Compiling Away Uncertainty in Strong Temporal Planning with Uncontrollable Durations

Andrea Micheli
FBK and University of Trento
Trento, 38123, Italy
amicheli@fbk.eu

Minh Do and **David E. Smith**
NASA Ames Research Center
Moffet Field, CA, 94035, USA
{minh.do, david.smith}@nasa.gov

Abstract

Real world temporal planning often involves dealing with uncertainty about the duration of actions. In this paper, we describe a sound-and-complete compilation technique for strong planning that reduces any planning instance with uncertainty in the duration of actions to a plain temporal planning problem without uncertainty.

We evaluate our technique by comparing it with a recent technique for PDDL domains with temporal uncertainty. The experimental results demonstrate the practical applicability of our approach and show complementary behavior with respect to previous techniques. We also demonstrate the high expressiveness of the translation by applying it to a significant fragment of the ANML language.

1 Introduction

For many real world planning problems there is uncertainty about the duration of actions. For example, robots and rovers have transit times that are uncertain due to terrain, obstacle avoidance, and traction. There is also uncertainty in the duration of manipulation and communication tasks. When there are no time constraints, and plan duration is unimportant, this uncertainty can often be ignored. However, if there are exogenous events that affect action conditions, or time-constrained goals, action durations and uncertainty must be considered.

In general, temporal conditional planning is very hard, particularly for actions with duration uncertainty [Younes and Simmons, 2004; Mausam and Weld, 2008; Beaudry *et al.*, 2010]. In practice, most practical planners take one of two much simpler approaches: 1) plan using expected action durations, and rely on runtime replanning and plan flexibility to deal with actions that take more or less time than expected, or 2) plan using worst case action durations. The first of these approaches is risky – there is no guarantee that the plan will succeed or that runtime replanning can achieve the goals. The second approach, while generally more conservative, can also fail if there are time constraints or goals with lower bounds (e.g. an action should not be completed, or a goal should not be completed before some particular time).

Recently, [Cimatti *et al.*, 2015] addressed these issues by explicitly modeling duration range for actions, and devising a planner that soundly reasons to produce robust plans. In that work, the authors introduced the "Strong Planning Problem with Temporal Uncertainty" (SPPTU) as the problem of finding a sequence of action instances and fixed starting times, such that for every possible duration of each action in the plan, the plan is valid and leads to the goal. In this work, we address the same problem, but consider a much richer language for representing temporal planning domains with action duration uncertainty. We use a variable-value language allowing effects at arbitrary time points during an action and durative conditions over arbitrary sub-intervals of actions. We address the SPPTU by automatically translating a planning instance with uncertainty on action durations into a plain temporal planning problem with controllable action durations. We exploit all the features in the planning language to cast the temporal uncertainty in action durations into discrete uncertainty over the problem variables. This compilation enables the solution of SPPTU using existing techniques and tools for temporal planning.

We also present an experimental evaluation of the compilation technique showing that it can be practically applied on very expressive domains.

Related Work. Temporal uncertainty is a well-understood concept in scheduling and has been widely studied [Morris, 2006; Santana and Williams, 2012; Muise *et al.*, 2013; Cimatti *et al.*, 2014]. The problem we address can be seen as a generalization of Strong Controllability for Temporal Problems [Vidal and Fargier, 1999; Peintner *et al.*, 2007] to planning rather than scheduling. Dealing with planning is harder because the actions (and thus the time points associated with them) in a plan are not known a-priori and must be discovered; moreover, causal relationships are much more complex.

In temporal planning, duration uncertainty is a known challenge [Bresina *et al.*, 2002], but few temporal planners address it explicitly. Some temporal planners [Frank and Jónsson, 2003; Cesta *et al.*, 2009] cope with this issue by generating flexible temporal plans: instead of fixing the execution time of each action, they return a (compactly represented) set of plans that must be scheduled at run-time by the plan executor. This approach cannot guarantee plan executability and goal achievement at runtime, because there is no formal modeling of the boundaries and contingencies in which the system is going to operate. In addition, this requires that the executor be able to schedule activities at runtime. Flexibility and controllability are complementary: controllability provides guarantees with respect to the modeled uncertainty, while flexibility allows the plan to be adjusted during execution. In principle, we can use any temporal planner (e.g., VHPOP) that can generate flexible plans in combination with our compilation to generate a flexible strong plan. IxTeT [Ghallab and Laruelle, 1994] was the first attempt to apply the results in temporal reasoning under uncertainty to planning, but the planner demanded the scheduling of a Simple Temporal Network with Uncertainty (STNU) [Vidal and Fargier, 1999] by the plan executor. Here, we aim at generating plans that are guaranteed to work regardless of the temporal uncertainty. IxTeT deals with dynamic controllability: it generates a strategy for scheduling the actions depending on observations. Although these plans can work in more situations, they are also complex to generate, understand, and execute. Strong plans are required for safety critical systems like space applications, where guarantees are needed, and computational power is limited.

In contrast to [Beaudry *et al.*, 2010] we are concerned with qualitative uncertainty: we are not dealing with probability distributions, but only with durations that are bounded in convex intervals. We aim to guarantee goal achievement, while Beaudry *et al.* maximize the probabilistic expectation.

There is a clear parallel between the problem we are solving and conformant planning [Ghallab *et al.*, 2004]. In this sense, our work is similar to [Palacios and Geffner, 2009] in which the authors transform conformant planning into deterministic planning, although the translation is very different.

