A Logical Semantics for PDDL+

Vitaliy Batusov
York University, Toronto, Canada
vbatusov@cse.yorku.ca

Mikhail Soutchanski[*]
Ryerson University, Toronto, Canada
mes@scs.ryerson.ca

Abstract

PDDL+ is an extension of PDDL2.1 which incorporates fully-featured autonomous processes and allows for better modelling of mixed discrete-continuous domains. Unlike PDDL2.1, PDDL+ lacks a logical semantics, relying instead on state-transitional semantics enriched with hybrid automata semantics for the continuous states. This complex semantics makes analysis and comparisons to other action formalisms difficult. In this paper, we propose a natural extension of Reiter's situation calculus theories inspired by hybrid automata. The kinship between PDDL+ and hybrid automata allows us to develop a direct mapping between PDDL+ and situation calculus, thereby supplying PDDL+ with a logical semantics and the situation calculus with a modern way of representing autonomous processes. We outline the potential benefits of the mapping by suggesting a new approach to effective planning in PDDL+.

Introduction

The Planning Domain Definition Language (PDDL) is the dominant AI formalism for expressing planning instances. Since its introduction by (McDermott et al. 1998), PDDL has undergone several revisions and extensions in response to the insights gained from the practice of automated planning at the International Planning Competitions and elsewhere in academia and the industry. PDDL+ is a prominent offshoot of PDDL2.1 (Fox and Long 2003) which was introduced in (Fox and Long 2006) in order to significantly improve the ability to model discrete-continuous domains.

In a nutshell, PDDL+ introduces a more powerful model of continuous temporal change, namely the *start-process-stop* model, which supersedes the previous *durative action* model. This is motivated by an abundance of examples of domains wherein the durative actions are inappropriate or insufficient due to the complex interactions of discrete occurrences and continuous processes (Boddy 2003; McDermott 2003). Fox and Long cite the hybrid automata (HA) and the knowledge representation (KR) communities as influences for the formulation of PDDL+. The key feature borrowed by PDDL+ from the hybrid automata approach is

its explicit recognition of two distinct modes of change — the logical state transitions and the continuous change over time within a logical state. Thus, in PDDL+, processes can be initiated or terminated by any discrete change, but have continuous and possibly concurrent effects on numerical fluents. In KR, the start-process-stop model first appeared in the work of (Pinto 1994; Ternovskaia 1994) on situation calculus (SC), but has not been developed far enough to formulate continuous time-dependent change explicitly and achieve the accompanying representational benefits.

The formal semantics of PDDL originated with the STRIPS semantics of (Lifschitz 1987) and reached maturity in (Fox and Long 2003) for PDDL2.1. As pointed out in (Claßen, Hu, and Lakemeyer 2007), this meta-theoretic semantics achieves its stated goals, but is difficult to grasp and is rather unwieldy for the purposes of analysis and comparison to other action formalisms. Claßen, Hu, and Lakemeyer remedy this by providing an alternative, purely logical semantics for PDDL2.1. Their semantics is based on a subset of situation calculus called $\mathcal{ES}$, due to (Lakemeyer and Levesque 2004), and handles the durative action model in a robust and concise manner. In PDDL+, the semantics of continuous change is defined in terms of hybrid automata, owing to the kinship in the modelling approach and aiming for cross-pollination between the fields of automated planning and control theory. However, while this HA semantics is more amenable to analysis, it remains wrapped in the complex meta-theoretic semantics of PDDL2.1.

In this paper, we present a natural extension of SC basic action theories (Reiter 2001) which aligns closely with the HA and PDDL+ approach to representing continuous change. Namely, in addition to the usual atemporal situation-dependent fluents, we introduce new temporal fluents such as $velocity(b, t, s)$ which explicitly depend on real-valued time t. Due to the shared origins, this enables our "hybrid" situation calculus to provide a situation calculus semantics for PDDL+. The benefit is mutual in that PDDL+, like PDDL2.1, gets a robust logical semantics and the situation calculus gets a modern HA-like means of representing discrete-continuous change, completing the loop of cross-pollination between all three fields. We discuss how the "hybrid" SC can contribute to previous work on planning in SC to perform automated planning in PDDL+ with only minimal grounding of the action and process schemas.

