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

Anonymous submission

Abstract

We introduce a new version of general game-playing in which one of the players chooses the length of the game. Consider a classical planning problem and suppose that two players 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 introduced by Chen and Bercher (2021). 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 focusses on the simplistic setting of classical planning. It however lacks the expressiveness to capture more complex and thus more realistic settings. Notably, classical planning assumes that the execution of actions is always deterministic – which implicitly means that the actor is in full control of the environment. For certain applications this is not the case. Let's consider a grid-shaped warehouse in which the planner controls a single robot. The robot's move actions are in reality non-deterministic, as it might fail to physically traverse from one cell to the next. But, the robot still has full observability of its environment, leading to the notion of Fully-Observable Non-Deterministic (FOND) planning.

In this paper, we extend the previous theoretical investigations into FOND planning two ways. Firstly, 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. Similar connections between planning and games were drawn (Muise et al. 2016; Camacho et al. 2018). 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 that 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. In our robot example, the robot has to announce when it is going to arrive at its goal before it starts moving. This happens e.g. if the robot is providing a transport or pick-up service (similar to a bus) and must announce its schedule before it starts operating. As our main result, we prove that solving chosen length planning games is EXPSPACE-complete and thus harder than general FOND planning, which is only EXPTIME-complete (Rintanen 2004).

Secondly, we show that this new type of planning games and FOND planning is related to FOND planning for Hierarchical Task Network (HTN) planning. HTN planning extends classical planning by adding the notion of compound tasks – which cannot be executed directly. Instead, 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 however left several open questions. With a reduction to planning games, we solve one of these cases – namely that strong FOND HTN planning for regular HTNs is EXPSPACE-complete.

Planning Games

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

- P is a finite set of *(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*.

$\checkmark = (\emptyset, \emptyset, \emptyset, \emptyset)$ denotes the empty action. A *(propositional) state* of Π is a subset of P. An action $a = (\pi_+, \pi_-, e_+, e_-) \in A$ consists of a a *positive precondition* π_+, a *negative precondition*[1] π_-, 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* $\mathbb{G}(\Pi, r)$ is played by two players, Éloise and Abélard, and has a maximal duration of r plies. The game is initialized in the initial state I. Éloise goes first. The players take turns choosing an applicable action in

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

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. a subset of) the current state; otherwise Abélard wins.

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

Example **1**. Let $\Pi = (P, A, I, G)$ 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 $\mathbb{G}(\Pi, r)$ iff $r = 195$ or $r \geq 197$. If $G = P$, Éloise has a winning strategy for $\mathbb{G}(\Pi, r)$ iff $r \geq 197$ and r is odd.

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

Transformations of Planning Games

To some extent, it is w.l.o.g. that players can pass a turn. This is expressed by the next lemma.

Lemma **2**. Let Π be a STRIPS planning problem. Then:

1. We can compute in polynomial time a STRIPS planning problem Π' that has $\checkmark$ as an action such that for all even numbers $r \in \mathbb{N}$, the games $\mathbb{G}(\Pi, r)$ and $\mathbb{G}(\Pi', r)$ 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 Π' that has an action without preconditions such that for all $r \in \mathbb{N}$, the games $\mathbb{G}(\Pi, r)$ and $\mathbb{G}(\Pi', r)$ are won by the same player.

Proof. We only prove 1. 2 and 3 are no more difficult to prove.

Write $\Pi = (P, A, I, G)$. Let

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

Define a set A'_a of variants of an action $a = (\pi_+, \pi_-, e_+, e_-) \in A$ as follows. In any case put

$$\big(\pi_+ \sqcup \{\star, \exists\}, \pi_-, e_+ \sqcup \{g\}, e_- \sqcup \{\exists\}\big) \in A'_a \tag{1}$$

and

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

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

$$\big(\pi_+ \sqcup \{\star\}, \pi_- \sqcup \{\exists\}, e_+ \sqcup \{\exists\}, e_- \sqcup \{g\}\big) \in A'_a. \tag{2}$$

Otherwise, for each $p \in G \setminus (\pi_+ \cup e_+)$ put

$$\big(\pi_+ \sqcup \{\star\}, \pi_- \cup \{p, \exists\}, e_+ \sqcup \{\exists\}, e_- \sqcup \{g\}\big) \in A'_a. \tag{3}$$

Let

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

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 $\Pi' = (P', A', I', G')$ is as desired.

