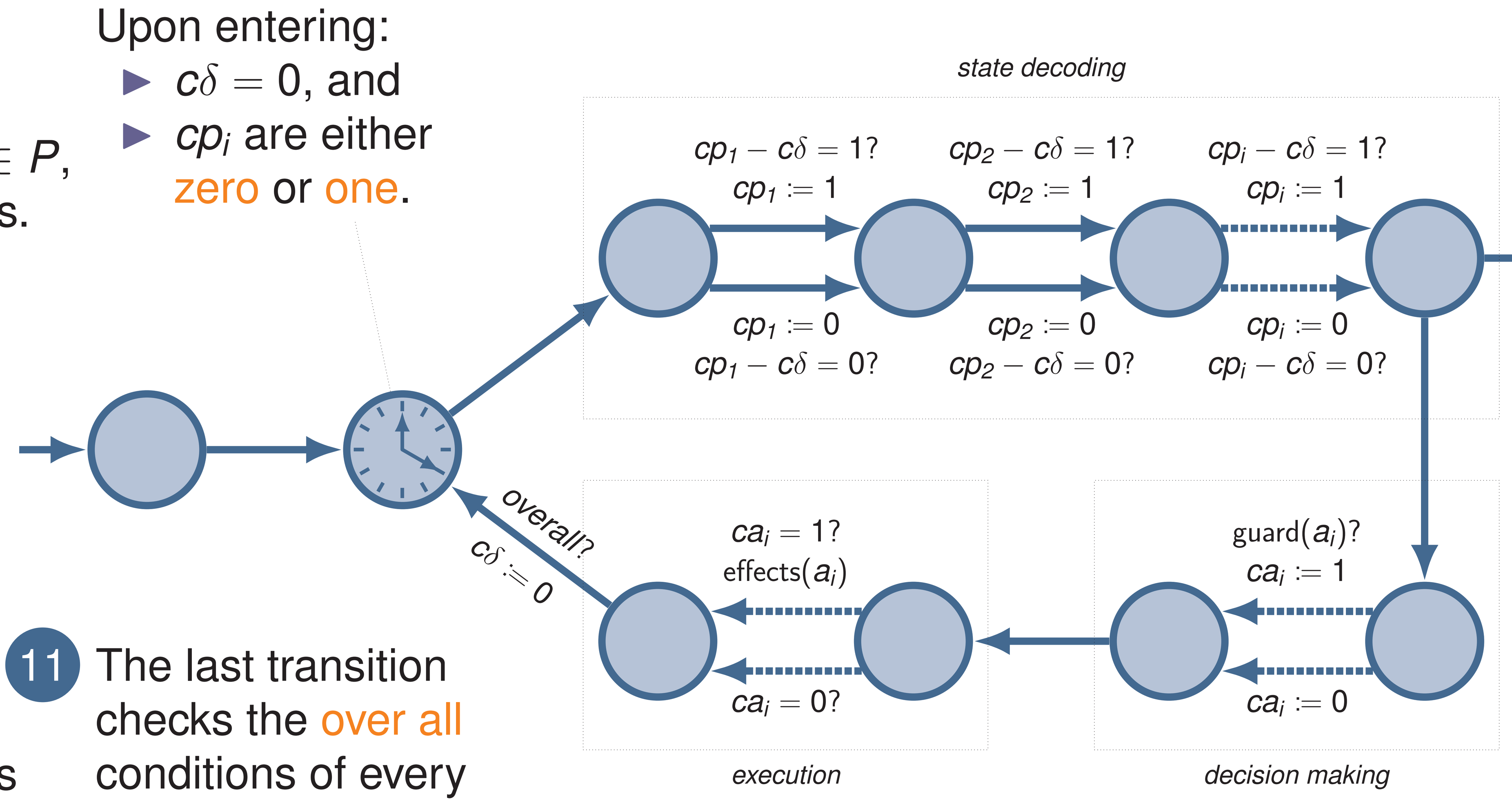
[1] J. Rintanen Complexity of Concurrent Temporal Planning ICAPS 2007

[2] M. Fox and D. Long PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains J. Artif. Intell. Res. vol. 20, 2003

[3] R. Alur and D. L. Dill A Theory of Timed Automata Theor. Comput. Sci. vol. 126, 1994

[4] Peter van Emde Boas The Convenience of Tilings Complexity, Logic, and Recursion Theory vol. 187, 1997

1. We encode temporal planning problems into timed automata [3].
2. We use a few clocks:
   - a global clock $c\delta$,
   - a clock cp_i for all $p_i \in P$,
   - some auxiliary clocks.
3. The automaton makes a loop through these locations for each event of the temporal plan.
4. Time flows only in the main location. The others are urgent. Upon entering:
   - $c\delta = 0$, and
   - cp_i are either zero or one.
5. With this invariant, the difference $cp_i - c\delta$ will always be either zero or one.
6. The state decoding phase can then restore the invariant.
7. From this point, the cp_i clocks can be used as boolean variables.