Hybrid Situation Calculus

Situation Calculus

Situation calculus (SC) is a second-order language for representing dynamic worlds. It has basic sorts (SITUATION, ACTION, OBJECT) and a rich vocabulary for constructing formulas over terms of these sorts. Reiter (2001) shows that to solve many reasoning problems about actions, it is convenient to work with *basic action theories* (BATs). (Lakemeyer and Levesque 2004) develop a variant of SC called $\mathcal{ES}$ (and an $\mathcal{ES}$-variant of BATs) which is used by (Claßen, Hu, and Lakemeyer 2007) to supply a SC semantics for PDDL2.1. While $\mathcal{ES}$ offers syntactic and proof-theoretic benefits, it gives up the ability to express interesting statements about situations such as "there exists a situation such that every situation preceding it satisfies P" — i.e., exactly the kind of statements about dynamic systems which control theory is interested in proving. For this reason, the present paper maps PDDL+ instances to Reiter's BATs and not to $\mathcal{ES}$.

The main ingredients of a BAT are precondition axioms and successor state axioms. For each action function $A(\bar{x})$, an *action precondition axiom* (APA) has the syntactic form

$$\forall \bar{x}, s. \ Poss(A(\bar{x}), s) \leftrightarrow \Pi_A(\bar{x}, s),$$

meaning that the action $A(\bar{x})$ is possible in situation s if and only if $\Pi_A(\bar{x}, s)$ holds in s, where $\Pi_A(\bar{x}, s)$ is a formula with free variables among $\bar{x} = (x_1, \ldots, x_n)$ and s. Situations are first order (FO) terms which denote sequences of actions. A distinguished constant S_0 is used to denote the *initial situation*, and function $do(\alpha, \sigma)$ denotes the situation that results from performing action α in situation σ. Every situation corresponds uniquely to a sequence of actions. We use $do([\alpha_1, \ldots, \alpha_n], S_0)$ to denote complex situation terms obtained by consecutively performing $\alpha_1, \ldots, \alpha_n$ in S_0. The notation $\sigma' \sqsubseteq \sigma$ means that either situation σ' is a subsequence of situation σ or $\sigma = \sigma'$. The formula $\forall a \forall s'(do(a, s') \sqsubseteq \sigma \rightarrow Poss(a, s'))$, abbreviated as $executable(\sigma)$, captures situation σ all of whose actions are consecutively possible. Every BAT contains a set Σ of foundational axioms which characterize situations as a single finitely branching infinite tree starting from S_0. Objects are FO terms other than actions and situations that depend on the domain of application. Above, $\Pi_A(\bar{x}, s)$ is a formula *uniform* in situation argument s: it does not mention the predicates $Poss$, $\sqsubseteq$, it does not quantify over variables of sort situation, it does not mention equality on situations, and it has no occurrences of situation terms other than s. For each relational fluent $F(\bar{x}, s)$ and each functional fluent $f(\bar{x}, s)$, respectively, a *successor state axiom* (SSA) has the form

$$F(\bar{x}, do(a, s)) \leftrightarrow \Phi_F(\bar{x}, a, s),$$
$$f(\bar{x}, do(a, s)) = y \leftrightarrow \phi_f(\bar{x}, y, a, s),$$

where $\Phi_F(\bar{x}, a, s)$ and $\phi_f(\bar{x}, y, a, s)$ are formulas uniform in s, all of whose free variables are among those explicitly shown. (As usual, all free variables are $\forall$-quantified at front.) In addition, a BAT $\mathcal{D}$ contains an *initial theory*: a finite set $\mathcal{D}_{S_0}$ of FO formulas whose only situation term is S_0 (and possibly formulas without a situational argument). Finally,

BATs include a set $\mathcal{D}_{una}$ of unique name axioms for actions (UNA) specifying that two actions are different if their names are different and that identical actions have identical arguments. If a BAT has functional fluents, it is required to satisfy a consistency property whereby, for the right-hand side $\phi_f(\bar{x}, y, a, s)$ of the SSA of each functional fluent f, there must exist a unique y such that $\phi_f(\bar{x}, y, a, s)$ is entailed by $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$.

BATs enjoy the *relative satisfiability* property (Pirri and Reiter 1999): a BAT $\mathcal{D}$ is satisfiable whenever $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is. This property allows one to disregard the more problematic parts of a BAT, like the second-order induction axiom, when checking satisfiability. Moreover, BATs benefit from *regression*-based reasoning, a powerful and efficient reasoning mechanism for answering queries about the future (a problem known as *projection*).