The closest works to ours are Cimatti *et al.* (2013, 2015). In the former, the authors present a logical characterization of the SPPTU for timelines with temporal uncertainty, as well as a first-order encoding of the problem having bounded horizon. Cimatti *et al.* (2015) cast this in PDDL by extending state-space temporal planning. In this paper, we generalize both these frameworks – we do not impose any bounded horizon for the problem and we consider a more expressive language allowing disjunctive preconditions, effects at arbitrary time points during actions and durative conditions on arbitrary sub-intervals. In Section 4 we provide a comparison with the techniques proposed in [Cimatti *et al.*, 2015].

2 Modeling Duration Uncertainty

In [Cimatti *et al.*, 2015], the authors propose an extension of PDDL 2.1 to model actions with uncontrollable duration. In this paper we use a richer language that includes timed-initial-literals (PDDL 2.2), durative goals (PDDL 3.0), and multi-valued variables (PDDL 3.1). In addition, we extend the language to allow conditions expressed over sub-intervals of actions, and effects at arbitrary time points during an action. These features turn out to be particularly useful for encoding many problems of interest, and for encoding our translation.[1] We first provide some brief background on PDDL 2.x and then describe our extensions.

[1]To simplify the presentation, we exclude some features of PDDL that are orthogonal to our approach of handling temporal uncertainty, such as *numeric variables* and *domain axioms*.

In PDDL 2.2, a planning problem P is represented by a tuple $P \doteq \langle V, I, T, G, A \rangle$ where:

- V is a set of propositions.
- I is the initial state: a complete set of assignments of value $\mathtt{T}$ or $\mathtt{F}$ to all propositions in V.
- T is a set of timed-initial-literals, which are tuples $\langle [t]\, f := v \rangle$ with $f \in V$ and $t \in \mathbb{R}^+$ is the wall-clock time at which f will be assigned the Boolean value v.
- $G \subseteq V$ is a goal state: a set of propositions that need to be true when the plan finishes executing.
- A is a set of durative actions a, each of the form $a \doteq \langle d_a, C_a, E_a \rangle$ where:
  - $d_a \in \mathbb{R}^+$ is the action duration. Let st_a and et_a be the start and end times of action a then $d_a \doteq et_a - st_a$.
  - C_a is the set of conditions, each $p \in C_a$ is of the form $\langle (st_p, et_p)\ f = v \rangle$ where st_p and et_p indicate the start and end time points of the condition p and are restricted to be equal to st_a or et_a. When $st_p = et_p = st_a$ or $st_p = et_p = et_a$ then p is an instantaneous *at-start* or *at-end* condition holding at the st_p time point. When $st_p = s_a$ and $et_p = e_a$ then p is an *overall* durative condition holding in the open interval (st_p, et_p). $f \in V$ is a proposition with value $v = \mathtt{T}$ or $v = \mathtt{F}$ over the specified time period.
  - E_a is a set of instantaneous effects, each $e \in E_a$ is of the form $\langle [t_e]\, f := v \rangle$ where $t_e \doteq st_a$ or $t_e \doteq et_a$ is the time at which the *at-start* or *at-end* effect e occurs.

We allow disjunctive action conditions p of the form $\langle (st_p, et_p)\ \bigvee_{i=1}^{n} f_i = v_i \rangle$ in which p is satisfied if at least one disjunct is satisfied for every time point in (st_p, et_p).

A plan π of P is a set of tuples $\langle t_a, a \rangle$, in which actions $a \in A$ are associated to wall-clock start times t_a. π is valid if it is executable in I and achieves all goals in G.

We extend the above features of PDDL 2.2 to include the following features from PDDL 3.0 and 3.1:

- Multi-valued variables, introduced in PDDL 3.1, allow variables f in V to have domains $Dom(f)$ with arbitrary values, instead of just $\mathtt{T}$ and $\mathtt{F}$.
- Durative goals, which can be modeled as constraints in PDDL 3.0, allow each goal $g \in G$ to be associated with an interval $[st_g, et_g]$ specifying when the goal must be true. We also allow the time constant et_π, which indicates that the goal must be reached at the end of the plan.

Beyond PDDL. Additionally, the key features in our framework that go beyond PDDL are: (1) actions can have uncontrollable durations, and (2) action conditions and effects are not restricted to action start or end time points. Specifically:

1. Action duration d_a is replaced by an interval $[d_a^{lb}, d_a^{ub}]$ specifying lower- and upper-bound values on action duration: $d_a^{lb} \leq d_a \leq d_a^{ub}$. We further divide the set of actions A into two subsets:
   - *Controllable* actions A_c, where the duration can be chosen by the planner within the bounds $[d_a^{lb}, d_a^{ub}]$.
   - *Uncontrollable* actions A_u, where the duration is not under the planner's control.
2. Instead of constraining the times st_p and et_p of each condition p or t_e of effect e to be either st_a or et_a, we allow

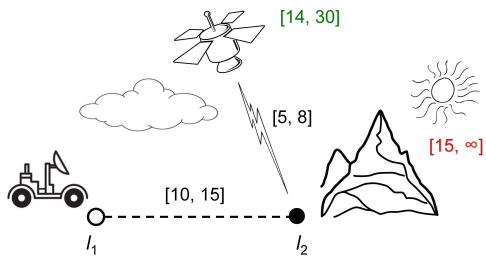


Figure 1: A graphical representation of the running example.

each of them to take an arbitrary value: $st_a + \delta$ or $et_a - \delta$, with $\delta \in \mathbb{R}^+$ (the temporal constraint $st_p \leq et_p$ should still be satisfied). We require δ to be less than or equal to the action duration to prevent effects before the start or after the end of the action[2].

