Chronicles for Representing Hierarchical Planning Problems with Time

Roland Godet, Arthur Bit-Monnot

Chronicles for Representing Hierarchical Planning Problems with Time

Roland Godet,[1,2] **Arthur Bit-Monnot**[1]

[1] LAAS-CNRS, Université de Toulouse, INSA, CNRS, Toulouse, France
[2] ENS Paris-Saclay, Université Paris-Saclay, France
rgodet@laas.fr, abitmonnot@laas.fr

Abstract

In temporal planning, chronicles can be used to represent the predictive model of durative actions. Unlike the classical state-oriented representation, the usage of chronicles allows a rich temporal qualification of conditions and effects, beyond the mere start and end times of an action.

In this paper we propose an extension of the standard chronicle representation to support hierarchical problems. In particular, we show that the addition of temporally qualified subtasks to chronicles makes them suitable to represent not only primitive actions but also HTN methods.

We show how the set of solutions to a chronicle-based hierarchical problem can be quite naturally represented as a Constraint Satisfaction Problem (CSP). To associate semantics to this extended chronicle representation, we propose a set of rules that must hold for any solution to the hierarchical problem, specified as constraints on the associated CSP.

Introduction

In Artificial Intelligence, planning with Hierarchical Task Networks (HTNs) is an approach to automated planning where actions are hierarchically structured (Erol, Hendler, and Nau 1994; Höller et al. 2020) with compound tasks, which can be broken down into subtasks. This hierarchy of tasks allows the planning problem to be described at several levels, starting with more abstract tasks and ending with directly applicable primitive tasks.

In its simplest expression, an HTN (Erol, Hendler, and Nau 1994; Höller et al. 2020) planning problem consists of *(i)* an initial state, *(ii)* an initial task network describing the aim, *(iii)* a set of actions, *(iv)* a set of compound tasks, and *(v)* a set of methods.

A *state* defines the values of a set of *state variables*, each describing a specific attribute of the environment (e.g. the position of a truck).

A *task network* is a set of *tasks* and *constraints*. Each task describes a particular operation to be fulfilled (e.g. bring a package from A to B). It is composed of its name and a list of parameters, which may be variables or constants. There are two kinds of tasks: the *actions* (or *primitive* tasks), which can be directly executed, and the *compound* tasks, that the planner must refine into actions. The constraints might, e.g., restrict the value of some variables or the order of tasks.

An *action* consists of *(i)* a set of conditions over state variables, they characterize the set of states in which the action is applicable, and *(ii)* a set of effects on state variables, that update the state to reflect the consequences of the action.

A *method* is a pair $m = (t_c, tn)$, where t_c is a compound task and *tn* is a task network. It represents the fact that one way to perform t_c is to execute the tasks laid out in *tn*.

Given an initial state s_0, an initial task network *tn*, a set of actions, and a set of methods, a *plan* is a sequence of actions $\langle a_1, ..., a_n \rangle$. To produce a plan, a planner must systematically replace any compound task in the initial task network by the subtasks of a compatible method, repeating the process until only primitive tasks remain. Along with ordering, this approach defines the set of candidate plans as the ones that can be decomposed from the initial task network. A candidate plan is a solution of the planning problem if the corresponding action sequence is applicable in the initial state.

Approach Rather than this state-oriented view, another representation for planning follows a time-oriented view, as proposed with chronicles (Ghallab, Nau, and Traverso 2004; Bit-Monnot 2018) or timelines (Smith, Frank, and Jónsson 2000). Several works have considered the introduction of hierarchies for time-oriented planners, notably FAPE (Bit-Monnot et al. 2020) and CHIMP (Stock et al. 2015). The ANML language is also a very relevant proposal, but lacks clearly defined semantics (Smith, Frank, and Cushing 2007).

In this paper, we introduce *hierarchical chronicles*, an extension of the chronicles proposed by Bit-Monnot (2018) for generative planning. This is done by associating a chronicle with the task it achieves as well as the subtasks it requires. We show this to be sufficient for chronicles to represent HTN methods, in addition to HTN actions.

While conceptually simple, our formalization of the resulting problem as a CSP allows for advanced temporal features (e.g. intermediate conditions and effects) and closely relates to the time-oriented representation of scheduling problems which we hope will facilitate the application of scheduling techniques to hierarchical planning.