To accommodate time, Reiter adds a temporal argument to all action functions and introduces two special function symbols. The symbol $time : \text{ACTION} \mapsto \text{TIME}$ is used to access the time of occurrence of an action via its term and is specified by an axiom $time(A(\bar{x}, t)) = t$ (included in $\mathcal{D}_{S_0}$) for every action function $A(\bar{x}, t)$ in the vocabulary of the BAT. The symbol $start : \text{SITUATION} \mapsto \text{TIME}$ is used to access the starting time of situation s and is specified by the new foundational axiom $start(do(a, s)) = time(a)$. The starting time of S_0 is not enforced, and the time points constituting the timeline with dense linear order are assumed to always have the standard interpretation (along with $+$, $<$, etc.). To outlaw temporal paradoxes, the abbreviation $executable(s)$ is redefined as

$$\forall a, s'. do(a, s') \sqsubseteq s \rightarrow (Poss(a, s') \wedge start(s') \leq time(a)).$$

To model exogenous ("coercive") events, Reiter develops an elegant theory of *natural actions* — non-agent actions that occur spontaneously as soon as their precondition is satisfied. Such actions are marked using the predicate symbol $natural$ as a part of $\mathcal{D}_{S_0}$, e.g., $natural(bounce(Ball_1, t))$, and their semantics are encoded by a further modification of $executable(s)$ (see below). We use natural actions to induce relational change based on the values of the continuous quantities.

Continuous Processes in Situation Calculus

The crux of our proposal to extend BATs is to add a new kind of axioms called *state evolution axioms* (SEA) to Reiter's successor state axioms. The successor state axioms specify, as usual, how fluents change when actions are executed, i.e., they characterize the transitions between logical states due to actions. The state evolution axioms specify how the flow of time can bring changes in system parameters within a given situation while no actions are executed. Thus, we maintain the fundamental assumption of SC that all *discrete* change is due to actions, though situations now include a temporal evolution.

Our starting point is the *temporal change axiom* (TCA) which describes a single law governing the evolution of a particular temporal fluent due to the passage of time in a particular context of an arbitrary situation. Each temporal fluent has time t as an argument, e.g. $velocity(b, t, s)$ — the

velocity of the ball b at the moment of time t in situation s. We assume that a TCA for a temporal functional fluent f has the general syntactic form

$$\gamma(\bar{x}, s) \wedge \delta(\bar{x}, y, t, s) \rightarrow f(\bar{x}, t, s) = y, \quad (1)$$

where t, s, $\bar{x}$, y are variables and $\gamma(\bar{x}, s)$, $\delta(\bar{x}, y, t, s)$ are formulas uniform in s whose free variables are among those explicitly shown. We call $\gamma(\bar{x}, s)$ the *context* as it specifies the condition under which the formula $\delta(\bar{x}, y, t, s)$ is to be used to compute the value of fluent f at time t. Note that contexts are time-independent. The formula $\delta(\bar{x}, y, t, s)$ encodes a function which specifies y in terms of the values of other fluents at s, t. Aside from the consistency conditions below, we leave the general form of δ without constraint.

For each TCA (1) to be *well-defined*, we require that the background theory entails

$$\gamma(\bar{x}, s) \rightarrow \exists y \, \delta(\bar{x}, y, t, s). \quad (2)$$

In other words, whatever the circumstance, the TCA must supply a value for the quantity modelled by f if its context is satisfied.

A set of k well-defined temporal change axioms for some fluent f can be equivalently expressed as an axiom

$$\Phi(\bar{x}, y, t, s) \rightarrow f(\bar{x}, t, s) = y, \quad (3)$$

where $\Phi(\bar{x}, y, t, s)$ is $\bigvee_{1 \leq i \leq k}(\gamma_i(\bar{x}, s) \wedge \delta_i(\bar{x}, y, t, s))$. We additionally require that the background theory entails

$$\Phi(\bar{x}, y, t, s) \wedge \Phi(\bar{x}, y', t, s) \rightarrow y = y'. \quad (4)$$

Condition (4) guarantees the consistency of the axiom (3) by preventing a continuous quantity from having more than one value at any moment of time. With condition (4), we can assume w.l.o.g. that all contexts in the given set of TCA are pairwise mutually exclusive wrt the background theory $\mathcal{D}$.

