Hardness of Chosen Length Planning Games and Regular Fixed Methods FOND HTN Planning

P. Maurice Dekker, Gregor Behnke

ILLC, Universiteit van Amsterdam, The Netherlands
p.m.dekker@uva.nl, g.behnke@uva.nl

Abstract

We introduce a new version of general game-playing in which one of the players chooses the length of the game up front. Consider a classical planning problem and two players who take turns applying actions. Player 1 wins iff the goal is true after a predetermined number of moves has been made. Is there a number r such that player 1 has a winning strategy for the game of length r? We show that this problem is EXPSPACE-complete. Moreover, we show that the problem is equivalent to the plan existence problem for a class of fully observable non-deterministic hierarchical task network planning problems under the solution concept with fixed methods, which was introduced in prior work. This class consists of all regular loop-unrolling problems, where a problem is loop-unrolling if it has at most one compound task name and at most two methods. As a corollary, we obtain hardness for regular problems, solving an open problem.

Introduction

The core of planning research focuses on the simplistic setting of classical planning. It lacks the expressiveness to capture more complex and thus more realistic settings. Classical planning assumes that the execution of actions is deterministic – which implicitly means the actor is in full control of the environment. This is not the case in certain applications. Vehicles can encounter traffic lights. If they happen to be red, they should execute a wait action. This leads to the notion of Fully Observable Non-Deterministic (FOND) planning.

We introduce the formal notion of planning games, interpreting non-determinism as a second player that acts against the first player controlled by the planner. Existing work (Muise et al. 2016; Camacho et al. 2018; Cimatti et al. 2016) drew similar connections between planning and games. The new twist in this paper is that the first player chooses the game length before the start of the game. In FOND, this means the planner must announce the length L of their plan before it is actually executed. The plan is only valid if a goal state is reached after exactly L many steps, regardless of non-determinism.

Suppose a truck driver needs to deliver a package. In our new setting, the driver has to announce when she will deliver the package before she heads out. This happens e.g. if the target location lacks parking space and the receiver lacks time. The driver communicates the delivery time in advance, so the receiver can make sure to be present. Then the truck must neither arrive early (as it cannot park and must leave immediately after arriving), nor late (forcing the receiver to wait). In Fig. 1, the driver (while at the starting location A) promises to arrive at the target location E at time 4. If she encounters a red traffic light at B, she must perform a wait action to wait for the green light, but if the light is already green, she must proceed immediately to C. If the traffic light was red, she now takes the direct route from C to E, but the longer route via D if it was green. This to compensate for the time not spent waiting at the red light at B.

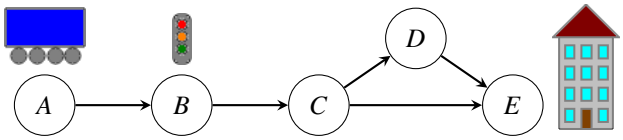

Figure 1: A fully observable non-deterministic planning problem. The traffic light is non-deterministic.

An example from the perspective of games is the variant of Chess in which White states the number of moves at the start of the game and players are allowed to move into check but cannot capture kings. Players can even continue moving after a player gets checkmated. White only wins the game if White's last move checkmates Black.

As our main result (Thm. 1), we prove that solving these "chosen length" planning games is EXPSPACE-complete, and thus harder than general FOND planning, which is only EXPTIME-complete (Rintanen 2004). As Chess is EXPTIME-complete (Fraenkel and Lichtenstein 1981), we conjecture that the variant from the previous paragraph is EXPSPACE-complete.

Secondly, we show that this new type of planning games and FOND planning relates to FOND planning with Hierarchical Task Networks (HTNs). HTN planning extends classical planning with compound tasks – which cannot be executed directly. The planner must apply decomposition methods that refine compound tasks into more concrete partially-ordered sets of other tasks. In previous work, Chen and Bercher (2021) introduced a formalism for FOND HTN

planning and studied its computational complexity. They left several open questions. With a reduction to planning games, we solve one of them: strong FOND HTN planning for regular HTNs is EXPSPACE-complete.

Planning Games

A *STRIPS planning problem* is a quadruple $\mathfrak{P} = (P, A, I, G)$ where:

- P is a finite set of (grounded) *(propositional) variables*;
- $A \subseteq \mathscr{P}(P)^4$ is a finite set of *actions*;
- $I \subseteq P$ is the *initial state*;
- $G \subseteq P$ is the *goal*.

A *(propositional) state* of $\mathfrak{P}$ is a subset of P. An action $a = (\Pi_+, \Pi_-, E_+, E_-) \in A$ consists of a set Π_+ of *positive preconditions*, a set Π_- of *negative preconditions*[1], a *positive effect* E_+, and a *negative effect* E_-. a is *applicable* in a state $S \subseteq P$ provided $\Pi_+ \subseteq S$ and $\Pi_- \cap S = \emptyset$. Then the result of applying a to S is $\gamma(S,a) = (S \setminus E_-) \cup E_+$.

For $r \in \mathbb{N}$, the *planning game* $\mathbb{G}(\mathfrak{P}, r)$ has two players, Éloise and Abélard, and lasts at most r turns. The game starts in the initial state I. Éloise goes first. The players take turns choosing an applicable action in A to apply to the current state. When no action in A is applicable to the current state because each action has a precondition that is violated, the player who is to move loses. Otherwise, after r actions have been executed, Éloise wins if the goal is satisfied in (viz.: is a subset of) the current state; otherwise Abélard wins.

By Zermelo's theorem (Zermelo 1913), one of the players has a winning strategy.

$\checkmark = (\emptyset, \emptyset, \emptyset, \emptyset)$ denotes the empty action. Note that players are allowed to pass a turn in $\mathbb{G}(\mathfrak{P}, r)$ if $\checkmark \in A$. The symbol $\sqcup$ denotes the union of disjoint sets.

Example 1. *Let $\mathfrak{P} = (P, A, I, G)$ be a STRIPS planning problem such that $|P| = 100$,*

$$A = \big\{(\emptyset, \emptyset, P', \emptyset) : P' \in \mathscr{P}(P) \text{ s.t. } |P'| = 2\big\} \\ \sqcup \big\{(\emptyset, \emptyset, \emptyset, P') : P' \in \mathscr{P}(P) \text{ s.t. } |P'| = 1\big\},$$

and $I = \emptyset$. If $|G| = 99$, Éloise has a winning strategy for $\mathbb{G}(\mathfrak{P}, r)$ iff $r = 195$ or $r \geq 197$. If $G = P$, Éloise has a winning strategy for $\mathbb{G}(\mathfrak{P}, r)$ iff $r \geq 197$ and r is odd.

The Game Description Language (Genesereth, Love, and Pell 2005) can efficiently formulate planning games. However, the choice of game length can only be encoded up to an exponential limit (in basic GDL).

Transformations of Planning Games

The following lemma introduces $\checkmark$ to a planning problem while preserving the outcomes of some of the planning games. The section on HTN planning will use item 1 (or 3).