Hierarchical Chronicles

A *type* is defined by a set of values. It can be either a set of domain constants (e.g. the type *Truck* $= \{ R_1, R_2 \}$ defines two truck objects R_1 and R_2) or a discrete set of numeric

values. A type of particular interest for this formalization is the set of *timepoints* that we assume to be a discrete set of evenly spaced numeric values representing absolute times at which events can occur.[1] A decision *variable* is related to a type which defines its initial domain (i.e. possible values).

A *state variable* describes the progression of a specific attribute of the environment over time. It is generally parameterized by one or multiple domain objects. For instance $loc(R_1)$ denotes the evolution of the position of the truck R_1 over time. A state variable expression is often parameterized by variables, in which case the particular state variable it refers to depends on the value taken by its parameters, e.g., $loc(r)$ will correspond to $loc(R_1)$ or $loc(R_2)$ depending on the value taken by the variable r of type *Truck*.

A *task* denotes a particular operation to be fulfilled in the environment over time. It is generally parameterized by one or multiple domain objects. For instance $[5, 10]Go(R_1, L_1)$ denotes the operation of moving the truck R_1 to the location L_1 over the temporal interval $[5, 10]$. A task might be parameterized by variables, in which case the particular task it refers to depends on the value taken by its parameters.

A *chronicle* can be thought of as defining the requirements of a process in the planning problem, and in particular the process of executing an action or carrying out a method. A chronicle is a tuple $\mathcal{C} = (V, \tau, X, C, E, S)$ where:

- V is a set of **variables** appearing in the chronicle, partitioned into a set of temporal variables V_T whose domains are timepoints and a set of non-temporal variables V_O.
- τ is the **task achieved by the chronicle**. It is of the form $[s, e]task(x_1, \ldots, x_n)$ where s and e are temporal variables in V_T, $task(x_1, \ldots, x_n)$ is the parameterized task (with each $x_i \in V_O$). s and e respectively denote the start and end instants at which the chronicle is active, and we will refer to them as $start(\mathcal{C})$ and $end(\mathcal{C})$ respectively.
- X is a set of **constraints** over the variables in V.
- C is a set of **conditions**, Each condition is of the form $[s, e]var(p_1, \ldots, p_n) = v$ where s and e are variables in V_T, $var(p_1, \ldots, p_n)$ is a parameterized state variable (with each $p_i \in V_O$) and v is a variable in V_O. A condition states that the state variable $var(p_1, \ldots, p_n)$ must have the value v over the temporal interval $[s, e]$.
- E is a set of **effects**. Each effect is in the form of $[s, e]var(p_1, \ldots, p_n) \leftarrow v$ where s and e are variables in V_T, $var(p_1, \ldots, p_n)$ is a parameterized state variable (with each $p_i \in V_O$) and v is a variable in V_O. An effect states that the state variable $var(p_1, \ldots, p_n)$ takes the value v at time e. Over the temporal interval $]s, e[$, the state variable is transitioning from its previous value to v, and has an undetermined value.
- S is a set of **subtasks**. Each subtask has the form $[t_s, t_e]task(x_1, \ldots, x_n)$ where t_s and t_e are temporal variables in V_T and $task(x_1, \ldots, x_n)$ is a parameterized task (with each $x_j \in V_O$). It denotes a required task that must be achieved by another chronicle over $[t_s, t_e]$.

A planning problem defines a set of *chronicle templates* $\mathcal{T}$ where each template can be instantiated into a *chronicle instance* by substituting all variables in the template with fresh variables. We typically consider two types of chronicles templates: *action chronicles* that have effects but no subtasks, and *method chronicles* with subtasks but no effects.

Considering a method template *Deliver* $\in \mathcal{T}$ that delivers a package p from a position l_s to a position l_e with a truck r, it can be instantiated as the method chronicle $\mathcal{C}^1_{Deliver}$ where:

- $\tau = [t_s, t_e]$*Deliver*(p, l_e), i.e., this chronicle should result in delivering the package p to l_e over the temporal interval $[t_s, t_e]$.
- $C = \{\ [t_s, t_s]loc(p) = l_s, [t_s, t_s]loc(r) = l_s\ \}$, i.e., the package p and the truck r have to be at the starting location l_s at the beginning of the method.
- $E = \varnothing$, this is a method chronicle, with no direct effects.
- $S = \{\ [t^L_s, t^L_e]$ *Load*(p, r), $[t^M_s, t^M_e]$ *Move*(r, l_s, l_e), $[t^U_s, t^U_e]$ *Unload*$(p, r)\ \}$, i.e., to achieve this delivery action, the following tasks have to be done: loading the package in the truck, moving the truck from l_s to l_e, and unloading the package.
- $V_O = \{\ p, r, l_s, l_e\ \}$ are the parameters of the method (package, truck, start and end location) and $V_T = \{\ t_s, t_e, t^L_s, t^L_e, t^M_s, t^M_e, t^U_s, t^U_e\ \}$ where t_s, t_e are timepoints representing the start and the end of the method (from τ), and $t^L_s, \ldots, t^U_e$ are the corresponding start/end of subtasks.
- $X = \{\ t_e \leq t_s + 10, l_s \neq l_e, t^L_e \leq t^M_s, \ldots\ \}$, e.g., impose that the method should take no more than 10 units of time, the two locations must be different and the *Move* subtask has to be executed after the $Load$ subtask.

Considering the action template *Move* $\in \mathcal{T}$ that moves a truck r from l_s to l_e, it can be instantiated as a chronicle $\mathcal{C}_{Move}$ with the following effects: $E = \{\, [t_s, t_e]loc(r) \leftarrow l_e\,\}$, i.e., the truck r will be at the ending location l_e at the end of the action, but its position is unknown during the action execution. This chronicle $\mathcal{C}_{Move}$ achieves the eponymous task $[t_s, t_e]$*Move*(r, l_s, l_e).

We distinguish an *initial chronicle* $\mathcal{C}_0$ encoding the initial state as effects and the objectives of the planning problem as conditions and subtasks. It might also specify the anticipated evolution of the environment outside the planner's control, e.g., that a bus will pass at 6pm. This chronicle is the only one not associated to a meaningful task τ. For instance, the problem where the package P_1 is initially in location L_0 and must be brought to location L_1 or L_2 before time 50, can be described by the following initial chronicle $\mathcal{C}_0$:

- $V_O = \{\ l\ \}, V_T = \{\ t_s, t_e\ \}$
- $X = \{\ t < 50, l = L_1 \vee l = L_2\ \}$, constraints restricting the solution set.
- $C = \{\ [t_e, t_e]loc(R_1) = l\ \}$, specifying the goals.
- $E = \{\, [0, 0]loc(P_1) \leftarrow L_0, [0, 0]loc(R_1) \leftarrow L_0\,\}$, effects specifying the initial state.

[1] While this definition assumes a discrete time representation, it could be equally interpreted with a continuous time representation. Also note that discrete time is no less expressive when instantaneous changes are forbidden, as common in temporal planning in general and PDDL in particular (Cushing 2012). In general however the computational complexity might differ between a discrete and a continuous time representation.

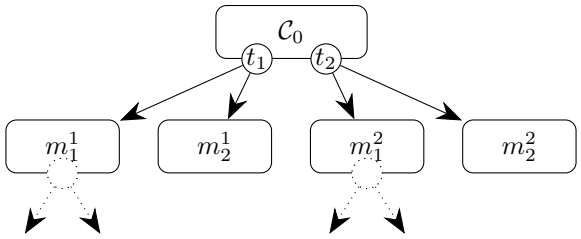

Figure 1: Decomposition graph resulting from the expansion of two tasks $t_1 : t(x)$ and $t_2 : t(y)$

- $S = \{\ [t_s, t_e]\textit{Deliver}(P_1, l)\ \}$, subtasks specifying specific tasks to be achieved by a solution plan.

A planning problem is the association $(\mathcal{C}_0, \mathcal{T})$ of an initial chronicle, defining the initial state and objectives, with a set of chronicle templates which can be instantiated, defining usable actions and methods.

Planning Problem as a CSP

Problem Instantiation

At this point we have defined the initial chronicle (representing the problem) and a set of chronicle templates (representing possible actions and methods). We now introduce the procedure to build a set of chronicle instances Π that can be used to represent a solution.

We initially set $\Pi = \{\ \mathcal{C}_0\ \}$, i.e., limited to the initial chronicle. Suppose the initial chronicle $\mathcal{C}_0$ contains two subtasks $t_1 : t(x)$ and $t_2 : t(y)$ and that the task $t(\cdot)$ can be achieved by one of two methods m_1 and m_2 (i.e. the task of m_1/m_2 are of the form $[\cdot, \cdot]t(\cdot)$). As the first task t_1 might be achieved by either m_1 or m_2, we add two fresh instantiations m_1^1 and m_2^1 to Π. We record the motivation for the introduction of both chronicles in a lookup-table $decomposes(m_1^1) \leftarrow t_1$ and $decomposes(m_2^1) \leftarrow t_1$. Similarly, we introduce two distinct chronicle instances m_1^2 and m_2^2 to represent the possible refinements of t_2. This process can be seen as creating a decomposition graph such as the one in Figure 1.