As long as only actions in

$$\bigsqcup_{a \in A} A'_a \tag{4}$$

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 (4) will be played, as playing $\checkmark$ allows the other player to win. In particular, Éloise's last move adds g (1), so Abélard wants to delete it with his last move. He can do this iff he can either apply an action that deletes some goal variable in G (2) or there is a false goal variable in G and he can apply an action that does not add it (3). □

Next we present two lemmas that manipulate the length of planning games.

Lemma **3**. Let Π be a STRIPS planning problem. Then:

1. We can compute a STRIPS planning problem Π' in polynomial time such that for all $r \in \mathbb{N}$, the games $\mathbb{G}(\Pi, r)$ and $\mathbb{G}(\Pi', 2r)$ are won by the same player.
2. As 1 but with $2\lceil r/2 \rceil$ instead of $2r$.
3. As 1 but with $2r + 1$ instead of $2r$.
4. As 1 but with $2\lfloor r/2 \rfloor + 1$ instead of $2r$.

Proof. We only prove 1. 3 follows from 1 and 4.

Write $\Pi = (P, A, I, G)$. Let

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

Define a set A'_a of variants of an action $a = (\pi_+, \pi_-, e_+, e_-) \in A$ by

$$A'_a = \Big\{\big(\pi_+ \sqcup \{\exists\}, \pi_- \sqcup \{w\}, e_+ \sqcup \{w\}, e_- \sqcup \{\exists\}\big), \tag{5}$$

$$\big(\pi_+, \pi_- \sqcup \{\exists, w\}, e_+, e_- \sqcup \{\exists\}\big)\Big\}. \tag{6}$$

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 $\Pi' = (P', A', I', G')$ is as desired.

Éloise starts with an action of the form (5) (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 (6), making $\exists$ true again. Éloise again proceeds with an action from (5) (provided $r \geq 3$), and so on.

We see that $\mathbb{G}(\Pi, 2n)$ and $\mathbb{G}(\Pi', 4n)$ are won by the same player, and the games $\mathbb{G}(\Pi, 2n+1)$, $\mathbb{G}(\Pi', 4n+1)$, $\mathbb{G}(\Pi', 4n+2)$ and $\mathbb{G}(\Pi', 4n+3)$ are all won by the same player, for all $n \in \mathbb{N}$. □

Lemma **4**. Let Π be a STRIPS planning problem and $m \in \mathbb{N}_{>0}$ a number in unary encoding. Then we can compute a STRIPS planning problem Π' in polynomial time such that the following are equivalent for all $r \in \mathbb{N}$:

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

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

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

$$A' = \Big\{ \big(\pi_+ \sqcup \{p_k\}, \pi_-, e_+ \sqcup \{p_{k+1}\}, e_- \sqcup \{p_k\}\big) : (\pi_+, \pi_-, e_+, e_-) \in A \;\&\; k \in \mathbb{Z}/m\mathbb{Z} \Big\},$$

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

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

Main Result

Theorem **5**. It is EXPSPACE-complete to decide given a STRIPS planning problem Π whether there exists $r \in \mathbb{N}$ such that Éloise has a winning strategy for the r ply planning game $\mathbb{G}(\Pi, r)$. This remains true when confining attention to problems Π with action ✓, and it does not matter whether r is restricted to be even or odd.

We introduce the following notation for the proof.

Notation **6**. If $\Pi = (P, A, I, G)$ and s is a propositional state, write $\Pi_s = (P, A, s, G)$.