Lemma 1. *Let $\mathfrak{P}$ be a STRIPS planning problem. Then:*

1. *We can compute in polynomial time a STRIPS planning problem $\mathfrak{P}'$ that has $\checkmark$ as an action such that for all even numbers $r \in \mathbb{N}$, the games $\mathbb{G}(\mathfrak{P}, r)$ and $\mathbb{G}(\mathfrak{P}', r)$ are won by the same player.*
2. *As 1 but with "odd" instead of "even".*
3. *We can compute in polynomial time a STRIPS planning problem $\mathfrak{P}'$ that has an action without preconditions such that for all $r \in \mathbb{N}$, the games $\mathbb{G}(\mathfrak{P}, r)$ and $\mathbb{G}(\mathfrak{P}', r)$ are won by the same player.*

Proof. We design the new planning problem in 1 so that playing $\checkmark$ makes you lose. If r is even such that Éloise wins $\mathbb{G}(\mathfrak{P}, r)$, perhaps only Abélard's last move makes the goal true. Since Abélard will be able to pass his last turn in $\mathbb{G}(\mathfrak{P}', r)$, we have to help Éloise satisfy the goal earlier.

Write $\mathfrak{P} = (P, A, I, G)$. Let

$$P' = P \sqcup \{\exists, \star, g\}.$$

Intuitively, $\exists$ says Éloise is to move, $\star$ says actions in A may be used, and g says Éloise threatens to win. Define a set A'_a of variants of an action $a = (\Pi_+, \Pi_-, E_+, E_-) \in A$ as follows. In any case put

$$\big(\Pi_+ \sqcup \{\star, \exists\}, \Pi_-, E_+ \sqcup \{g\}, E_- \sqcup \{\exists\}\big) \in A'_a \qquad (1)$$

and

$$\big(\Pi_+ \sqcup \{\star\}, \Pi_- \sqcup \{\exists\}, E_+ \sqcup \{\exists\}, E_-\big) \in A'_a.$$

If $E_- \cap G \neq \emptyset$, also put

$$\big(\Pi_+ \sqcup \{\star\}, \Pi_- \sqcup \{\exists\}, E_+ \sqcup \{\exists\}, E_- \sqcup \{g\}\big) \in A'_a. \qquad (2)$$

Otherwise, for each $p \in G \setminus (\Pi_+ \cup E_+)$ put

$$\big(\Pi_+ \sqcup \{\star\}, \Pi_- \cup \{p, \exists\}, E_+ \sqcup \{\exists\}, E_- \sqcup \{g\}\big) \in A'_a. \qquad (3)$$

The actions in (1) are intended for Éloise; the other actions in A'_a are intended for Abélard. Let

$$A' = \Big\{\checkmark, \big(\{\star, \exists\}, \emptyset, \emptyset, \{\star, g\}\big), \big(\{\star\}, \{\exists\}, \{g\}, \{\star\}\big)\Big\} \sqcup \bigsqcup_{a \in A} A'_a.$$

The two $\star$-deleting actions will refute $\checkmark$. Finally define

$$I' = \begin{cases} I \sqcup \{\exists, \star, g\} & (G \subseteq I) \\ I \sqcup \{\exists, \star\} & (G \not\subseteq I) \end{cases}$$

and $G' = \{g\}$. Then we claim that $\mathfrak{P}' = (P', A', I', G')$ is as desired.

As long as only actions in $\bigsqcup_{a \in A} A'_a$ are played, $\exists$ is true if Éloise is to move and false if Abélard is to move. Note that all actions in A' except $\checkmark$ require $\star$. Hence if Éloise is to move and $\exists$ is false, she can win by deleting $\star$ and adding g. Similarly, if Abélard is to move and $\exists$ is true, he can win by deleting $\star$ and g. Hence, up until Abélard's last move, only actions in $\bigsqcup_{a \in A} A'_a$ will be played, as playing $\checkmark$ allows the other player to win. In particular, Éloise's last move in (1) adds g, so Abélard wants to delete it with his last move. He can do this iff he can either apply an action in (2) that deletes some goal variable in G or some variable in G is false and he can apply an action in (3) that does not add it.

For 2 and 3, let $G' = G \sqcup \{g\}$ and let the $\star$-deleting action that adds g also add G; replace $\checkmark$ by $(\emptyset, \emptyset, G, \emptyset)$ for 3. □

[1] It is well-known that negative preconditions can be compiled away in linear time; cf. (Gazen and Knoblock 1997, sect. 2.6).

The HTN section will apply item 1 of the next lemma.

Lemma 2. *Let $\mathfrak{P}$ be a STRIPS planning problem. Then:*

1. *We can compute a STRIPS planning problem $\mathfrak{P}'$ in polynomial time such that for all $r \in \mathbb{N}$, the games $\mathbb{G}(\mathfrak{P}, r)$ and $\mathbb{G}(\mathfrak{P}', 2r)$ are won by the same player.*
2. *As 1 but with $2r+1$ instead of $2r$.*

Proof. We prove 1 by inserting dummy actions. Four turns of the new game will model two turns of the old game, the second and third of each cycle being dummy actions.

Write $\mathfrak{P} = (P, A, I, G)$. Let

$$P' = P \sqcup \{\exists, w\}.$$

Intuitively, $\exists$ says Éloise is to move and w says the players have to wait. Define a set A'_a of variants of an action $a = (\Pi_+, \Pi_-, E_+, E_-) \in A$ by

$$A'_a = \Big\{ \big(\Pi_+ \sqcup \{\exists\}, \Pi_- \sqcup \{w\}, E_+ \sqcup \{w\}, E_- \sqcup \{\exists\}\big), \quad (4)$$

$$\big(\Pi_+, \Pi_- \sqcup \{\exists, w\}, E_+, E_- \sqcup \{\exists\}\big) \Big\}. \quad (5)$$

Let

$$A' = \Big\{ \big(\{w\}, \{\exists\}, \{\exists\}, \emptyset\big), \big(\{\exists, w\}, \emptyset, \emptyset, \{\exists, w\}\big) \Big\} \sqcup \bigsqcup_{a \in A} A'_a.$$

Finally define $I' = I \sqcup \{\exists\}$ and $G' = G$. Then we claim that $\mathfrak{P}' = (P', A', I', G')$ is as desired.

Éloise starts with an action of the form in (4) (if $r \geq 1$). Then $\exists$ is false and w is true, so Abélard must do $\big(\{w\}, \{\exists\}, \{\exists\}, \emptyset\big)$. Then $\exists$ and w are true, so Éloise must do $\big(\{\exists, w\}, \emptyset, \emptyset, \{\exists, w\}\big)$ (if $r \geq 2$). Afterwards, $\exists$ and w are false and Abélard chooses an action of the form (5), making $\exists$ true again. Éloise again proceeds with an action from (4) (provided $r \geq 3$), and so on.

For even r, a state S after r turns of $\mathfrak{P}$ (so with Éloise to move) corresponds to the state $S \sqcup \{\exists\}$ after $2r$ turns of $\mathfrak{P}'$. For odd r, a state S after r turns of $\mathfrak{P}$ (with Abélard to move) corresponds to the state $S \sqcup \{\exists, w\}$ after $2r$ turns of $\mathfrak{P}'$.