We refer to this process as *chronicle expansion*. More formally, for each subtask t of a chronicle instance $\mathcal{C} \in \Pi$, and each chronicle template $\mathcal{T}_i \in \mathcal{T}$ whose task is unifiable with t, we instantiate a new chronicle instance $\mathcal{T}_i^k$ and add it to Π. Each instantiated chronicle is uniquely associated to the task it was introduced to decompose with the $decomposes$ lookup table, effectively defining a decomposition tree. Any chronicle instance added to Π as a result of expansion will need to be expanded itself, making this a recursive process. In the case of cyclic HTN problems, the depth of resulting decomposition tree might not be bounded. For practical purpose – e.g. to encode the problem in a CSP solver – one might decide to bound the depth of the tree to obtain a finite number of chronicles.

Of course, it is not the case that each chronicle instance will be part of the solution plan. In Figure 1, the m_1^1 and m_1^2 chronicle instances are mutually exclusive as only one method can be used to decompose the t_1 task. To capture this fact, we associate each a chronicle instance $\mathcal{C} \in \Pi$ to a boolean variable *present* $(\mathcal{C})$ that is true $(\top)$ if $\mathcal{C}$ is present in the solution plan and false $(\bot)$ otherwise. Note that the initial chronicle must always be present, so *present* $(\mathcal{C}_0)$ is always true.

With this process, we have created a set of chronicle instances Π, representing all possible methods and actions that might appear in the plan. Each such chronicle instance $\mathcal{C} \in \Pi$ is associated with a boolean variable *present* $(\mathcal{C})$ that represents whether it is part of a solution, and a task $decomposes(\mathcal{C})$ that it might decompose. We now ought to define what are the constraints that must hold for this set of optional chronicle instances to form a solution to the original planning problem. Before doing so, we introduce a synthetic representation for conditions, effects and tasks that will allow us to specify their behavior independently of the context in which they appear.

Core structures

Considering a planning problem Π, the core structures to express the constraints it must fulfill are described here.

Condition Token Given a chronicle instance $\mathcal{C} \in \Pi$, each condition in $\mathcal{C}$ is associated to a *condition token*:

$$\textit{present}\,(\mathcal{C}) : [s, e]var(p_1, \ldots, p_n) = v$$

This token states that, if $\mathcal{C}$ is present in the solution plan (*present* $(\mathcal{C}) = \top$), then the state variable $var(p_1, \ldots, p_n)$ must have the value v over the temporal interval $[s, e]$. The set of condition tokens in Π is denoted C_Π.

Effect Token Given a chronicle instance $\mathcal{C} \in \Pi$, each effect in $\mathcal{C}$ is associated to an *effect token*:

$$\textit{present}\,(\mathcal{C}) : [s, e, t]var(p_1, \ldots, p_n) \leftarrow v$$

Note the new temporal variable $t \in V_T$. This token states that, if $\mathcal{C}$ is present in the solution plan (*present* $(\mathcal{C}) = \top$), then the state variable $var(p_1, \ldots, p_n)$ is undefined over the temporal interval $]s, e[$ and has the value v over the temporal interval $[e, t]$. The expansion with the new timepoint t allows us to encode a minimal persistence of the effect until a later time t. The set of effect tokens in Π is denoted E_Π.

Task Token Given a chronicle instance $\mathcal{C} \in \Pi$, each subtask in $\mathcal{C}$ is associated to a *task token*:

$$\textit{present}\,(\mathcal{C}) : [s, e]task(x_1, \ldots, x_n)$$

This token states that, if $\mathcal{C}$ is present in the solution plan (*present* $(\mathcal{C}) = \top$), then the task $task(x_1, \ldots, x_n)$ must be achieved over the temporal interval $[s, e]$. The set of task tokens in Π is denoted Γ_Π.

Token Characteristics Effect tokens here address the evolution of state variables over time. Each (present) effect token forces another value to its state variable, which is compelled to be maintained within a given temporal interval, thus encoding the state evolution. Condition tokens put requirements on the state evolution by imposing a state variable to have a given value over a temporal interval.