Membership

Alg. 1 requires $O(2^{|P|})$ memory and searches for a winning even length. We derived if from the algorithm given by (Chen and Bercher 2021, Thm. 5.3) using the proof of Thm. 11 below. Line 1 sets S to the set of all states s such that Éloise wins $\mathbb{G}(\Pi_s, 0)$. During the loop on Line 2, it is maintained that S is the set of all states s such that Éloise wins $\mathbb{G}(\Pi_s, 2r)$. Then whenever $I \in S$, we have found an even length of the game that is winning for Éloise (Lines 3-4). To calculate the next value of S, initialize it to $\emptyset$ in Line 5 while keeping the old set S in memory until the new one has been computed (Line 14). We take a state s (Line 6) and wonder whether Éloise can win $\mathbb{G}(\Pi_s, 2r+2)$, meaning that 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 S. Indeed, if Abélard has a legal (Line 10) reply to a leading out of S (Line 11), we reach Line 12 discarding the action a; if Abélard has no such option, Line 13 adds s to the new S. If the same set S occurs twice during the main loop, there is no r such that Éloise wins $\mathbb{G}(\Pi, 2r)$. We cannot keep a history of the sets S

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

```
Data: STRIPS planning problem $\Pi = (P, A, I, G)$
Result: $r \in \mathbb{N}$ such that Éloise has a winning strategy
        for $\mathbb{G}(\Pi, 2r)$, or NIL if such $r$ does not exist
1  Let $S = \{s \subseteq P : G \subseteq s\}$.
2  forall $r = 0, \ldots, 2^{2^{|P|}} - 1$ do
3      if $I \in S$ then
4          return $r$
5      Let $S' = \emptyset$.
6      for $s \subseteq P$ do
7          for $a \in A$ do
8              if $a$ is applicable in $s$ then
9                  for $b \in A$ do
10                     if $b$ is applicable in $\gamma(s, a)$ then
11                         if $\gamma(\gamma(s, a), b) \notin S$ then
12                             continue with the loop on
                               Line 7
13                 Add $s$ to $S'$.
14     Let $S = S'$.
15 return NIL
```

as this may require doubly exponential space, but we know that the same set S must occur twice if we iterate $2^{2^{|P|}}$ times, so at that point we return a negative answer with Line 15. Since r can be encoded in binary, it can be iterated until this limit using only a singly exponential amount of space.

To search for a winning odd length $2r+1$ instead, change the initialization of 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

Let T be a Turing machine. 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. Assume that $n \in \mathbb{N}$ such that the head of T remains in the interval $[0, 2^n)$. We construct a STRIPS planning problem Π in time polynomial in n and an encoding size of T, such that for all $r \in \mathbb{N}$, machine T halts after exactly r transitions iff Éloise wins $\mathbb{G}\left(\Pi, (2n+4)r\right)$. This is sufficient for hardness in Thm. 5 in view of Lemma 4. 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.

The basic idea is that the set of all possible states after some number of plies in Éloise's strategy encodes a configuration of T. Each consecutive sequence of $2n+4$ plies of the planning game corresponds to a single regression step in the run of T. By the proof of Lemma 4, we 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.

Variables Introduce the following set of variables:

$$\{d_i : i < n\} \sqcup \Sigma \sqcup \{h\} \sqcup Q. \quad (7)$$

The variables d_i encode the location of a tape cell of T. For $c < 2^n$, let P_c be the set containing d_i for each $i < n$ such that digit i in the binary representation of c has value 1.

The variables in (7) describe a configuration of T. The idea is that after every full cycle of the planning game, exactly one variable in the set

$$\Sigma \sqcup \{h\} \sqcup Q \quad (8)$$

is true. 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 $\mathbb{G}\left(\Pi_{P_c \sqcup \{\sigma\}}, (2n+4)t\right)$ iff tape cell c of T contains symbol σ at time t.

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

(state) For all $q \in Q$, Éloise wins $\mathbb{G}\left(\Pi_{P_c \sqcup \{q\}}, (2n+4)t\right)$ iff T is in state q at time t.

In particular, we mean that Éloise should lose the games of length $(2n+4)t$ if T halts after less than t transitions (viz.: tape, head and machine cease to exist after T halts).

We next describe the rules of the game, omitting the technical details on how to encode them in STRIPS using additional variables. However, we mention that a total number of variables that is linear in n and an encoding size of T suffices.

Initial State The initial state is $\{q_\omega\}$ (disregarding auxiliary variables).

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

- If some $\sigma \in \Sigma$ is true, Éloise adds h and chooses from two possible courses of action:
  - *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.
  - *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. Changing this number means that she chooses a different number than what was encoded at the start of the cycle. I.e. she flips at least one of the variables d_i and more if she wants to, but flips none of them twice. (Indeed, with the help of additional variables this can be described by STRIPS over $n+1$ of Éloise's moves, viz. $2n+1$ plies.)
- 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 (if T moves the head to the right in that situation) 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 $2n+1$ plies.)