For 2, let $G' = G \sqcup \{g, g'\}$ and introduce the actions $a_{\text{end}} = \big(\{\exists\}, \{w, g'\}, \{g, g'\}, \emptyset\big)$ (Éloise's last move) and $\big(\{g\}, \emptyset, \emptyset, \{g\}\big)$ (refuting a_{end} if Éloise plays it earlier). □

The hardness proof in the next section uses the following:

Lemma 3. *Let $\mathfrak{P}$ be a STRIPS planning problem and $m \in \mathbb{N}_{>0}$ a number in unary encoding. Then we can compute a STRIPS planning problem $\mathfrak{P}'$ in polynomial time such that the following are equivalent for all $r \in \mathbb{N}$:*

1. *Éloise wins $\mathbb{G}(\mathfrak{P}, r)$ and $r \cong 0 \mod m$.*
2. *Éloise wins $\mathbb{G}(\mathfrak{P}', r)$.*

Proof. Write $\mathfrak{P} = (P, A, I, G)$. Problem $\mathfrak{P}' = (P', A', I', G')$ features a cyclic counter of length m. Let

$$P' = P \sqcup \{p_k : k \in \mathbb{Z}/m\mathbb{Z}\},$$

$$A' = \Big\{ \big(\Pi_+ \sqcup \{p_k\}, \Pi_-, E_+ \sqcup \{p_{k+1}\}, E_- \sqcup \{p_k\}\big) :$$

$$(\Pi_+, \Pi_-, E_+, E_-) \in A \text{ and } k \in \mathbb{Z}/m\mathbb{Z} \Big\},$$

$I' = I \sqcup \{p_0\}$, and $G' = G \sqcup \{p_0\}$. □

Algorithm 1: Search for winning even length of the planning game

Data: STRIPS planning problem $\mathfrak{P} = (P, A, I, G)$
Result: $r \in \mathbb{N}$ such that Éloise has a winning strategy for $\mathbb{G}(\mathfrak{P}, 2r)$, or NIL if such r does not exist

```
1  Let 𝕊 = {S ∈ 𝒫(P) : G ⊆ S}.
2  forall r = 0, ..., 2^(2^|P|) − 1 do
3      if I ∈ 𝕊 then
4          return r
5      Let 𝕊' = ∅.
6      for S ∈ 𝒫(P) do
7          for a ∈ A do
8              if a is applicable in S then
9                  for b ∈ A do
10                     if b is applicable in γ(S, a) then
11                         if γ(γ(S, a), b) ∉ 𝕊 then
12                             continue with the loop on
                                   Line 7
13                 Add S to 𝕊'.
14     Let 𝕊 = 𝕊'.
15 return NIL
```

Generalizations of this lemma exist, e.g. with m in binary encoding.

Main Result

Theorem 1. *It is EXPSPACE-complete to decide given a STRIPS planning problem $\mathfrak{P}$ whether there exists $r \in \mathbb{N}$ such that Éloise has a winning strategy for the r turns planning game $\mathbb{G}(\mathfrak{P}, r)$. This remains true when confining attention to problems $\mathfrak{P}$ with action ✓, and it does not matter whether r is restricted to be even or odd.*

We introduce a bit of notation for the proof: if $\mathfrak{P} = (P, A, I, G)$ and $S \subseteq P$ write $\mathfrak{P}_S = (P, A, S, G)$.

Membership

Alg. 1 requires $O(2^{|P|})$ memory and searches for a winning even length. We derived it from the algorithm given by Chen and Bercher (2021, Thm. 5.3) using the proof of Thm. 5 below.

Line 1 sets $\mathbb{S}$ to the set of all states S such that Éloise wins $\mathbb{G}(\mathfrak{P}_S, 0)$. During the loop on Line 2, it is maintained that $\mathbb{S}$ is the set of all states S such that Éloise wins $\mathbb{G}(\mathfrak{P}_S, 2r)$. Then whenever $I \in \mathbb{S}$, we have found an even length of the game that is winning for Éloise (Lines 3-4).

To calculate the next value of $\mathbb{S}$, initialize it to $\emptyset$ in Line 5 while keeping the old set $\mathbb{S}$ in memory until the new one is ready (Line 14). We take a state S (Line 6) and wonder whether Éloise can win $\mathbb{G}(\mathfrak{P}_S, 2r+2)$, meaning she can choose an action a (Line 7) that is applicable in S (Line 8) such that no matter Abélard's reply (Line 9), the resulting state will be in the old set $\mathbb{S}$. Indeed, if Abélard has a legal

(Line 10) reply to a leading out of $\mathbb{S}$ (Line 11), we reach Line 12 discarding the action a; if Abélard has no such option, Line 13 adds S to the new $\mathbb{S}$.

If the same set $\mathbb{S}$ occurs twice during the main loop, there is no r such that Éloise wins $\mathbb{G}(\mathfrak{P}, 2r)$. We cannot keep a history of the sets $\mathbb{S}$ as this may require doubly exponential space, but we know that the same set $\mathbb{S}$ must occur twice if we iterate $2^{2^{|P|}}$ times, so at that point we return a negative answer in Line 15. We can iterate r until this limit by encoding it in binary using a singly exponential amount of space.

To search for a winning odd length $2r+1$ instead, change the initialization of $\mathbb{S}$ in Line 1 to the set of all states $S \subseteq P$ such that $G \subseteq \gamma(S,a)$ for some $a \in A$.

Hardness

We show that every exponential space Turing machine T can be modelled by a planning game. The basic idea is that the set of all possible states after any given number of turns in Éloise's strategy encodes a configuration of T. But, similarly to Alg. 1, time is reversed: the goal says T is just starting and the initial state says T is halting.

T works with only two symbols: the alphabet is $\Sigma = \{\bigcirc, \odot\}$. A transition of T always consists of reading a symbol, writing a symbol, moving either left or right, and going to a new internal state. Moreover, T has only one halting state. We only run T on the constant $\bigcirc$ input. The head of T starts at location 0. This machine is sufficiently general (Arora and Barak 2007, sect. 1.2). Let $n \in \mathbb{N}$ be the size of T (w.r.t. any reasonable encoding). Assume the head of T remains in the interval $[0, 2^n)$.

We wish to compute a STRIPS planning problem $\mathfrak{P}'$ in time polynomial in n such that T halts iff there exists $r \in \mathbb{N}$ such that Éloise wins $\mathbb{G}(\mathfrak{P}', r)$. We shall relate r to the length of the run of T. We shall use $2n+4$ actions to model a single transition of T. Indeed, by Lemma 3 it suffices to construct $\mathfrak{P}$ such that for all $r \in \mathbb{N}$, machine T halts after exactly r transitions iff Éloise wins $\mathbb{G}\left(\mathfrak{P}, (2n+4)r\right)$.

Let Q be the set of internal states of T. Let $q_0 \in Q$ be the initial state of T and let $q_\omega \in Q$ be the halting state of T.