Analogously to PDDL, If $st_p = et_p$ the condition is instantaneous and is required to hold at the single point st_p, otherwise, the condition is required to hold in the open interval (st_p, et_p).

A (strong) plan π^u for a planning problem with uncertainty P^u is valid iff each instance of π^u, obtained by fixing the duration d_a for each uncontrollable action $a \in \pi^u$ to any value within $[d_a^{lb}, d_a^{ub}]$, is a valid plan.

Example. A rover, initially at location l_1, needs to transmit some science data from location l_2 to an orbiter that is only visible in the time window $[14, 30]$. The rover can move from l_1 to l_2 using an action $move$ (abbreviated μ) that has uncontrollable duration between 10 and 15 time units. The data transmission action $transmit$ (abbreviated τ) takes between 5 and 8 time units to complete. The goal of the rover is to transmit the data to the orbiter. Because of the harsh daytime temperatures at location l_2 the rover cannot arrive at l_2 until the sun goes behind the mountains at time 15. Figure 1 illustrates this scenario, which we encode as:

$$V \doteq \{pos : \{l_1, l_2\}, visible : \{\mathtt{T}, \mathtt{F}\}, hot : \{\mathtt{T}, \mathtt{F}\}, sent : \{\mathtt{T}, \mathtt{F}\}\}$$
$$I \doteq \{pos = l_1, visible = \mathtt{F}, sent = \mathtt{F}, hot = \mathtt{T}\}$$
$$T \doteq \{\langle [14]\ visible := \mathtt{T}\rangle, \langle [30]\ visible := \mathtt{F}\rangle, \langle [15]\ hot := \mathtt{F}\rangle\}$$
$$G \doteq \{\langle [et_\pi, et_\pi]\ sent = \mathtt{T}\rangle\}$$
$$A_c \doteq \varnothing$$
$$A_u \doteq \{\langle [10, 15], C_\mu, E_\mu\rangle, \langle [5, 8], C_\tau, E_\tau\rangle\}$$
$$C_\mu \doteq \{\langle (st_\mu, st_\mu)\ pos = l_1\rangle, \langle (et_\mu, et_\mu)\ hot = \mathtt{F}\rangle\}$$
$$C_\tau \doteq \{\langle (st_\tau, et_\tau)\ pos = l_2\rangle, \langle (st_\tau, et_\tau)\ visible = \mathtt{T}\rangle\}$$
$$E_\mu \doteq \{\langle [et_\mu]\ pos := l_2\rangle\}$$
$$E_\tau \doteq \{\langle [et_\tau]\ sent := \mathtt{T}\rangle\}$$

Figure 2 graphically shows a valid plan:

$$\pi^u \doteq \{\langle 6, move(l_1 \rightarrow l_2)\rangle, \langle 22, transmit\rangle\}$$

Note that all the actions in π^u have uncontrollable duration.

[2] In our implementation, which handles the ANML modeling language (see Section 4), we allow even more freedom in expressing st_p, et_p, and t_e such as: (i) $st_p = st_a + 0.3 \times duration(a)$ (i.e., condition starts at 30% into the total action execution time) or (ii) conditions and effects outside of the action duration.

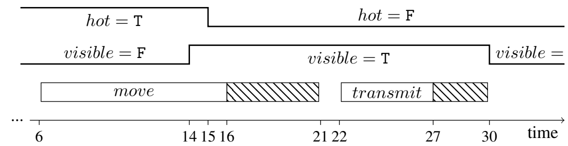


Figure 2: Graphical execution of π^u. Striped regions represent the uncertainty in the action duration.

Discussion. In general, finding a strong plan for a problem with duration uncertainty is more complex than simply considering the maximum or the minimum duration for each action. Consider our rover example and its strong plan shown in Figure 2. The μ (i.e., $move$) action must terminate before the transmit action can start and μ cannot terminate before time 15 due to the temperature constraint. If we only consider the lower-bound on the duration of μ (i.e., planning with $d_\mu^{lb} = 10$) one valid plan is: $\pi_{lb} \doteq \{\langle 11, \mu\rangle, \langle 22, \tau\rangle\}$. However, because of the uncertainty in the actual execution duration of μ, it may actually take 14 time units to arrive at l_2. Thus, the rover would start transmitting at time 22 before it actually arrives at l_2 at time $11 + 14 = 25$. Thus, plan π_{lb} is invalid. Similarly, if we consider only the maximal duration (i.e., planning with $d_\mu^{ub} = 15$), then one possible plan would be: $\pi_{ub} \doteq \{\langle 1, \mu\rangle, \langle 22, \tau\rangle\}$. However, during the actual execution of μ, it may again take only 11 time units (and not the planned maximum 15 time units) to arrive at l_2. This would violate the constraint that it should arrive at l_2 after $t = 15$ to avoid the sun and thus π_{ub} is also not a valid plan. In some special cases it is possible to consider only the maximal duration for an action but this optimization is not sound in general.