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 that 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 (8) except one variable of choice.

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

Goal At the end of the planning game, Éloise wins iff all variables in $\{\odot\} \sqcup Q \setminus \{q_0\}$ are false and, if h is true, also all variables $d_0, \ldots, d_{n-1}$ are false. 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 Π. We next prove (tape), (head) and (state) by induction on t.

Inductive Basis Consider $t = 0$. Then (tape) says that $\sigma = \bigcirc$ iff tape cell c of T contains symbol σ at time 0, which was one of our assumptions. (head) says that $P_c = \emptyset$ iff the head of T is at position c at time 0, which was one of our assumptions. (state) says that $q = q_0$ iff T is in state q at time 0, which was one of our assumptions.

Inductive Hypothesis Suppose that (tape), (head) and (state) hold for t.

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 that the cycle ends in a propositional state of the form $F_{c'} \sqcup \{h\}$ 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 that the cycle ends in a propositional state of the form $P_c \sqcup \{h\}$, 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 she can choose $c' \in [0, 2^n) \setminus \{c\}$ 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

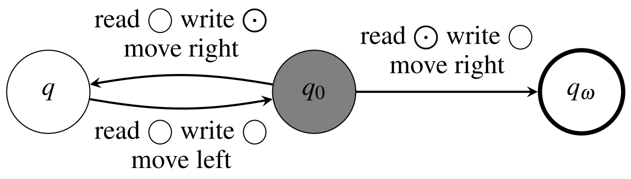

Figure 1: A Turing machine. The initial state is filled and the halting state has a thick border. The irrelevant specification on reading $\odot$ in state q is omitted.

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

Time	State	Cell 0	Cell 1
0	q_0	▨ $\bigcirc$	$\bigcirc$
1	q	$\odot$	▨ $\bigcirc$
2	q_0	▨ $\odot$	$\bigcirc$
3	q_ω	$\bigcirc$	▨ $\bigcirc$

- 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 $\mathbb{G}\left(\Pi, (2n+4)r\right)$.

Example Consider the Turing machine T in Fig. 1. Its run is given in Table 1. With $n = 1$, the head stays in the interval $[0, 2^n)$. The machine halts after three transitions, meaning that Éloise can win $\mathbb{G}(\Pi, 18)$.

- The first six plies (time 3 $\rightsquigarrow$ time 2): Éloise starts by deleting q_ω and adding h, $\odot$ and q_0, which is allowed because T transitions to q_ω when reading $\odot$ in state q_0. She makes d_0 false, encoding location $0 \cdot 2^0 = 0$ on the tape. At the end of this cycle, it holds that
$$(h \vee \odot \vee q_0) \wedge \neg d_0.$$
- The next six plies (time 2 $\rightsquigarrow$ time 1):
  - If h is true, Éloise adds $\bigcirc$ and q and makes d_0 true (incrementing the encoded number to $1 \cdot 2^0 = 1$), which is allowed because T moves the head to the left when reading $\bigcirc$ in state q.
  - If $\odot$ is true, Éloise adds h and chooses Preserve, and if Abélard then keeps h, she makes d_0 true, changing the location from 0 to 1.
  - Otherwise q_0 is true, in which case Éloise deletes it and adds h, $\bigcirc$ and q, which is allowed because T transitions to q_0 when reading $\bigcirc$ in state q, and chooses location 1 on the tape.

  At the end of this cycle, it holds that
$$(h \vee \bigcirc \vee \odot \vee q) \wedge (d_0 \leftrightarrow \neg\odot).$$
- The final six plies (time 1 $\rightsquigarrow$ time 0):
  - If h is true, Éloise adds $\bigcirc$ and q_0 and makes d_0 false (decrementing the encoded number), which is allowed because T moves the head to the right when reading $\bigcirc$ in state q_0.
  - If $\bigcirc$ is true, Éloise adds h and chooses Preserve, and if Abélard then keeps h, she makes d_0 false, changing the location from 1 to 0.
  - If $\odot$ is true, Éloise adds h and chooses Write, deleting $\odot$ and adding $\bigcirc$ and q_0, which is allowed because T writes $\odot$ when reading $\bigcirc$ in state q_0. In this case, d_0 was already false.
  - If q is true, Éloise deletes it and adds h, $\bigcirc$ and q_0, which is allowed because T transitions to q when reading $\bigcirc$ in state q_0, and chooses location 0 on the tape.