Each consecutive sequence of $2n+4$ turns of the planning game corresponds to a single regression step in the run of T. The turncounter from the proof of Lemma 3 can restrict which actions are applicable at which moments in this cycle. At the start of the planning game, machine T is in state q_ω.

Overview We first introduce $\mathfrak{P}$'s most important variables and, using them, state properties that we desire for $\mathfrak{P}$ (based on T). Next we describe the rules of the game, omitting the technical details on how to encode them in STRIPS using additional variables.[2] Then we prove $\mathfrak{P}$'s desired properties.

Variables Introduce the following set of variables for $\mathfrak{P}$:

$$\{d_i : i < n\} \sqcup \Sigma \sqcup \{h\} \sqcup Q.$$

The variables d_i encode the location of a tape cell of T. For $c < 2^n$, let $[c]$ be the set containing d_i for each $i < n$ such that digit i in the binary encoding of c is 1. h stands for "head".

[2] A total number of variables that is linear in n suffices.

Properties After every full cycle of the planning game, exactly one variable in the set $\Sigma \sqcup \{h\} \sqcup Q$ will be true. Further, we want to arrange that for all $t \in \mathbb{N}$ and for all $c < 2^n$:

tape For each $\sigma \in \Sigma$, Éloise wins $\mathbb{G}\left(\mathfrak{P}_{[c]\sqcup\{\sigma\}}, (2n+4)t\right)$ iff tape cell c of T contains symbol σ at time t.

head Éloise wins $\mathbb{G}\left(\mathfrak{P}_{[c]\sqcup\{h\}}, (2n+4)t\right)$ iff the head of T is at position c at time t.

state For all $q \in Q$, Éloise wins $\mathbb{G}\left(\mathfrak{P}_{[c]\sqcup\{q\}}, (2n+4)t\right)$ iff T is in state q at time t.

Éloise should lose all these games if T halts before time t (i.e.: tape, head, and machine vanish after T halts).

Initial State $\mathfrak{P}$'s initial state is $\{q_\omega\}$ (disregarding auxiliary variables). We do not care about T's final tape content.

Actions Each cycle of $2n+4$ turns proceeds as follows:

- If some $\sigma \in \Sigma$ is true, Éloise adds h and chooses:
  - *Write:* she deletes σ and adds a symbol $\sigma' \in \Sigma$ of choice and an internal state $q \in Q \setminus \{q_\omega\}$ of choice, under the restriction that T writes value σ when reading σ' in state q; or
  - *Preserve:* In this case, Abélard first deletes either σ or h. Then if h is still true, Éloise **changes** the number encoded by the variables d_i. I.e.: she flips at least one of the variables d_i and more if she wants to, but none of them twice. (With the help of additional variables this can be described by STRIPS over $n+1$ of Éloise's moves, viz. $2n+1$ turns.)
- If h is true, Éloise adds a symbol $\sigma \in \Sigma$ of choice and an internal state $q \in Q \setminus \{q_\omega\}$ of choice. If T moves the head to the left when reading σ in state q, she must increment the number encoded by the variables d_i. Otherwise she must decrement the number.
- If some $q \in Q$ is true, she deletes it. Then she adds h, a symbol $\sigma \in \Sigma$ of choice, and an internal state $q' \in Q \setminus \{q_\omega\}$ of choice, under the restriction that T transitions to state q when reading σ in state q'. Moreover, she chooses any location on the tape, viz. she can either add or delete each variable d_i. (Again, choosing the location is done over the course of $2n+1$ turns.)

In general, when Éloise chooses a symbol in Σ, we think of it as the symbol read by T at the previous moment in time. When she chooses an internal state in $Q \setminus \{q_\omega\}$, we think of it as the state the machine was in at the previous moment in time. When she chooses a number encoded by the variables d_i, we think of it as the position of the head of T at the previous moment in time.

In any case, at his last move of the cycle, Abélard has to delete all of $\Sigma \sqcup \{h\} \sqcup Q$ except one variable of choice.

All of this fits into $2n+4$ turns.

Goal At the end of the planning game, Éloise wins iff

$$\bigcirc \vee q_0 \vee (h \wedge \neg d_0 \wedge \cdots \wedge \neg d_{n-1}).$$

This can be compiled into STRIPS by creating several copies of all actions: ones that make the goal true and ones that make the goal false.

This completes the description of $\mathfrak{P}$. We next prove tape, head, and state for $\mathfrak{P}$. We proceed by induction on t.

Inductive Basis Consider $t = 0$ and $c < 2^n$. Then tape says that $\sigma = \bigcirc$ iff tape cell c of T contains symbol σ at time 0. head says that $[c] = \emptyset$ iff the head of T is at position c at time 0. state says that $q = q_0$ iff T is in state q at time 0. These statements were all among our assumptions.

Inductive Hypothesis Suppose tape, head, and state hold for t and for all $c < 2^n$.

Inductive Step Consider $t+1$ and let $c < 2^n$ be given.

- To prove tape for $t+1$, let σ be given and distinguish two cases:
  - At time t, the head of T is at location c. If Éloise chooses Preserve, Abélard can make sure the cycle ends in a propositional state of the form $[c'] \sqcup \{h\}$ for some $c' \in [0, 2^n) \setminus \{c\}$, and win by the inductive hypothesis on head. So Éloise chooses Write, and, by the inductive hypothesis, she can win iff she can choose σ' and q such that T writes σ when reading σ' in state q, cell c contains symbol σ' at time t, the head of T is at position c at time t, and T is in state q at time t.
  - At time t, the head of T is at a different location than c. If Éloise chooses Write, Abélard can make sure the cycle ends in a propositional state of the form $[c] \sqcup \{h\}$, and win by the inductive hypothesis on head. So Éloise chooses Preserve, and, by the inductive hypothesis on tape and head, she can win iff tape cell c contains symbol σ at time t and there is a $c' \in [0, 2^n) \setminus \{c\}$ such that the head of T is at c' at time t. Note that such c' does exist.
- For head, the inductive hypothesis implies that Éloise can win iff she can choose σ and q such that T is in state q at time t and either
  - cell $c+1$ contains symbol σ at time t, the head of T is at position $c+1$ at time t, and T moves the head to the left when reading σ in state q; or
  - cell $c-1$ contains symbol σ at time t, the head of T is at position $c-1$ at time t, and T moves the head to the right when reading σ in state q.
- For state, let q be given. The inductive hypothesis implies that Éloise can win iff she can choose σ, q', and c' such that T transitions to q when reading σ in state q', cell c' contains symbol σ at time t, the head of T is at position c' at time t, and T is in state q' at time t.

Conclusion tape, head, and state hold for all $t \in \mathbb{N}$ and $c < 2^n$. Consider state for q_ω. Then T halts after exactly r transitions iff Éloise wins $\mathbb{G}\left(\mathfrak{P}, (2n+4)r\right)$.