Each token (condition, effect, and task) is present in the solution plan if and only if its associated chronicle instance

is present in the solution plan. Given a token (condition, effect, or task) ω from a chronicle instance $\mathcal{C} \in \Pi$, we have $\mathit{present}(\omega) = \mathit{present}(\mathcal{C})$.

Constraints

Considering a planning problem Π, the constraints used to encode the consistency of a plan are described here.[2]

Coherence Constraint A state variable cannot take several values at the same time. This implies that for two distinct effect tokens ε and ε' in E_Π to be *coherent*, they may not concurrently impose a value or transition to the same state variable.

$$\text{Given } \begin{cases} \varepsilon = \langle p : [s, e, t] var(p_1, \dots, p_n) \leftarrow v \rangle \in E_\Pi \\ \varepsilon' = \langle p' : [s', e', t'] var(p'_1, \dots, p'_n) \leftarrow v' \rangle \in E_\Pi \end{cases}$$

the constraint $\mathit{coherent}(\varepsilon, \varepsilon')$ is defined as:

$$p \wedge p' \implies t \leq s' \vee t' \leq s \vee p_1 \neq p'_1 \vee \dots \vee p_n \neq p'_n$$

By forcing two effect tokens to be non overlapping (over the presence p, time $[s, t]$ and the state variable $var(p_1, \dots, p_n)$ dimensions), this constraint ensures that a state variable will be given at most one value at any timepoint.

Support Constraint A condition token $\beta \in C_\Pi$ is said *supported by* an effect token $\varepsilon' \in E_\Pi$ if this effect establishes the value required by β and this value persists for the span of β.

$$\text{Given } \begin{cases} \beta = \langle p : [s, e] var(p_1, \dots, p_n) = v \rangle \in C_\Pi \\ \varepsilon' = \langle p' : [s', e', t'] var(p'_1, \dots, p'_n) \leftarrow v' \rangle \in E_\Pi \end{cases}$$

the constraint $\mathit{supported\text{-}by}(\beta, \varepsilon')$ is defined by:

$$p' \wedge e' \leq s \wedge e \leq t' \wedge p_1 = p'_1 \wedge \dots \wedge p_n = p'_n \wedge v = v'$$

A condition token $\beta \in C_\Pi$ is said *supported* if it is supported by at least one effect token in E_Π. More formally, the constraint $\mathit{supported}(\beta)$ is defined by:

$$\mathit{present}(\beta) \implies \bigvee_{\varepsilon \in E_\Pi} \mathit{supported\text{-}by}(\beta, \varepsilon)$$

Refinement Constraint A task token $\gamma \in \Gamma_\Pi$ is said *refined by* a chronicle instance $\mathcal{C} \in \Pi$ if *(i)* $\mathcal{C}$ was introduced as a potential decomposition of γ, *(ii)* it is part of the solution and *(iii)* the task it achieves is identical to the one of γ. Given

$$\gamma = \langle p : [s, e] task(x_1, \dots, x_n) \rangle \in \Gamma_\Pi$$

and $\mathcal{C} \in \Pi$ whose achieved task is of the form,

$$task(\mathcal{C}) = \langle [s', e'] task(x'_1, \dots, x'_n) \rangle$$

the constraint $\mathit{refined\text{-}by}(\gamma, \mathcal{C})$ is defined by:

$$p \wedge \mathit{decomposes}(\mathcal{C}) = \mathit{id}(\gamma) \wedge s = s' \wedge e = e' \wedge x_1 = x'_1 \wedge \dots \wedge x_n = x'_n$$

where $id(\gamma)$ uniquely identifies the task associated to γ.

A task token $\gamma \in \Gamma_\Pi$ is said *refined* if it is refined by a chronicle instance in Π. More formally, the constraint $\mathit{refined}(\gamma)$ is defined by:

$$\mathit{present}(\gamma) \implies \bigvee_{\mathcal{C} \in \Gamma_\Pi} \mathit{refined\text{-}by}(\gamma, \mathcal{C})$$

[2]Note that the coherence and support constraints are identical to the ones of Bit-Monnot (2018) and repeated here for completeness.

Motivation constraint For an HTN problem it is normally the case that a method or action is only allowed to be part of the solution if it derives from the initial task network. In a sense, it means that the presence of a chronicle must be motivated by the achievement of a task higher up in the hierarchy.