  At the end, it holds that
$$(h \vee \bigcirc \vee q_0) \wedge (d_0 \rightarrow \bigcirc),$$
  so the goal is achieved.

Adding Passes If we naively add ✓ to this construction, the proof falls apart, e.g. when T halts after only one transition. To obtain the second statement in Thm. 5, extend the cycle of $2n+4$ plies to a cycle of $2n+6$ plies. During the last two plies of the cycle, nothing happens. Relax the goal: for Éloise to win, the cyclic plycounter need not necessarily be at 0, but may also be at $2n+5$. Add the action ✓. Moreover add a special variable $\star$ that is true in the initial state and is a precondition of every action except ✓. Introduce $2n+6$ more actions: when the plycounter is at an even value, one can delete $\star$ while making the goal false, and when the plycounter is at an odd value one can delete $\star$ while making the goal true. (These actions also require $\star$.) Call Π' the resulting problem. Then neither player ever has an incentive to play ✓ (except possibly on the very last ply 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 $\mathbb{G}\left(\Pi', (2n+6)r + \varepsilon\right)$ for some $-1 \leq \varepsilon \leq 1$.
3. As 2 but with "all" instead of "some".

And Éloise loses $\mathbb{G}(\Pi', r)$ whenever r is not congruent to -1, 0 or 1 modulo $2n+6$.

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*

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

For $t \in T$ we write $T \setminus t = T \setminus \{t\}$. We write $\mathfrak{t} \setminus t$ as a shorthand for

$$\mathfrak{t}|(T \setminus t).$$

An *embedding* $\phi : \mathfrak{t} \hookrightarrow \mathfrak{t}'$ of task networks $\mathfrak{t} = (T, \prec, \alpha)$ and $\mathfrak{t}' = (T', \prec', \alpha')$ 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 $\phi_i : \mathfrak{t}_i \hookrightarrow \mathfrak{t}$ such that

$$T = \bigsqcup_{i \in I} \operatorname{Im} \phi_i$$

and $\phi_i(t_i) \not\prec \phi_j(t_j)$ 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*;
- $\delta : O \to \mathscr{P}(F)^2 \times \mathscr{P}\big(\mathscr{P}(F)^2\big)$ 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. Let $\mathfrak{t} = (T, \prec, \alpha)$ be a task network over $C \sqcup O$. It is called a task network *of H* if $\mathfrak{t} = \mathfrak{t}_0$ or $(\mathbf{c}, \mathfrak{t}) \in M$ for some $\mathbf{c} \in C$. We call a task $t \in T$ *compound* if $\alpha(t) \in C$ and *primitive* if $\alpha(t) \in O$. We call $\mathfrak{t}$ *primitive* if all tasks in T are primitive (i.e. $\mathfrak{t}$ is a task network over O).

A decomposition method is applied to change a non-primitive task network into another task network. Suppose that $\mathfrak{t}_1 = (T_1, \prec_1, \alpha_1)$, $\mathfrak{t}_2 = (T_2, \prec_2, \alpha_2)$ and $\mathfrak{t} = (T, \prec, \alpha)$ are task networks, $t \in T_1$ and $\mu = \big(\alpha_1(t), \mathfrak{t}\big)$ is a method. We write $\mathfrak{t}_1 \to_{t/\mu} \mathfrak{t}_2$ provided there exists an embedding $\phi : \mathfrak{t} \hookrightarrow \mathfrak{t}_2$ such that

$$T_2 = (T_1 \setminus t) \sqcup \operatorname{Im} \phi$$

and for all $t_1 \in T_1 \setminus t$ and $t' \in T$ and $* \in \{\prec, \succ\}$ it holds

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

Intuitively this means that t got replaced by $\mathfrak{t}$. The task network $\mathfrak{t}_2$ exists and is unique up to isomorphism.