Example 2. *Consider the Turing machine T in Fig. 2. Its run is given in Table 1. If $n = 16$, each cycle comprises 36 turns. For transparency we instead use $n = 1$, as the variables $d_1, d_2, \ldots$ are redundant since the head stays in the interval $[0,2)$. The machine halts after three transitions, meaning Éloise can win $\mathbb{G}(\mathfrak{P}, 18)$.*

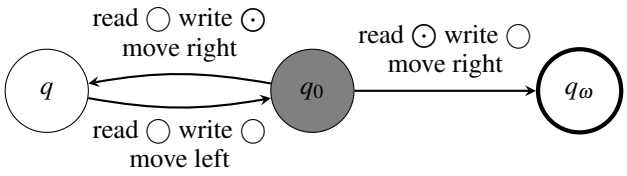

Figure 2: A Turing machine. The initial state has shading and the halting state has a thick border. The irrelevant specification on reading $\odot$ in state q is omitted.

Time	State	Cell 0	Cell 1
0	q_0	◯	
1	q	⊙	◯
2	q_0	⊙	
3	q_ω		◯

Table 1: The run of the Turing machine in Fig. 2. The position of the head is the shaded cell.

- *The first six turns (time 3 $\rightsquigarrow$ time 2): Éloise starts by deleting q_ω and adding h, $\odot$, and q_0. (This is allowed because T transitions to q_ω when reading $\odot$ in state q_0.) She deletes d_0, encoding location $0 \cdot 2^0 = 0$ on the tape. At the end of this cycle, it holds that*

$$(\odot \vee h \vee q_0) \wedge \neg d_0.$$

- *The next six turns (time 2 $\rightsquigarrow$ time 1):*
  - *If h is true, Éloise adds ◯, q, and d_0 (incrementing the encoded number to $1 \cdot 2^0 = 1$). (This is allowed because T moves the head to the left when reading ◯ in state q.)*
  - *If $\odot$ is true, Éloise adds h and chooses Preserve, and if Abélard then keeps h, she adds d_0, changing the location from 0 to 1.*
  - *Otherwise q_0 is true. Then Éloise deletes it and adds h, ◯, and q, and chooses location 1 on the tape.*

  At the end of this cycle, it holds that

$$(\bigcirc \vee \odot \vee h \vee q) \wedge (d_0 \leftrightarrow \neg\odot).$$

- *The final six turns (time 1 $\rightsquigarrow$ time 0):*
  - *If h is true, Éloise adds ◯ and q_0 and deletes d_0 (decrementing the encoded number).*
  - *If ◯ is true, Éloise adds h and chooses Preserve, and if Abélard then keeps h, she deletes d_0, changing the location from 1 to 0.*
  - *If $\odot$ is true, Éloise adds h and chooses Write, deleting $\odot$ and adding ◯ and q_0. (This is allowed because T writes $\odot$ when reading ◯ in state q_0.) In this case, d_0 was already false.*
  - *If q is true, Éloise deletes it and adds h, ◯, and q_0, and chooses location 0 on the tape.*

  At the end, it holds that

$$(\bigcirc \vee h \vee q_0) \wedge (d_0 \rightarrow \bigcirc),$$

so, since $\neg(\bigcirc \wedge h)$, the goal is achieved.

Éloise loses $\mathbb{G}(\mathfrak{P},6)$: in view of the goal she should add q_0 and ◯, but this is forbidden as T does not transition to q_ω when reading ◯ in state q_0.

Éloise loses $\mathbb{G}(\mathfrak{P},24)$: she has to choose either q_0 or q in the first cycle and state implies that she loses $\mathbb{G}(\mathfrak{P}_{P_c \sqcup \{q_0\}},18)$ and $\mathbb{G}(\mathfrak{P}_{P_c \sqcup \{q\}},18)$ for each $c = 0,1$.

Adding Passes If we naively add ✓ to this construction, Abélard easily wins by letting the turncounter end at a nonzero value. To prove the second statement in Thm. 1, we modify the construction to create a problem $\mathfrak{P}^{\checkmark}$ as follows.

Extend $\mathfrak{P}$'s cycle of $2n+4$ turns to a cycle of $2n+6$ turns. During the last two turns of the cycle, nothing happens. Éloise wins the new game if the turncounter is at 0 **or** $2n+5$. Add the action ✓. Moreover add a special variable $\star$ that is true in the initial state and is a precondition of every action of $\mathfrak{P}^{\checkmark}$ except ✓. Introduce $2n+6$ more actions: when the turncounter is at an even value, one can consume[3] $\star$ while making the goal false, and when the turncounter is at an odd value one can consume $\star$ while making the goal true. Then neither player ever has an incentive to play ✓ (except possibly on the very last turn of the game), as the other player will use one of the $\star$-consuming actions to secure a win. Hence the following are equivalent for all $r \in \mathbb{N}_{>0}$:

1. T halts after exactly r transitions.
2. Éloise wins $\mathbb{G}\left(\mathfrak{P}^{\checkmark},(2n+6)r+\varepsilon\right)$ for some $-1 \le \varepsilon \le 1$.
3. As 2 but with "all" instead of "some".

And Éloise loses $\mathbb{G}\left(\mathfrak{P}^{\checkmark},(2n+6)r+\varepsilon\right)$ for $2 \le \varepsilon \le 2n+4$.

FOND HTN Planning

We revisit a FOND version of HTN planning introduced by Chen and Bercher (2021).

A *task network* over a set N of *task names* is a triple $\mathfrak{t} = (T,\prec,\alpha)$ where (i) T is a finite set of *tasks*; (ii) $\prec$ is a strict partial order on T; and (iii) $\alpha : T \to N$ assigns a task name to each task in the network. Let TN_N be the set of all task networks over N. If $T' \subseteq T$, we define the *task subnetwork*

$$\mathfrak{t}|T' = \left(T', \prec \cap (T' \times T'), \alpha|T'\right) \in \mathrm{TN}_N.$$

For $t \in T$ we write $T \smallsetminus t = T \setminus \{t\}$. We write $\mathfrak{t} \smallsetminus t$ as a shorthand for $\mathfrak{t}|(T \smallsetminus t)$. An *embedding* $\phi : \mathfrak{t} \hookrightarrow \mathfrak{t}'$ of $\mathfrak{t} = (T,\prec,\alpha)$ into $\mathfrak{t}' = (T',\prec',\alpha')$ is an injection $\phi : T \hookrightarrow T'$ that preserves the partial order both ways and satisfies $\alpha' \circ \phi = \alpha$. An *isomorphism* is a bijective embedding. Let $\mathfrak{o} = (\emptyset,\emptyset,\emptyset)$ be the empty task network. A task network $\mathfrak{t} = (T,\prec,\alpha)$ is a *disjoint union* of a family $\{\mathfrak{t}_i : i \in I\}$ of task networks if there exist embeddings $\phi_i : \mathfrak{t}_i \hookrightarrow \mathfrak{t}$ whose images form a partition of T such that $\phi_i(t_i) \not\prec \phi_j(t_j)$ whenever $i,j \in I$ are distinct and t_i and t_j are tasks of $\mathfrak{t}_i$ and $\mathfrak{t}_j$ respectively.