In contrast to ordinary temporal planning, it is not possible to compile away *disjunctive conditions* using the action duplication technique [Gazen and Knoblock, 1997], because the set of satisfied disjuncts in the presence of uncertainty can depend on the contingent execution. For example, consider an action a starting at time t, where two Boolean variables p_1 and p_2 are $\mathtt{F}$. a has uncontrollable duration in $[l, u]$, an at-start effect $e_1 \doteq \langle [st_a]\ p_1 := \mathtt{T}\rangle$ and two at-end effects $e_2 \doteq \langle [et_a] p_1 := \mathtt{F}\rangle$ and $e_3 \doteq \langle [et_a] p_2 := \mathtt{T}\rangle$. An at-start condition $p_1 \vee p_2$ of another action b is satisfied anywhere between the start of the action a and the next deletion of p_2. Thus, b can start anytime within $d \doteq (t+l, t+u]$. However, if we compile away this disjunctive condition by replacing b with two actions b_1 and b_2: one with an at-start condition p_1 and the other with an at-start condition p_2, then b_1 is not executable within d because there is no time point in d in which we can guarantee that $p_1 = \mathtt{T}$ (because a may take the minimum duration l and thus the at-end effect e_2 will occur at $t + l$ to set $p_1 = \mathtt{F}$). Similarly, we cannot start b_2 within d because we also cannot guarantee that $p_2 = \mathtt{T}$ at any time point within d (because a may take the maximum duration u and thus e_3 that set $p_2 = \mathtt{T}$ will not happen until $t + u$). Thus, compiling away disjunctive conditions leads to incompleteness when there are uncontrollable durative actions. For this reason it is important to explicitly model disjunctive conditions in our language.

3 Compilation Technique

In this section, we present our compilation technique, which can be used to reduce any planning instance P having duration uncertainty into a temporal planning instance P' in which all actions have controllable durations. The translation guarantees that P is solvable if and only if P' is solvable. Moreover, given any plan for P' we can derive a plan for P. This approach comes at the cost of duplicating some of the variables in the domain, but allows for the use of off-the-shelf temporal planners.

The overall intuition behind the translation is the following. Consider the $transmit$ (τ) action in our example, and suppose it is scheduled to start at time k. Let v be the value of $sent$ at time $k + 5$; since $transmit$ has an at-end effect $\langle [et_\tau]\ sent := \mathtt{T}\rangle$, we know that the value of the variable $sent$ during the interval $(k+5, k+8]$ will be either v or $\mathtt{T}$ depending on the duration of the action. After time $k+8$ we are sure that the effect took place, and we are sure of the value of $sent$ until another effect is applied. Since we are not allowed to observe anything at run-time in strong planning, we need to consider this uncertainty in the value of $sent$ and produce a plan that works regardless. Since $sent$ could appear as a condition of another action (or as a goal condition, as in our example) we must rewrite such conditions to be true only if both $\mathtt{T}$ and v are values that satisfy the condition.

To achieve this, we create an additional variable $sent_\sigma$ (the *shadow variable of* $sent$). This secondary variable stores the alternative value of $sent$ during uncertainty periods. When there is no uncertainty in the value of $sent$, both $sent$ and $sent_\sigma$ will have the same value. In this way, all the conditions involving $sent$ can be rewritten in terms of $sent$ and $sent_\sigma$ to ensure they are satisfied by both the values.

In general, our translation works by rewriting a planning instance $P \doteq \langle V, I, T, G, A\rangle$ into a new planning instance $P' \doteq \langle V', I', T', G', A'\rangle$ that does not contain actions with uncontrollable duration.

Uncertain Variables. The first step is to identify the set of variables $L \subseteq V$ that appear as effects of uncontrollable actions and are executed at a time depending on the end of the action.

$$L \doteq \{f \mid a \in A_u, \langle [t]\ f := v\rangle \in E_a, t \doteq et_a - \delta\}$$

Intuitively, this is the set of variables that can possibly have uncertain value during plan execution. A variable that is modified only at times linked to the start of actions or by timed initial literals, cannot be uncertain as neither the starting time of actions nor the timed initial literals can be uncertain in our model. In our running example, the set L becomes $\{sent, pos\}$.

We now define the set V' as the original variables V plus a shadow variable for each variable appearing in L.

$$V' \doteq V \cup \{f_\sigma \mid f \in L\}$$

We use the pair of variables f and f_σ to represent uncertainty: if $f = f_\sigma$ we know that there is no uncertainty in the value of f, while if $f \neq f_\sigma$ we know that the actual value of f in the original problem is either f or f_σ.

Disjunctive Conditions. At the end of Section 2, we outlined the reason why existing approaches of compiling away disjunctive conditions will not work with uncontrollable action durations. In order to rewrite a disjunctive condition $p \doteq \langle (st_p, et_p)\ \bigvee_{i=1}^n f_i = v_i\rangle$ we need to ensure that the result is satisfied if and only if both the values of f and f_σ for each $f \in L$ satisfy p. For this reason, we define an auxiliary function $\chi(\psi)$ that takes a single disjunctive condition in P and returns a set of disjunctive conditions in P'.

$$\chi(\psi) \doteq \begin{cases} \{\langle f = v\rangle\} & \text{if } \psi \doteq \langle f = v\rangle, f \notin L \\ \{\langle f = v\rangle, \langle f_\sigma = v\rangle\} & \text{if } \psi \doteq \langle f = v\rangle, f \in L \\ \{r \vee s \mid r \in \chi(\psi_1), s \in \chi(\psi_2)\} & \text{if } \psi \doteq \psi_1 \vee \psi_2 \end{cases}$$