The action mapping δ associates with each $\mathsf{pr} \in O$ a triple $\delta(\mathsf{pr}) = (\pi_+, \pi_-, E)$ consisting of a *positive precondition* π_+, a *negative precondition* π_-, and a set E of *effects*. An 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

$$\Gamma(s, \mathsf{pr}) = \big\{(s \setminus e_-) \cup e_+ : (e_+, e_-) \in E\big\}.$$

We say that a belief state $\mathbb{S} \subseteq \mathrm{TN}_O \times \mathscr{P}(F)$ is *solvable* if for all $\big(\mathfrak{t} = (T, \prec, \alpha), s\big) \in \mathbb{S}$, 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

$$\Big\{(\mathfrak{t} \setminus t, s') : s' \in \Gamma\big(s, \alpha(t)\big)\Big\}$$

is, recursively, solvable. In this paper, we consider plans whose method applications are independent of non-determinism. That is, 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

$$\big\{(\mathfrak{t}_n, s_0)\big\}$$

is a solvable belief state.

Example **7**. Let $F = \{$`var0`, `var1`$\}$, $C = \{\mathbf{comp}\}$, $O = \{\mathsf{pr0}, \mathsf{pr1}, \mathsf{pr2}\}$,

$$M = \big\{\mathsf{stop} = (\mathbf{comp}, \mathfrak{o}), \mathsf{cont} = (\mathbf{comp}, \mathfrak{t})\big\}$$

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

$$\delta(\mathsf{pr0}) = \Big(\{\mathtt{var0}\}, \emptyset, \Big\{\big(\{\mathtt{var1}\}, \emptyset\big)\Big\}\Big)$$

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

$$\delta(\mathsf{pr1}) = \Big(\{\mathtt{var1}\}, \emptyset, \Big\{\big(\{\mathtt{var0}\}, \emptyset\big)\Big\}\Big)$$

(i.e. pr1 requires `var1` and adds `var0`),

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

(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. We may write $\delta(\mathsf{pr0})$ as `var0` ➔ `var1`.

Fragments

Many fragments of HTN planning have analogues in the FOND setting.

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 it does contain a compound task, this task is the last task; formally:

$$\forall \mathfrak{t} = (T, \prec, \alpha) \in \mathrm{TN}_{C \sqcup O} : \Big(\big(\exists \mathbf{c} : (\mathbf{c}, \mathfrak{t}) \in M\big) \mid \mathfrak{t}_0 = \mathfrak{t}\Big)$$
$$\implies \forall t, t' \in T : \alpha(t) \in C \implies \big(t' \prec t \mid t = t'\big).$$

- H is *loop-unrolling* (Dekker and Behnke 2024) if it contains at most one compound task name and two methods:

$$|C| \leq 1 \; \& \; |M| \leq 2.$$

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 **8**. 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 **9**. It is undecidable whether a given loop-unrolling FOND HTN problem is solvable.

The following is our new result:

Theorem **10**. 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 **11**. Let Π 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 $\mathbb{G}(\Pi, r)$.
2. H can be solved by applying cont a total of r times before applying stop.

Proof. Calculate $\Pi' = (P', A', I', G')$ using Lemma 3-1. We construct the problem H such that for all $r \in \mathbb{N}$, Éloise wins $\mathbb{G}(\Pi', 2r)$ iff 2 holds. W.l.o.g., A' contains an action with a precondition and – by Lemma 2 – an action without preconditions.

The propositional variables of H:

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

Write $\mathtt{v}[s] = \{\mathtt{v}(p) : p \in s\}$ for $s \subseteq P'$.

We introduce a "backup" tool. See Fig. 2. Notice that there are no order constraints in this task network. It creates an extra copy of (the HTN encoding of) a state of Π' and the variable $\mathtt{win}_\exists$. More precisely, it copies the truth values of all $\mathtt{v}(p)$ and $\mathtt{win}_\exists$ onto the respective $\mathtt{BU}$ variables. The task network "restore" is the same except that all $\mathtt{BU}$ variables are swapped with the other variables.