A *FOND HTN problem* is a tuple $H = (F,C,O,M,\delta,\mathfrak{t}_0,S_0)$ where

- F, C, and O are finite sets of *(propositional) variables*, *compound task names*, and *primitive task names*, respectively;
- $M \subseteq C \times \mathrm{TN}_{C \sqcup O}$ is a finite set of *(decomposition) methods*;
- $\delta : O \to \mathscr{P}(F)^2 \times \mathscr{P}\left(\mathscr{P}(F)^2\right)$ is a *non-deterministic action mapping*;
- $\mathfrak{t}_0 \in \mathrm{TN}_{C \sqcup O}$ is an *initial task network*;
- $S_0 \subseteq F$ is an *initial propositional state*.

A *propositional state* of H is a subset of F. For better readability, we will adopt the following style guide: propositional variables are `typewriter blue`, compound task names **bold and brown**, primitive task names sans serif pink and decomposition methods green. A *compound* task is a task whose name is in C. A *primitive* task is a task whose name is in O. A task network over O is also called *primitive*. $\mathfrak{t}$ is a task network *of* H if $\mathfrak{t} = \mathfrak{t}_0$ or $(\mathbf{c},\mathfrak{t}) \in M$ for some $\mathbf{c} \in C$.

A decomposition method is applied to change a non-primitive task network into another task network. Suppose that $\mathfrak{t}_1 = (T_1,\prec_1,\alpha_1)$ and $\mathfrak{t} = (T,\prec,\alpha)$ are task networks, $t \in T_1$, and $\mu = \left(\alpha_1(t),\mathfrak{t}\right)$ is a method. When we create $\mathfrak{t}_2 = (T_2,\prec_2,\alpha_2)$ by replacing t by a copy of $\mathfrak{t}$, we write $\mathfrak{t}_1 \to_{t/\mu} \mathfrak{t}_2$. I.e.: there exists an embedding $\phi : \mathfrak{t} \hookrightarrow \mathfrak{t}_2$ such that (i) T_2 is the disjoint union of $T_1 \smallsetminus t$ and the image of ϕ and (ii) for all $t_1 \in T_1 \smallsetminus t$, $t' \in T$, and $* \in \{\prec,\succ\}$ it holds

$$t_1 *_1 t \iff t_1 *_2 \phi(t').$$

The task network $\mathfrak{t}_2$ exists and is unique up to isomorphism.

The action mapping δ associates with each pr $\in O$ a triple $\delta(\mathsf{pr}) = (\Pi_+,\Pi_-,\mathbb{E})$ consisting of a set Π_+ of *positive preconditions*, a set Π_- of *negative preconditions*, and a set $\mathbb{E}$ of *effects*. Each effect is a pair (E_+,E_-) consisting of a *positive effect* E_+ and a *negative effect* E_-. Now if propositional state $S \subseteq F$ satisfies $\Pi_+ \subseteq S$ and $\Pi_- \cap S = \emptyset$, we say pr is *applicable* in S and define

$$\Gamma(S,\mathsf{pr}) = \left\{(S \setminus E_-) \cup E_+ : (E_+,E_-) \in \mathbb{E}\right\}.$$

We say that a belief state $\mathbb{S} \subseteq \mathrm{TN}_O \times \mathscr{P}(F)$ is *solvable* if for all $\left(\mathfrak{t} = (T,\prec,\alpha),S\right) \in \mathbb{S}$, either $\mathfrak{t} = \mathfrak{o}$ or there exists a primitive $\prec$-minimal task $t \in T$ such that $\alpha(t)$ is applicable in S and the belief state

$$\left\{\left(\mathfrak{t} \smallsetminus t, S'\right) : S' \in \Gamma\left(S,\alpha(t)\right)\right\}$$

is, recursively, solvable. In this paper, we consider plans whose method applications are before task executions. I.e., the problem H is *solvable* if there exist $n \in \mathbb{N}$ and a sequence

$$\mathfrak{t}_0 \to_{t_0/\mu_0} \dots \to_{t_{n-1}/\mu_{n-1}} \mathfrak{t}_n \in \mathrm{TN}_O$$

of applications of methods $\mu_i \in M$ ending in a primitive task network such that $\left\{(\mathfrak{t}_n,S_0)\right\}$ is a solvable belief state.

Example 3. *Let* $F = \{$`var0`, `var1`$\}$, $C = \{$**comp**$\}$, $O = \{$pr0, pr1, pr2$\}$,

$$M = \left\{\mathit{stop} = (\mathbf{comp},\mathfrak{o}), \mathit{cont} = (\mathbf{comp},\mathfrak{t})\right\}$$

where $\mathfrak{t}$ *contains four unordered tasks with names* **comp**, pr0, pr1, *and* pr2,

$$\delta(\mathsf{pr0}) = \left(\{\texttt{var0}\},\emptyset,\left\{\left(\{\texttt{var1}\},\emptyset\right)\right\}\right)$$

(i.e. pr0 *requires* `var0` *and adds* `var1`*),*

$$\delta(\mathsf{pr1}) = \left(\{\texttt{var1}\},\emptyset,\left\{\left(\{\texttt{var0}\},\emptyset\right)\right\}\right),$$

[3] Consuming means requiring and deleting.

$$\delta(\mathit{pr2}) = \Big(\emptyset, \emptyset, \big\{(\{\mathtt{var0}\}, \emptyset), (\{\mathtt{var1}\}, \emptyset)\big\}\Big)$$

(i.e. pr2 adds either var0 *or* var1*),* $\mathfrak{t}_0$ *contains three ordered tasks named* comp*,* comp*, and* pr0*, and* $S_0 = \emptyset$*. Then* $H = (F, C, O, M, \delta, \mathfrak{t}_0, S_0)$ *is a FOND HTN problem. Every sequence of method applications solves* H*, except when one immediately applies the* stop *method twice or when one applies the* cont *method infinitely often.*

Fragments

Many fragments of HTN planning have FOND analogues.

Let $H = (F, C, O, M, \delta, \mathfrak{t}_0, S_0)$ be a FOND HTN problem.

- H is *regular* (Erol, Hendler, and Nau 1994) if every task network of H contains at most one compound task, and, if a task network does contain a compound task, this task is ordered after all other tasks in the task network.
- H is *loop-unrolling* (Dekker and Behnke 2024) if it contains at most one compound task name and two methods:

$$|C| \leq 1 \text{ and } |M| \leq 2.$$

If a loop-unrolling problem with a non-primitive initial task network is solvable, at least one of its methods (which we denote "stop") must have a primitive task network, so only one of its methods (which we denote "cont") can have a non-primitive task network.

The following is known (Chen and Bercher 2021, Thm. 5.3):

Theorem 2. *It is in EXPSPACE to decide whether a given regular FOND HTN problem is solvable.*

Even *deterministic* loop-unrolling HTN problems are undecidable; see Höller et al. (2023). Hence:

Theorem 3. *It is undecidable whether a given loop-unrolling FOND HTN problem is solvable.*

The following is our new result:

Theorem 4. *It is EXPSPACE-hard to decide whether a given regular loop-unrolling FOND HTN problem is solvable.*

Translations

In this section, we show that planning games in which Éloise chooses the game length at the start are equivalent to regular loop-unrolling FOND HTN problems.