For example, the condition of the τ action, $pos = l_2$, is translated as the two conditions $pos = l_2$ and $pos_\sigma = l_2$. Analogously, assuming that both f and g are in L, a given condition $(f = \mathtt{T}) \vee (g = \mathtt{F})$ in P is translated by function χ as the set of conditions $\{(f = \mathtt{T}) \vee (g = \mathtt{F}), (f_\sigma = \mathtt{T}) \vee (g = \mathtt{F}), (f = \mathtt{T}) \vee (g_\sigma = \mathtt{F}), (f_\sigma = \mathtt{T}) \vee (g_\sigma = \mathtt{F})\}$ in P'.

Uncertain Temporal Intervals. We also need to identify the temporal interval in which the value of a given variable can be uncertain. Given an action a with uncertain duration d_a in $[l, u]$, let $\lambda(t)$ and $\nu(t)$ be the earliest and latest possible times at which an at-end effect at $t \doteq et_a - \delta$ may happen. Thus: $\lambda(t) \doteq st_a + l - \delta$ and $\nu(t) \doteq st_a + u - \delta$. Both functions are equal to t if $t \doteq st_a + \delta$. For example, consider the effect $e_1 \doteq \langle [et_\tau]\ sent := \mathtt{T}\rangle$ of action τ. We know that the duration of transmit is uncertain in $[5, 8]$, therefore the effect can be applied between $\lambda(et_\tau) \doteq st_\tau + 5$ and $\nu(et_\tau) \doteq st_\tau + 8$ and the $sent$ variable has an uncertain value within that interval.

Uncontrollable Actions. For each uncontrollable action $a \doteq \langle [l, u], C_a, E_a\rangle\rangle$ in A_u in the original model we create a new action $a' \doteq \langle [u, u], C_{a'}, E_{a'}\rangle$ in A'_c. Specifically, we first fix the maximal duration u as the only allowed duration for a' and then insert appropriate effects and conditions *during* the action to capture the uncertainty.

The effects $E_{a'}$ are partitioned in two sets $E^l_{a'}$ and $E^u_{a'}$ to capture possible values within the uncertain action execution duration. The conditions $C_{a'}$ are also composed of two elements: the rewritten conditions $C^R_{a'}$ and the conditions added to protect the new effects $C^E_{a'}$ (thus $C' \doteq C^R_{a'} \cup C^E_{a'}$).

Rewritten conditions $C^R_{a'}$*:* are obtained by rewriting existing action conditions by means of the χ function. The intervals specifying the duration of the conditions are preserved; since the action duration is set to its maximum, the intervals of the conditions are "stretched" to match their maximal duration.

$$C^R_{a'} \doteq \{\langle (\lambda(t_1), \nu(t_2))\ \alpha\rangle \mid \alpha \in \chi(\psi), \langle (t_1, t_2)\ \psi\rangle \in C_a\}$$

For example, the set C^R_τ for the τ action is: $\{\langle (st_\tau, st_\tau + 8)\ pos = l_2\rangle, \langle (st_\tau, st_\tau + 8)\ pos_\sigma = l_2\rangle, \langle (st_\tau, st_\tau + 8)\ visible = \mathtt{T}\rangle\}$. This requires variables $visible$, pos and pos_σ to be true throughout the execution of τ.

Compiling action effects: The effects of the original action are duplicated: both the affected variable f and its shadow f_σ are modified, but at different times. We first identify the earliest and latest possible times at which an effect can happen due to the duration uncertainty (see earlier discussion on $\lambda(t)$ and $\nu(t)$). We then apply the effect on f_σ at the earliest time point $\lambda(t)$, and at the latest time point $\nu(t)$ we re-align f and f_σ by

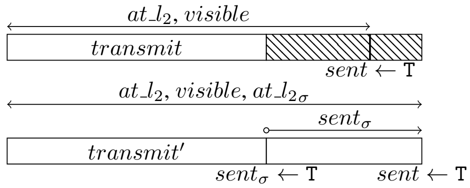


Figure 3: Graphical view of the original $transmit$ action instance (top) and its compilation (bottom).

also applying the effect on the original variable f:

$$E^l_{a'} \doteq \{\langle[\lambda(t)]\ f_\sigma := v\rangle \mid \langle[t]\ f := v\rangle \in E_a\}$$

$$E^u_{a'} \doteq \{\langle[\nu(t)]\ f := v\rangle \mid \langle[t]\ f := v\rangle \in E_a\}$$

For example, the τ action has $E^l_\tau \doteq \{\langle[st_\tau + 5]\ sent_\sigma := \mathtt{T}\rangle\}$ and $E^u_\tau \doteq \{\langle[st_\tau + 8]\ sent := \mathtt{T}\rangle\}$.