8. For example, the goal condition can be expressed as a simple conjunction.

$$goal \equiv \bigwedge_{p_i \in G} cp_i = 1$$

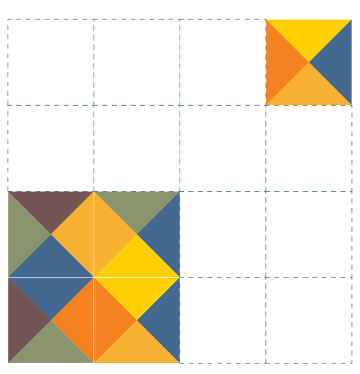
If the transition is enabled the plan is accepted.

9. The decision making phase choose the events to run at the current time. The guard enforces the semantics:
   - the event's conditions hold
   - action durations are valid
   - mutex events are separated
   - both non-zero separation and ε-separation are supported here
10. The execution phase applies the effects of the previously selected events.
11. The last transition checks the over all conditions of every running action.
12. A word is accepted iff a solution plan exists.
    - Reachability in TAs is PSPACE-complete.
    - Hardness comes from classical planning.
13. The problem is PSPACE-complete.

	ε-separation	non-zero separation
self-overlap forbidden	PSPACE-complete	PSPACE-complete
self-overlap allowed	EXPSPACE-complete	undecidable

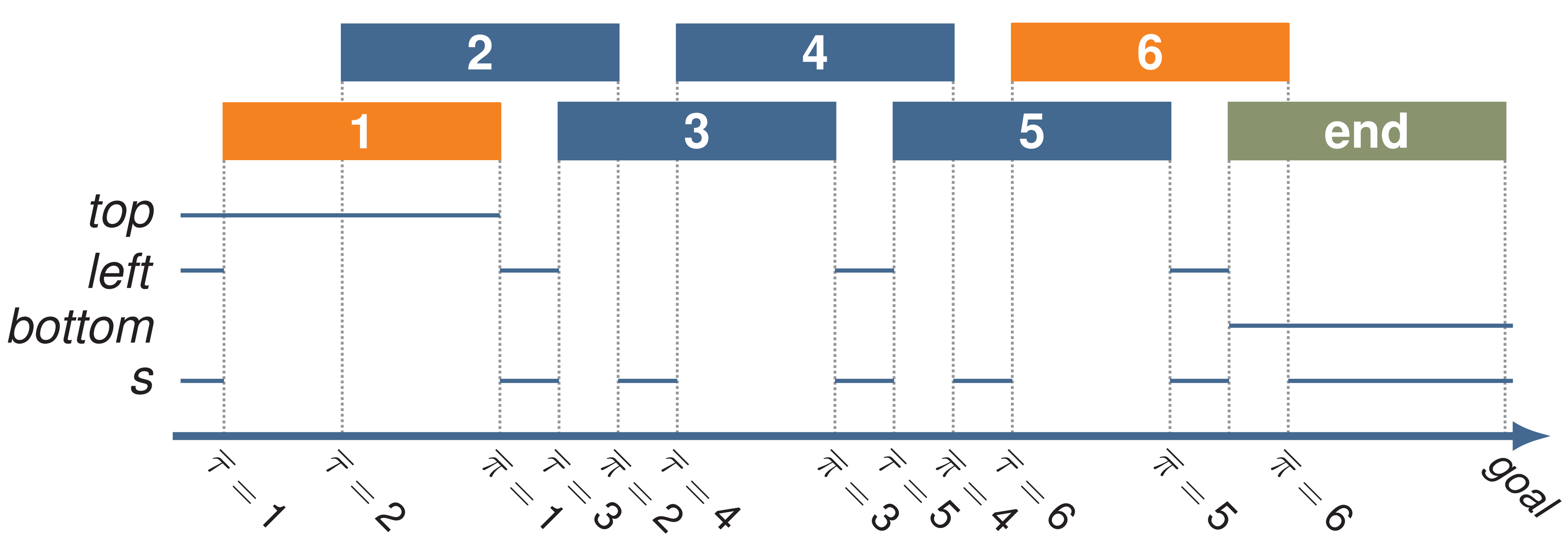
The tiling problem [4] asks to tile a rectangle using a given set of tiles whose colors on the sides have to match.

The general tiling problem is undecidable.

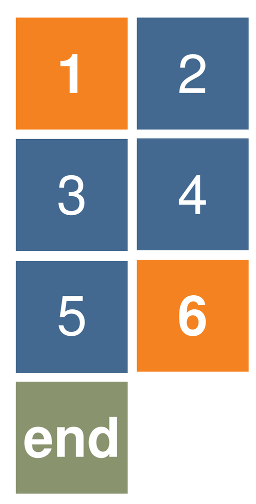


We reduce tiling to planning.
A plan encodes a tiling in a row-major layout:

- fluents $\overline{\tau} = \{\tau_0, \tau_1, \ldots\}$ encode the current tile,
- fluents $\overline{\pi} = \{\pi_0, \pi_1, \ldots\}$ encode the tile at the previous row,
- auxiliary fluents *top*, *bottom*, and *left* mark specific locations of the tiling,
- fluent s strictly alternates start and end events.



encoding of a tiling as a temporal plan



example tiling

The encoding assumes non-zero separation.

- If a side of the tiling is bounded by n, the tiling problem becomes EXPSPACE-complete.
- The exponentially bounded corridor tiling problem can be reduced to the planning problem with ε-separation.