Theorem 5. *Let* $\mathfrak{P}$ *be a STRIPS planning problem. Then we can compute a regular loop-unrolling FOND HTN problem* H *in polynomial time such that the following are equivalent for all* $r \in \mathbb{N}$*:*

1. *Éloise wins* $\mathbb{G}(\mathfrak{P}, r)$.
2. H *can be solved by applying the* cont *method a total of* r *times before applying the* stop *method.*

Proof. Calculate $\mathfrak{P}' = (P', A', I', G')$ using Lemma 2-1. We construct H such that for all $r \in \mathbb{N}$, Éloise wins $\mathbb{G}(\mathfrak{P}', 2r)$ iff 2 holds. W.l.o.g., A' contains an action with a precondition and – by Lemma 1 – an action without preconditions.

We design the cont task network to model two turns of the planning game. This task network will contain tasks for all

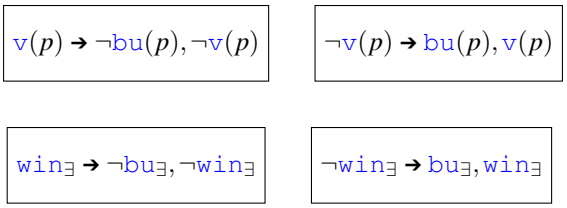

Figure 3: The task network "backup" in the proof of Thm. 5. Each box is a task. ➔ separates preconditions from effects. Include tasks for all $p \in P'$. This task network has no nondeterminism or order constraints.

actions of $\mathfrak{P}$, so we need to "undo" actions of $\mathfrak{P}$. We show how to backup a state of $\mathfrak{P}'$ into new variables $\mathtt{bu}(p)$.

The propositional variables of H:

$$\big\{\mathtt{v}(p), \mathtt{bu}(p) : p \in P'\big\} \sqcup \big\{\mathtt{do}(a) : a \in A'\big\}$$
$$\sqcup \big\{\mathtt{done}, \mathtt{win}_\exists, \mathtt{bu}_\exists\big\}.$$

$\mathtt{win}_\exists$ (or $\mathtt{bu}_\exists$) says Abélard chose a non-applicable action and therefore loses. Write $\mathtt{v}[S] = \big\{\mathtt{v}(p) : p \in S\big\}$ for $S \subseteq P'$.

The "backup" tool (Fig. 3) copies the truth values of all $\mathtt{v}(p)$ and $\mathtt{win}_\exists$ onto the respective bu variables. The task network "restore" is the same except that all bu variables are swapped with the other variables.

For $a = (\Pi_+, \Pi_-, E_+, E_-) \in A'$, introduce two primitive task names Éloise(a) and Abélard(a) of H. Define

$$\mathbb{E}_a = \Big\{\Big(E_+ \sqcup \{\mathtt{done}\}, E_- \sqcup \{\mathtt{do}(a)\}\Big)\Big\},$$

$$\delta\big(\text{Éloise}(a)\big) = \Big(\mathtt{v}[\Pi_+], \mathtt{v}[\Pi_-] \sqcup \{\mathtt{done}\}, \mathbb{E}_a\Big),$$

and

$$\delta\big(\text{Abélard}(a)\big) = \Big(\{\mathtt{do}(a)\}, \emptyset, \mathbb{E}_a\Big).$$

Let comp be the compound task name of H. Define the cont task network of H as in Fig. 4. The stop task network contains just two unorderded tasks – one requiring $\mathtt{v}[G']$, the other requiring $\mathtt{win}_\exists$ – that both add the preconditions of the other. The initial task network of H contains just the compound task, and the initial propositional state of H is $\mathtt{v}[I']$. Evidently, H is a regular loop-unrolling problem.

When starting to execute the primitive tasks of some copy of the cont task network, done is false. Éloise's move choice $a \in A'$ corresponds to the task Éloise(a) that is executed first. Then done is added and a backup is made. Next, the tasks ordered immediately after the backup ensure that the remaining tasks Éloise(a') can be executed, but they have no net effect because the backup is restored afterwards.

Abélard's move choice $a' \in A'$ corresponds to the nondeterministic effect $\mathtt{do}(a')$ of the next task. Again done is false. Here w.l.o.g. an *applicable* action is chosen – otherwise $\mathtt{win}_\exists$ can be added and H is easily solved (using an action without preconditions for Éloise's choices in the remainder of the plan). Then one is forced to execute Abélard(a'), make a backup and execute the remaining tasks before restoring again, which is again always possible

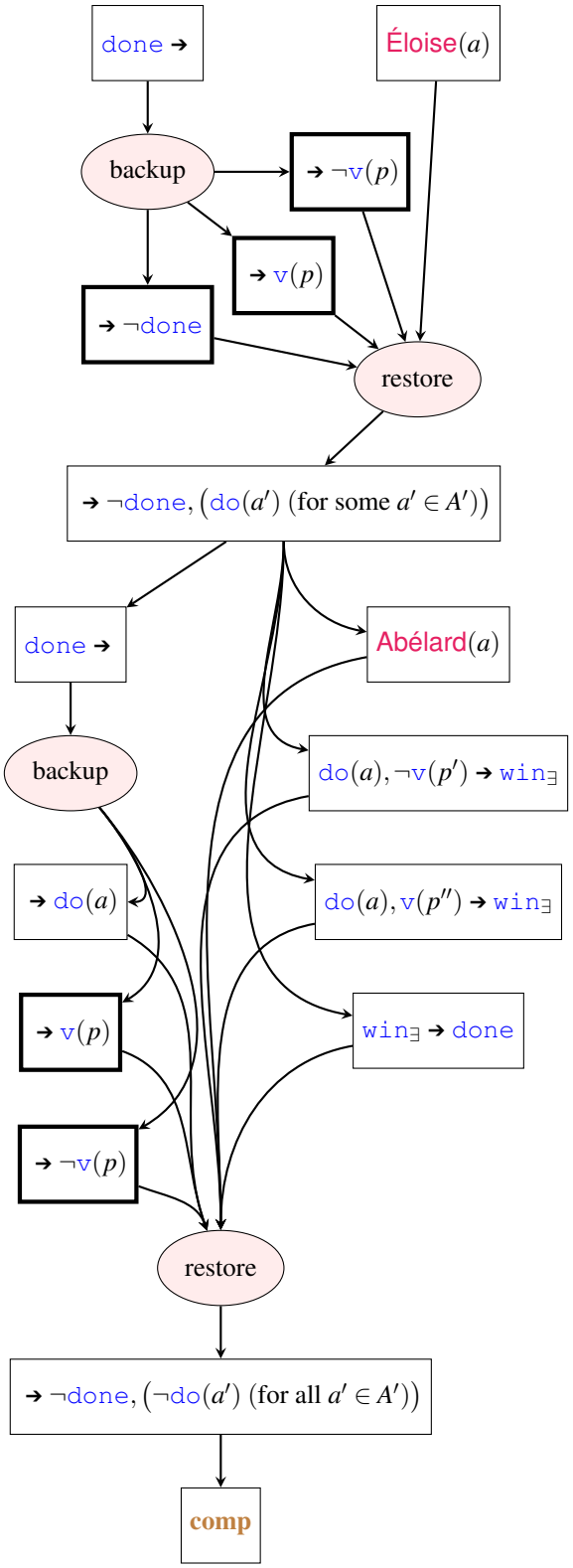

Figure 4: The non-primitive method task network in the proof of Thm. 5. Each box is a task. ➔ separates preconditions from effects. Include tasks for all $a \in A'$ and $p \in P'$ and positive preconditions p' and negative preconditions p'' of a. Include enough (e.g., $|A'|$ many) copies of the tasks marked with thick border.