Consider a chronicle instance $\mathcal{C} \in \Pi$ that was introduced to decompose a task γ (i.e. $\mathit{decomposes}(\mathcal{C}) = id(\gamma)$). Then the $\mathit{motivated}(\mathcal{C})$ constraint is expressed as:

$$\mathit{present}(\mathcal{C}) \implies \mathit{present}(\gamma) \wedge \mathit{refined\text{-}by}(\gamma, \mathcal{C}) \bigwedge_{\mathcal{C}' \in \mathit{siblings}(\mathcal{C})} \neg \mathit{present}(\mathcal{C}')$$

where $\mathit{siblings}(\mathcal{C})$ is the set of all other chronicle instances $\mathcal{C}'$ that were introduced as a potential decomposition of the same task γ. This constraint effectively enforces that, if present, a chronicle instance uniquely achieves a task higher-up in the hierarchy.

Internal Chronicle Consistency Considering a chronicle $\mathcal{C} \in \Pi$, all its constraints X must be verified if $\mathcal{C}$ is part of the solution. It is represented by the constraint $\mathit{consistent}(\mathcal{C})$:

$$\mathit{present}(\mathcal{C}) \implies \bigwedge_{x \in X} x$$

Any requirement regarding a chronicle structure should be explicitly or implicitly encoded in the set X of chronicle constraints. In particular, a common requirement for hierarchical problems is that a method spans exactly the same time interval as its subtasks, i.e., that for any method chronicle $\mathcal{C}$ with a non-empty set of subtasks $subtasks(\mathcal{C})$:

$$start(\mathcal{C}) = \min_{st \in subtasks(\mathcal{C})} start(st)$$
$$end(\mathcal{C}) = \max_{st \in subtasks(\mathcal{C})} end(st)$$

Likewise, a method chronicle with no subtasks should be instantaneous (i.e. $start(\mathcal{C}) = end(\mathcal{C})$).

Formulation as a CSP Finally, a planning problem Π can be encoded as a CSP with variables V_Π and constraints X_Π where:

$$\begin{aligned} V_\Pi &= \{ V \mid (V, \tau, X, C, E, S) \in \Pi \} \\ &\cup \{ \mathit{present}(\mathcal{C}) \mid \mathcal{C} \in \Pi \} \\ X_\Pi &= \{ \mathit{coherent}(\varepsilon, \varepsilon') \mid (\varepsilon, \varepsilon') \in E_\Pi^2, \varepsilon \neq \varepsilon' \} \\ &\cup \{ \mathit{supported}(\beta) \mid \beta \in C_\Pi \} \\ &\cup \{ \mathit{refined}(\gamma) \mid \gamma \in \Gamma_\Pi \} \\ &\cup \{ \mathit{motivated}(\mathcal{C}) \mid \mathcal{C} \in (\Pi \setminus \{ \mathcal{C}_0 \}) \} \\ &\cup \{ \mathit{consistent}(\mathcal{C}) \mid \mathcal{C} \in \Pi \} \end{aligned}$$

Solution and Plan Extraction A solution to a hierarchical planning problem Π is an assignment to all variables in V_Π that satisfies all constraints in X_Π. The decision process involved in building the assignment will effectively impose the presence of methods and actions (through presence variables) as well as their instantiation (through parameter variables) and orderings (through temporal variables). We say that a chronicle $\mathcal{C} \in \Pi$ is present in the solution if its presence variable $\mathit{present}(\mathcal{C})$ evaluates to true given the assignment.

From a solution assignment, it is straightforward to extract a solution plan: any action chronicle $\mathcal{C} \in \Pi$ present in the solution corresponds to an action in the plan. The value of its parameters as well as start and end times are given by the value of the corresponding variables in the assignment.

Likewise, the assignment encodes a full decomposition from the initial task network into the solution plan as the *refinement* and *motivation* constraints ensure that, for any subtask of a chronicle present in the solution, there is exactly one refining chronicle (action or method) present in the solution.

It should be noted that setting the presence variable of a chronicle to false has the same effect as not including it in the set Π of chronicles considered in the CSP. This suggests that bounding the depth of a decomposition tree should be interpreted as a decision restricting the set of solution. To maintain the completeness of a decision procedure, this bound should be reconsidered (i.e. increased) when it is shown that no solution exists within a given depth.