For $a = (\pi_+, \pi_-, e_+, e_-) \in A'$, introduce two primitive task names $\text{Éloise}(a)$ and $\text{Abélard}(a)$ of H. Define

$$E_a = \Big\{ \big(e_+ \sqcup \{\mathtt{done}\}, e_- \sqcup \{\mathtt{do}(a)\} \big) \Big\},$$

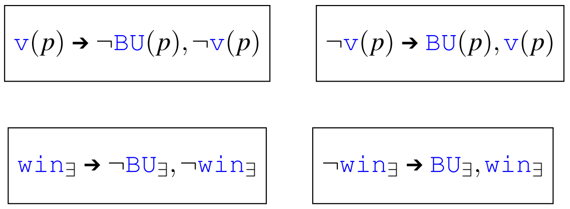
$\mathtt{v}(p) \rightarrow \neg\mathtt{BU}(p), \neg\mathtt{v}(p)$
$\neg\mathtt{v}(p) \rightarrow \mathtt{BU}(p), \mathtt{v}(p)$
$\mathtt{win}_\exists \rightarrow \neg\mathtt{BU}_\exists, \neg\mathtt{win}_\exists$
$\neg\mathtt{win}_\exists \rightarrow \mathtt{BU}_\exists, \mathtt{win}_\exists$

Figure 2: The task network "backup" in the proof of Thm. 11. Include tasks for all $p \in P'$.

$$\delta\big(\text{Éloise}(a)\big) = \big(\mathtt{v}[\pi_+], \mathtt{v}[\pi_-] \sqcup \{\mathtt{done}\}, E_a\big)$$

and

$$\delta\big(\text{Abélard}(a)\big) = \big(\{\mathtt{do}(a)\}, \emptyset, E_a\big).$$

Let **comp** be the compound task name of H. Define the cont task network of H as in Fig. 3. The stop task network is given in Fig. 4. 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, $\mathtt{done}$ is false. Éloise's move choice $a \in A'$ corresponds to the task $\text{Éloise}(a)$ that is executed first. Then $\mathtt{done}$ becomes true and a backup is made. Next, the tasks ordered immediately after the backup ensure that the remaining tasks $\text{Éloise}(a')$ 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 $\mathtt{done}$ is false. Here w.l.o.g. an action is chosen whose preconditions are satisfied; otherwise $\mathtt{win}_\exists$ can be made true 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 $\text{Abélard}(a')$, make a backup and execute the remaining tasks before restoring again, which is again always possible (in particular, $\mathtt{win}_\exists$ can temporarily be made true because A' contains an action with a precondition).

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

Proof of Thm. 10. Thms. 5 and 11. □

Theorem **12**. Let H be a regular loop-unrolling FOND HTN problem. Let $T_0, T_{\text{cont}}, T_{\text{stop}}$ be the sets of primitive tasks in the three task networks of H. Then we can compute a STRIPS planning problem Π in polynomial time such that for all $r \in \mathbb{N}$ the following are equivalent:

1. Éloise wins $\mathbb{G}\Big(\Pi, 2\big(|T_0| + r|T_{\text{cont}}| + |T_{\text{stop}}|\big)\Big)$.
2. H can be solved by applying cont a total of r times before applying stop.

Proof. The execution of a primitive task in a plan for H corresponds to two plies in the planning game. Éloise chooses the task and Abélard chooses the effect.

Assume w.l.o.g. that T_0, T_{cont} and T_{stop} are pairwise disjoint. The variables of Π are the variables of H and $\text{do}(t), \text{done}(t)$ for all $t \in T := T_0 \sqcup T_{\text{cont}} \sqcup T_{\text{stop}}$. Write