(in particular, $\texttt{win}_\exists$ can be temporarily added because A' contains an action with a precondition).

The stop task network says that either the goal of $\mathfrak{P}'$ is satisfied or $\texttt{win}_\exists$ is true. □

Proof of Thm. 4. Thms. 1 and 5. □

Theorem 6. *Let H be a regular loop-unrolling FOND HTN problem. Let T_0, T_{cont}, T_{stop} be the sets of primitive tasks in the three task networks of H. Then we can compute a STRIPS planning problem $\mathfrak{P}$ in polynomial time such that for all $r \in \mathbb{N}$ the following are equivalent:*

1. *Éloise wins $\mathbb{G}\left(\mathfrak{P}, 2\left(|T_0| + r|T_{cont}| + |T_{stop}|\right)\right)$.*
2. *H can be solved by applying the cont method a total of r times before applying the stop method.*

Proof. The execution of a primitive task in a plan for H corresponds to two turns in the planning game. Éloise chooses the task and Abélard chooses the effect. Special variables $\text{done}(t)$ will track the progress of the task network. For the cont task network, Éloise can use special actions $t^{\circlearrowleft}_{\text{Éloise}}$ to reset these variables, mimicing a cont method application.

Assume w.l.o.g. that T_0, T_{cont}, and T_{stop} are pairwise disjoint. The variables of $\mathfrak{P}$ are the variables of H and $\text{do}(t), \text{done}(t)$ for all $t \in T := T_0 \sqcup T_{\text{cont}} \sqcup T_{\text{stop}}$. For $T' \subseteq T$ write $\text{done}[T'] = \{\text{done}(t) : t \in T'\}$ and similarly for do. For each $t \in T$ with name pr, construct some actions of $\mathfrak{P}$:

- $t_{\text{Éloise}}$ has the preconditions of pr and moreover requires $\neg\text{done}(t)$, $\neg\text{do}(t')$ for all t', and $\text{done}(t')$ for every t' that is ordered before t (in the same task network of H that t is in). If $t \in T_{\text{cont}} \sqcup T_{\text{stop}}$, the action $t_{\text{Éloise}}$ also requires $\text{done}[T_0]$. If $t \in T_{\text{stop}}$, the action $t_{\text{Éloise}}$ also requires $\text{done}[T_{\text{cont}}]$. The effect of $t_{\text{Éloise}}$ is that $\text{do}(t)$ becomes true.
- If t is an order-minimal task in T_{cont}, also introduce the action $t^{\circlearrowleft}_{\text{Éloise}}$ that is the same as $t_{\text{Éloise}}$ except that it consumes $\text{done}[T_{\text{cont}}]$ (so $\text{done}(t)$ is not a negative precondition anymore) and requires $\neg\text{done}(t')$ for every $t' \in T_{\text{stop}}$.
- For each possible effect $e = (E_+, E_-)$ of pr, the action $t^e_{\text{Abélard}}$ of $\mathfrak{P}$ consumes $\text{do}(t)$, adds $\{\text{done}(t)\} \sqcup E_+$, and deletes E_-.

The initial state of $\mathfrak{P}$ is the initial propositional state of H together with $\text{done}[T_{\text{cont}}]$, and the goal of $\mathfrak{P}$ is $\text{done}[T]$.

The only legal play is that Éloise chooses tasks $t \in T$ to execute in an order complying with the hierarchical structure of H (she uses the corresponding action $t_{\text{Éloise}}$ unless she is starting a new copy of T_{cont}: then she uses $t^{\circlearrowleft}_{\text{Éloise}}$), and Abélard replies with actions $t^e_{\text{Abélard}}$ mimicing nondeterministic effects. □

Conclusion

We introduced a formalism for general game-playing and proved EXPSPACE-completeness for the case in which one of the players chooses the game length up front. Then we provided a reformulation of those games in terms of FOND HTN planning.

References

Arora, S.; and Barak, B. 2007. *Computational Complexity: A Modern Approach*. Princeton, Online Draft Version.

Camacho, A.; Baier, J. A.; Muise, C.; and McIlraith, S. A. 2018. Finite LTL Synthesis as Planning. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 29–38. Delft: AAAI Press.

Chen, D.; and Bercher, P. 2021. Fully Observable Nondeterministic HTN Planning – Formalisation and Complexity Results. In *Proceedings of the 31st International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 74–84.

Cimatti, A.; Hunsberger, L.; Micheli, A.; Posenato, R.; and Roveri, M. 2016. Dynamic Controllability via Timed Game Automata. *Acta Informatica*, 53: 681–722.

Dekker, P. M.; and Behnke, G. 2024. Barely Decidable Fragments of Planning. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI 2024)*, 4198–4206. Santiago de Compostela.

Erol, K.; Hendler, J.; and Nau, D. 1994. HTN Planning: Complexity and Expressivity. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI 1994)*, 1123–1128.

Fraenkel, A. S.; and Lichtenstein, D. 1981. Computing a perfect strategy for n x n chess requires time exponential in n. *Journal of Combinatorial Theory, Series A*, 31: 199–214.

Gazen, B.; and Knoblock, C. 1997. Combining the expressivity of UCPOP with the efficiency of graphplan. In *Proceedings of the European Conference on Planning (ECP 1997): Recent Advances in AI Planning*, volume 4, 221–233. Springer.

Genesereth, M.; Love, N.; and Pell, B. 2005. General Game Playing: Overview of the AAAI Competition. *AI Magazine*, 26(2): 62–72.

Höller, D.; Lin, S.; Erol, K.; and Bercher, P. 2023. From PCP to HTN Planning Through CFGs. *The 10th International Planning Competition (IPC 2023) – Planner and Domains Abstracts*.

Muise, C.; Felli, P.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2016. Planning for a Single Agent in a Multi-Agent Environment Using FOND. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3206–3212. New York: AAAI Press.

Rintanen, J. 2004. Complexity of Planning with Partial Observability. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 345–354.

Zermelo, E. 1913. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth Congress of Mathematicians (ICM 1913), Vol. 2*, 501–504. Cambridge: Cambridge University Press.