Discussion and Related Work

To the best of our knowledge all existing temporal HTN planners, including FAPE (Bit-Monnot et al. 2020), CHIMP (Stock et al. 2015) and SIADEX (Castillo et al. 2006), interpret HTN planning as a constructive process: they start from an initial task network that is iteratively expanded into a solution, each expansion bringing its own new subtasks, actions and additional constraints. This proposal adopts a different interpretation of hierarchical planning as a constraint satisfaction problem. While the two interpretations remain compatible, a key feature of the CSP interpretation is its proximity with the representation of scheduling problems. In particular, the notion of optional time intervals in scheduling solvers such as CPOptimizer (Laborie et al. 2018) brought many modeling capabilities that have a clear mapping with the structure of HTN problem (e.g. the *alternative* and *span* constraints of Laborie and Rogerie, 2008).

The closest formalism is the one used by FAPE (Bit-Monnot et al. 2020) that supports both hierarchical and generative planning. In FAPE this is allowed by annotating some actions as *task-dependent*, meaning that they can only be introduced as a refinement of an existing task. Unlike FAPE, this proposal focuses on pure hierarchical (HTN-like) planning allowing to remove this distinction. The addition of explicit temporal variables representing the end of the persistence of an effect is a subtle change that avoids an explicit handling of *causal-links* and *threats* (threats being particularly problematic for scalability as they might involve any combination of two effects and one condition, leading to a cubic number of constraints).

The notion of tokens (including the effect persistence) is inspired by the homonymous tokens of timeline-based planners such as Europa (Barreiro et al. 2012) or CHIMP (Stock et al. 2015). Unlike timeline-based planners however, we do maintain a distinction between conditions and effects and use an action-centric formalism.

Several hierarchical planners such as SIADEX (Castillo et al. 2006) or SHOP2 extensions (Goldman 2006) have enabled temporal features in a state-progression setting by timestamping states. Relationships with these state-oriented representations are less obvious as this paper adopts a time-oriented view where the evolution of each state variable is handled independently of the others.

Conclusion

In this paper, we introduced a constraint-based representation for hierarchical planning under temporal constraints. This encoding assumes a time-oriented view and a fully lifted representation. Its foundation on previous chronicle approaches allows leveraging their temporal expressiveness with a very limited increase in complexity.

While the paper is restricted to the discussion of the planning formalism, our current work is focused on the exploitation of this formalism and associated encoding in a constraint-based planner that leverages scheduling techniques.

References

Barreiro, J.; Boyce, M. E.; Do, M. B.; Frank, J. D.; Iatauro, M.; Kichkaylo, T.; Morris, P. H.; Ong, J. C.; Remolina, E.; Smith, T. B.; and Smith, D. E. 2012. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization.

Bit-Monnot, A. 2018. A Constraint-Based Encoding for Domain-Independent Temporal Planning. *CP*.

Bit-Monnot, A.; Ghallab, M.; Ingrand, F.; and Smith, D. E. 2020. FAPE: a Constraint-based Planner for Generative and Hierarchical Temporal Planning. *ArXiv*, abs/2010.13121.

Castillo, L.; Fdez-Olivares, J.; García-Pérez, Ó.; and Palao, F. 2006. Temporal Enhancements of an HTN Planner. In Marín, R.; Onaindía, E.; Bugarín, A.; and Santos, J., eds., *Current Topics in Artificial Intelligence*, 429–438.

Cushing, W. A. 2012. *When is Temporal Planning Really Temporal?* Ph.D. thesis, Arizona State University.

Erol, K.; Hendler, J.; and Nau, D. 1994. HTN planning: Complexity and Expressivity. *AAAI*, 2.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Goldman, R. P. 2006. Durative Planning in HTNs. In *ICAPS*.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. *AAAI*.

Laborie, P.; and Rogerie, J. 2008. Reasoning with Conditional Time-Intervals. In *FLAIRS Conference*.

Laborie, P.; Rogerie, J.; Shaw, P.; and Vilím, P. 2018. IBM ILOG CP Optimizer for Scheduling. *Constraints*.

Smith, D. E.; Frank, J.; and Cushing, W. 2007. The ANML Language.

Smith, D. E.; Frank, J. D.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review*, 15: 47 – 83.

Stock, S.; Mansouri, M.; Pecora, F.; and Hertzberg, J. 2015. Hierarchical Hybrid Planning in a Mobile Service Robot. In *KI*.