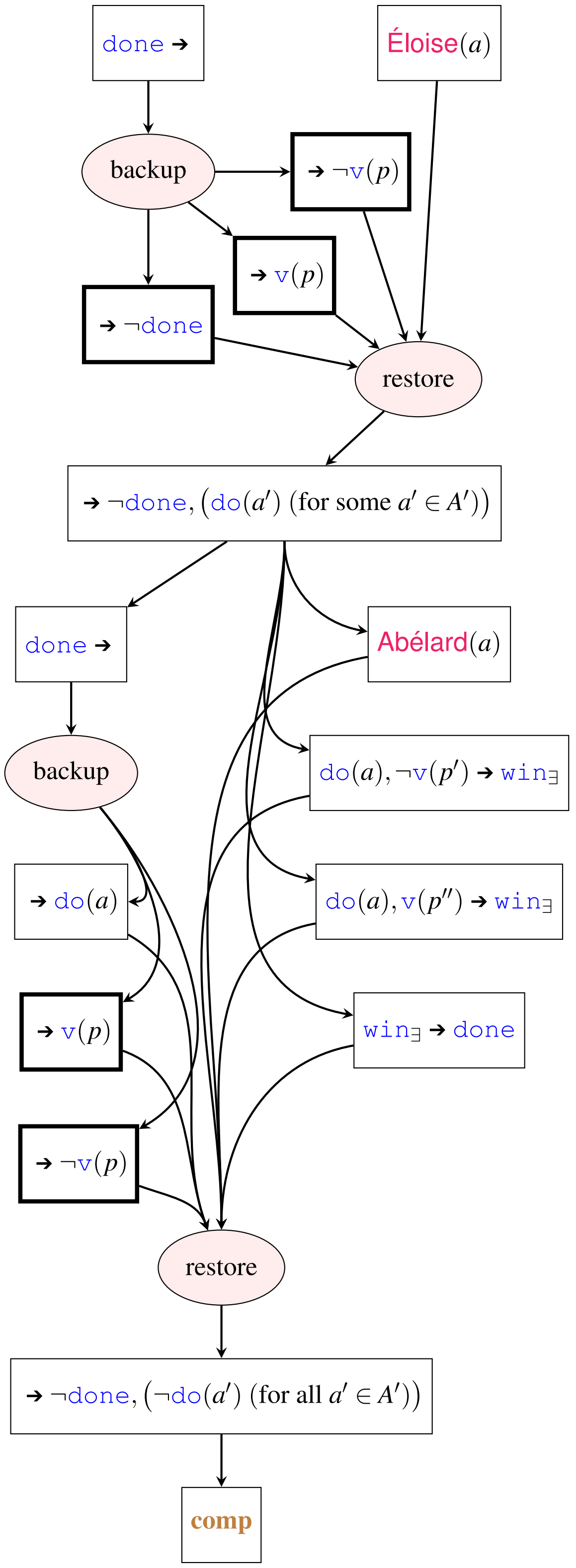

Figure 3: The non-primitive method task network in the proof of Thm. 11. Include tasks for all $a \in A'$ and $p \in P'$ and positive preconditions p' and negative preconditions p'' of a. Include enough (e.g., $|A'|$ many) copies of the tasks marked with thick border.

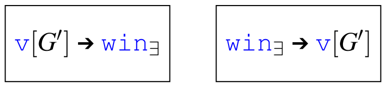

Figure 4: The primitive method task network in the proof of Thm. 11.

$\text{done}[T'] = \{\text{done}(t) : t \in T'\}$ and sim. for do, for $T' \subseteq T$. For each $t \in T$ with name pr, construct some actions of Π:

- $t_{\text{Éloise}}$ has the preconditions of the action associated with 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 only 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[2] $\text{done}[T_{\text{cont}}]$ (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 Π consumes $\text{do}(t)$, adds e_+ and deletes e_-.

The initial state of Π is the initial propositional state of H together with $\text{done}[T_{\text{cont}}]$, and the goal of Π is the set of all variables $\text{done}(t)$.

The only possible 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} in which case she uses $t^{\circlearrowleft}_{\text{Éloise}}$), and Abélard replies with actions $t^e_{\text{Abélard}}$ mimicing non-deterministic 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 this setting in terms of FOND HTN planning.

As an example application, consider the variant of Chess in which players are allowed to move into check and kings cannot be captured. Players can even continue moving after a player gets checkmated. If after a predetermined odd number of plies Black is checkmated, White wins the game; otherwise Black wins the game. Such a game can be modeled as a planning game. Assuming that White can choose the game length at the time of making their first move, by Thm. 5 it is in EXPSPACE to decide whether White has a winning strategy in a given position on an $n \times n$ board. We conjecture that this upper bound is tight. In comparison, standard Chess is EXPTIME-complete (Fraenkel and Lichtenstein 1981).

References

Camacho, A.; Baier, J. A.; Muise, C.; and McIlraith, S. A. 2018. Finite LTL Synthesis as Planning. In *Proceedings of*

[2]Consuming means requiring and deleting.

the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS-18), 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*, 74–84.

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*, 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: 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 – 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-16)*, 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.