Additional conditions $C^E_{a'}$: let $t \doteq et_a - \delta$ be the time of an at-end effect that affects the value of f. In order to prevent other actions from changing the value of f during the interval $(\lambda(t), \nu(t)]$ where the value of f is uncertain, we add a condition in $C^E_{a'}$ to maintain the value of f_σ throughout the uncertain duration $(\lambda(t), \nu(t)]$.

$$C^E_{a'} \doteq \{\langle(\lambda(t), \nu(t))\ f_\sigma = v\rangle, \mid \langle[t]\ f := v\rangle \in E_a\} \cup \{\langle(\nu(t), \nu(t))\ f_\sigma = v\rangle \mid \langle[t]\ f := v\rangle \in E_a\}$$

We are in fact using a left-open interval $(\lambda(t), \nu(t)]$ by specifying the same condition on the open interval $(\lambda(t), \nu(t))$ and the single point $[\nu(t)]$. Since the effect on f_σ (belonging to $E^l_{a'}$) is applied at time $\lambda(t)$, the condition is satisfied immediately after the effect and we want to avoid concurrent modifications of either f or f_σ until the uncertainty interval ends at $\nu(t)$. For example, the τ action has $C^E_{\tau'} \doteq \{\langle(st_\tau + 5, st_\tau + 8)\ sent_\sigma = \mathtt{T}\rangle\}$. The full compilation of the τ action is depicted in Figure 3.

Controllable actions: are much simpler. For each $a \doteq \langle[l, u], C_a, E_a\rangle \in A_c$ we introduce a replacements action $a' \doteq \langle[l, u], C_{a'}, E_{a'}\rangle \in A'_c$, in which: (1) each condition in C is rewritten to check the values of both the variables and their shadows, and (2) each effect is applied to a variable and its shadow, if any.

$$C_{a'} \doteq \{\langle(\lambda(t_1), \nu(t_2))\ \alpha\rangle \mid \alpha \in \chi(\psi), \langle(t_1, t_2)\ \psi\rangle \in C_a\}$$

$$E_{a'} \doteq E_a \cup \{\langle[t]\ f_\sigma := v\rangle \mid f \in L, \langle[t]\ f := v\rangle \in E_a\}$$

Initial state I: is handled by initializing variables and their corresponding shadow variables in the same way as in the original problem.

$$I' \doteq I \cup \{f_\sigma = v \mid f \in L, f = v \in I\}$$

For example, the initial state of our running problem is the original initial state plus $\{sent_\sigma = \mathtt{F}, pos_\sigma = l_1\}$.

Timed Initial Literals: T' are set similarly to the effects.

$$T' \doteq T \cup \{\langle[t]\ f_\sigma := v\rangle \mid f \in L, \langle[t]\ f := v\rangle \in T\}$$

In our example, we do not have timed initial literals operating on uncertain variables, thus $T' \doteq T$.

Goal conditions: G is augmented to consider both the original variables and the shadow variables, without modifying the application times as they are fixed and cannot be uncertain.

$$G' \doteq G \cup \{\langle[t_1, t_2]\ f_\sigma = v\rangle \mid f \in L, \langle[t_1, t_2]\ f = v\rangle \in G\}$$

In our example, the set G' becomes $\{\langle[et_\pi, et_\pi]\ sent = \mathtt{T}\rangle, \langle[et_\pi, et_\pi]\ sent_\sigma = \mathtt{T}\rangle\}$.

Discussion: This compilation is sound and complete, in the sense that the original problem is solvable if and only if the resulting problem is solvable[3]. Given any plan for the rewritten temporal planning problem, it is automatically a strong plan for the original problem (with the obvious mapping from the rewritten to the original actions).

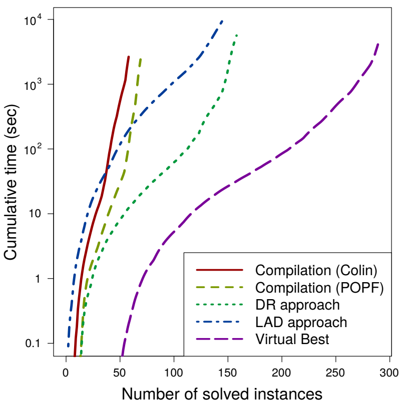
The compilation produces a problem that: (i) has at most twice the number of variables of the original problem, (ii) at most twice the initial and timed assignments and (iii) exactly the same number of actions. The only point in which the compilation might produce exponentially large formulae is in the application of the χ function, which is exponential in the number of disjuncts constraining variables appearing in L.

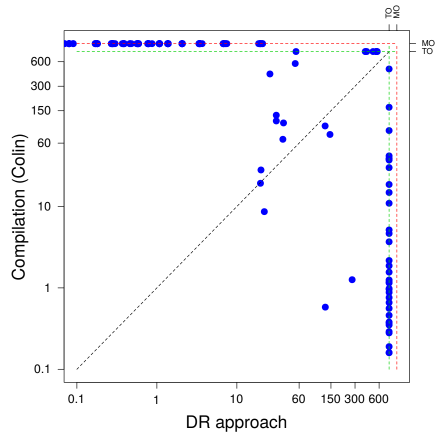
4 Implementation and Experiments

We conducted two sets of experiments. In the first, we compare our approach against the techniques proposed in [Cimatti *et al.*, 2015]. This is the only domain-independent planner that we are aware of that can find strong plans for PDDL 2.1 planning problems with uncontrollable durations. For this experiment, we use an extension to PDDL 2.1 that includes actions with uncontrollable durations (but none of the other extensions that we described in Section 2 such as preconditions and effects at arbitrary times, multi-valued variables, timed-initial-literals, or disjunctive preconditions). In the second, we show the applicability of our technique on a very expressive fragment of the ANML [Smith *et al.*, 2008] language extended with uncertainty in action durations. Except for action duration uncertainty, ANML natively supports all the features described in Section 2.

PDDL with duration uncertainty. Cimatti *et al.* (2015) extended the COLIN planner [Coles *et al.*, 2012] to solve SPPTUs. They address the problem by substituting the STN scheduler with a solver for strong controllability of STNUs. This simple replacement yields a solver that is sound but incomplete for SPPTU because of the ordering constraints that are checked by the scheduler. Following this idea, Cimatti *et al.* propose two techniques to overcome the incompleteness based on the reordering of actions in the plan. "Last Achiever Deordering" (LAD) is a sound-but-incomplete technique that tries to limit the incompleteness by using a least-commitment encoding of the STNU by considering, for each condition in the plan, the last achiever of the condition, thus freeing the plan from being a total order. "Disjunctive Reordering" (DR) is a sound-and-complete technique obtained by considering, at each step, all the possible valid action reorderings using a disjunctive form of STNU.

[3]An extended version of this paper including the proof is available at `http://es.fbk.eu/people/amicheli/resources/ijcai2015`.





ANML Instance	Controllable	SPPTU
match 1	0.517	0.626
match 2	0.522	0.637
match 3	0.593	1.115
rovers 1	1.196	1.293
rovers 2	1.497	1.810
rovers 3	1.190	2.009
handover 1	0.800	1.081
handover 2	2.302	4.043
handover 3	2.863	32.484

Figure 4: Experimental results. Cumulative time plot (left) of the solving time for the PDDL benchmarks. Scatter plot (center) of the running time in seconds for the compilation approach (solved using the COLIN planner) against the DR approach. Results for the ANML benchmarks (right table).

We compare against this approach by first compiling away temporal uncertainties and then using both the COLIN and POPF planners to solve the compiled instances[4]. We compared our *sound and complete* technique against both the *complete* DR and the *incomplete* LAD approaches presented by Cimatti *et al.*. We used a timeout of 600 seconds, with 8 GB of memory and the full benchmark set of 563 problems described in [Cimatti *et al.*, 2015].

The left plot of Figure 4 reports the cumulative time of the three techniques and the "Virtual Best" solver, obtained by picking the best solving technique for each instance. The central scatter-plot compares our technique (instantiated with COLIN) with the DR approach. The left plot shows that the compilation technique cannot solve as many instances as DR or LAD. However, we note that the "Virtual Best" solver solves many more problems than both DR and LAD. This shows that the techniques are complementary: problem instances that cannot be solved by LAD or DR are solved quickly by our compilation, and vice-versa. This situation is also visible in the scatter plot: there is a clear subdivision of the problem instances solved by these two different planners.

Our investigation indicates that the main factor that hinders the performance of our approach is the "clip-action" construction [Fox *et al.*, 2004] needed to reduce our compilation to PDDL 2.1. Our compilation generates actions with conditions and effects that occur at intermediate times. Compiling this to PDDL 2.1 requires three PDDL 2.1 actions for each action in A_u: a container action, and two actions inside the container action that are clipped together. This deepens the search and lengthens the plans for COLIN and POPF.

ANML with duration uncertainty. As described in Section 2 and 3, our framework handles many useful features beyond PDDL 2.1. Some of these can be represented in higher levels of PDDL (e.g., multi-valued variables), some cannot (e.g., arbitrary timed action conditions and effects). While comparing against current state-of-the-art in PDDL2.1 shows the feasibility of our approach, it restricts us to a small subset of features that can be handled by our compilation. Moreover, as discussed above, the limitations of PDDL 2.1 adversely impacts the performance of our approach.

To show the full expressive potential of our approach, we used the Action Notation Modeling Language (ANML) [Smith *et al.*, 2008], which can natively model all those constraints. ANML is a variable-value language that allows high-level modeling, complex conditions and effects, HTN decomposition and timeline-like temporal constraints. Our only addition to ANML is the capability to model uncertain action durations: `duration :in [l, u]` where l and u are constant values specifying the lower and upper bounds on the duration of a. We name our ANML extension: ANuML.

We implemented our compilation approach in an automatic translator that accepts an ANuML planning instance and produces plain ANML. We then use the FAPE [Dvorak *et al.*, 2014] planner to produce a plan for the compiled ANML problem instance. To the best of our knowledge no other approach is able to solve the problems we are dealing with in ANML. We considered two domains adapted from the FAPE distribution, namely "rover" and "handover". The former models a remote rover for scientific operations, similar to our running example, while the latter models a situation in which two robots must synchronize to exchange items. Additionally, we model a "match" domain derived from the "matchcellar" domain used in IPC 2011. For each domain, we tested with three different configurations: different initial states, goals, and variable domains.

The right table in Figure 4 compares the time needed for FAPE to produce a plan ignoring the temporal uncertainty (i.e. considering the environment to be completely cooperative) with the time needed to solve the compiled problem.

Although the performance of the encoding depends on the

[4] Our approach allows the use of any PDDL2.1 planner that can handle required concurrency. Unfortunately, many temporal planners such as LPG and TemporalFastDownward do not support this, and therefore cannot find solutions to the problems generated by our compilation.

planning instance, the results show that the slowdown is acceptable for the tested instances. An exception is "handover 3", in which the translation shows a significant slowdown. We remark that this is not a comparison between two equivalent techniques, as the two columns correspond to results in solving very different problems: plain temporal planning vs. strong planning with temporal uncertainty. Instead, this is an indication of the slowdown introduced by the translation compared to the same problem without uncertainty. Even though the results are preliminary, we can infer that our approach is more than a theoretical exercise and can be practically applied for expressive temporal planning domains modeled natively in ANML.

5 Future Work

While the preliminary results are promising, we are considering several possible extensions.
Model simplification: it is sometimes possible to simplify a strong planning problem with temporal uncertainty by considering the maximal or minimal duration of an action having uncertain duration. As we discussed in Section 2, this "worst-case" approach is in general unsound; nonetheless, it is possible to recognize some special cases in which it is sound and complete. This simplification can be done upfront and could be beneficial for both our compilation and the approaches in [Cimatti *et al.*, 2015].
Increase expressiveness: Even though the formalization we presented is quite expressive and general, the ANML language has many features that are not covered. A prominent example is the support of conditional effects, which cannot be expressed in our language but are possible in both ANML and PDDL. We note that, analogously to disjunctive preconditions, the common compilation of conditional effects is unsound in the presence of temporal uncertainty, because it transforms a possibly uncontrollable effect into a controllable decision for the planner.
Improve performance: Finally, we would like to study ways to overcome the disappointing performance of the compilation into PDDL by hybridizing the "native" DR and LAD techniques with our approach to exploit their complementarity. Another possibility is to modify a temporal planner so that it understands the clip-action construct and avoids useless search when dealing with our translations.

Acknowledgments

We thank Jeremy Frank, Alessandro Cimatti and Paul Morris for suggestions, fruitful discussion, and feedback on an early version of this paper. This work was supported by the NASA Automation for Operations (A4O) project.

References

[Beaudry *et al.*, 2010] E. Beaudry, F. Kabanza, and F. Michaud. Planning for concurrent action executions under action duration uncertainty using dynamically generated bayesian networks. In *ICAPS*, 2010.

[Bresina *et al.*, 2002] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *UAI*, pages 77–84, 2002.

[Cesta *et al.*, 2009] A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, and R. Rasconi. The APSI Framework: a Planning and Scheduling Software Development Environment. In *ICAPS Application Showcase*, 2009.

[Cimatti *et al.*, 2013] A. Cimatti, A. Micheli, and M. Roveri. Timelines with temporal uncertainty. In *AAAI*, pages 195–201, 2013.

[Cimatti *et al.*, 2014] A. Cimatti, A. Micheli, and M. Roveri. Solving strong controllability of temporal problems with uncertainty using SMT. *Constraints*, 2014.

[Cimatti *et al.*, 2015] A. Cimatti, A. Micheli, and M. Roveri. Strong temporal planning with uncontrollable durations: a state-space approach. In *AAAI*, 2015.

[Coles *et al.*, 2012] A. Coles, A. Coles, M. Fox, and D. Long. COLIN: Planning with continuous linear numeric change. *JAIR*, 44:1–96, 2012.

[Dvorak *et al.*, 2014] F Dvorak, A. Bit-Monnot, F. Ingrand, and M. Ghallab. A flexible ANML actor and planner in robotics. In *ICAPS Planning and Robotics Workshop*, 2014.

[Fox *et al.*, 2004] M. Fox, D. Long, and K. Halsey. An investigation into the expressive power of PDDL2.1. In *ECAI*, pages 328–342, 2004.

[Frank and Jónsson, 2003] J. Frank and A. Jónsson. Constraint-based Attribute and Interval Planning. *Constraints*, 8(4):339–364, 2003.

[Gazen and Knoblock, 1997] B. Gazen and C. Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In *ECP*, New York, 1997. Springer-Verlag.

[Ghallab and Laruelle, 1994] M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In *AIPS*, pages 61–67, 1994.

[Ghallab *et al.*, 2004] M. Ghallab, D. Nau, and P. Traverso. *Automated planning - Theory and Practice*. Morgan Kaufmann, 2004.

[Mausam and Weld, 2008] Mausam and D. Weld. Planning with durative actions in stochastic domains. *JAIR*, 31:33–82, 2008.

[Morris, 2006] P. Morris. A structural characterization of temporal dynamic controllability. In *CP*, pages 375–389, 2006.

[Muise *et al.*, 2013] C. Muise, C. Beck, and S. McIlraith. Flexible execution of partial order plans with temporal constraints. In *IJCAI*, 2013.

[Palacios and Geffner, 2009] H. Palacios and H. Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *JAIR*, 35:623–675, 2009.

[Peintner *et al.*, 2007] B. Peintner, K. Venable, and N. Yorke-Smith. Strong controllability of disjunctive temporal problems with uncertainty. In *CP*, pages 856–863, 2007.

[Santana and Williams, 2012] P. Santana and B. Williams. A bucket elimination approach for determining strong controllability of temporal plans with uncontrollable choices. In *AAAI*, 2012.

[Smith *et al.*, 2008] D. Smith, J. Frank, and W. Cushing. The ANML language. In *ICAPS Poster session*, 2008.

[Vidal and Fargier, 1999] T. Vidal and H. Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.*, 11(1):23–45, 1999.

[Younes and Simmons, 2004] H. Younes and R. Simmons. Solving generalized Semi-Markov Decision Processes using continuous phase-type distributions. In *AAAI*, 2004.