Having combined all laws which govern the evolution of f with time into a single axiom (3), we can make a causal completeness assumption: *there are no other conditions under which the value of f can change in s from its initial value at $start(s)$ as a function of t*. We capture this assumption formally by the explanation closure axiom

$$f(\bar{x}, t, s) \neq f(\bar{x}, start(s), s) \rightarrow \exists y \, \Phi(\bar{x}, y, t, s). \quad (5)$$

Theorem 1. *The conjunction of axioms (3) and (5) in the models of (4) is logically equivalent to*

$$f(\bar{x}, t, s) = y \leftrightarrow [\Phi(\bar{x}, y, t, s) \vee$$
$$y = f(\bar{x}, start(s), s) \wedge \neg\Psi(\bar{x}, y, t, s)], \quad (6)$$

where $\Psi(\bar{x}, s)$ denotes $\bigvee_{1 \leq i \leq k} \gamma_i(\bar{x}, s)$.

We call the formula (6) a *state evolution axiom* (SEA) for the fluent f. Note what the SEA says: f evolves with time during s according to some law whose context is realized in s or stays constant if no context holds in s. Our causal completeness assumption (5) applies both to physical quantities and to their derivatives. This assumption simply states that all reasons for change have been already accounted for in (3). The proof of this theorem is available in (Batusov, De Giacomo, and Soutchanski 2018). It is important to realize that $\mathcal{D}_{se}$, a set of SEAs, complements the SSAs that are derived in (Reiter 1991) using a similar technique.

"Hybrid" Basic Action Theories

The SEA for some temporal fluent f does not completely specify the behaviour of f because it talks only about change within a single situation s. To complete the picture, we need a SSA describing how the value of f changes (or does not change) when an action is performed. A straightforward way to accomplish this would be by an axiom which would enforce continuity of value. However, this choice would preclude the ability to model action-induced discontinuous jumps in the value of the continuously varying quantities or their derivatives, such as the sudden change of acceleration from 0 to $-9.8m/s^2$ when an object is dropped. To circumvent this limitation, for each temporal functional fluent $f(\bar{x}, t, s)$, we introduce an auxiliary atemporal functional fluent $f_{init}(\bar{x}, s)$ whose value in s represents the value of the physical quantity modelled by f in s at the time instant $start(s)$. We axiomatize f_{init} using a SSA derived from the axioms

$$e(\bar{x}, y, a, s) \rightarrow f_{init}(\bar{x}, do(a, s)) = y,$$
$$\neg\exists y(e(\bar{x}, y, a, s)) \rightarrow f_{init}(\bar{x}, do(a, s)) = f(\bar{x}, time(a), s),$$

where the former is a Reiter's effect axiom in normal form and the latter asserts that if no relevant effect is invoked by the action a, f_{init} assumes the most recent value of the continuously evolving fluent f. This latter axiom enforces temporal continuity in the value of f in the case when there is no reason for change.

The SSA for f_{init} has the general syntactic form

$$f_{init}(\bar{x}, do(a, s)) = y \leftrightarrow \Omega(\bar{x}, y, a, s), \quad (7)$$

where $\Omega(\bar{x}, y, a, s)$ is a formula uniform in s whose purpose is to describe how the initial value of f in $do(a, s)$ relates to its value at the same time instant in s (i.e., prior to a). To establish a consistent relationship between temporal fluents and their atemporal *init*-counterparts, we require that, in an arbitrary situation, the continuous evolution of each temporal fluent f starts with the value computed for f_{init} by its successor state axiom.

A *hybrid basic action theory* is a collection of axioms $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{se}$ such that

1. Every action symbol mentioned in $\mathcal{D}$ is temporal;

2. $\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ constitutes a BAT as per Definition 4.4.5 in (Reiter 2001);

3. $\mathcal{D}_{se}$ is a set of state evolution axioms of the form

$$f(\bar{x}, t, s) = y \leftrightarrow \psi_f(\bar{x}, t, y, s) \quad (8)$$

where $\psi_f(\bar{x}, t, y, s)$ is uniform in s, such that for every temporal functional fluent f in $\mathcal{D}_{se}$, $\mathcal{D}_{ss}$ contains an additional SSA of the form (7) for f_{init};

4. For each SEA of the form (8), the following consistency properties must be entailed by $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$:

$$\forall\bar{x}\forall t. \, \exists y(\psi_f(\bar{x}, t, y, s)) \wedge$$
$$\forall y\forall y'(\psi_f(\bar{x}, t, y, s) \wedge \psi_f(\bar{x}, t, y', s) \rightarrow y = y'), \quad (9)$$

$$\exists y(f_{init}(\bar{x}, s) = y \wedge \psi_f(\bar{x}, start(s), y, s)); \quad (10)$$

It can be shown that "hybrid" BATs retain a variant of the relative satisfiability property, and a variant of the regression operator can also be defined.

Differential Equations in Situation Calculus

Both science and engineering describe dynamic systems in terms of differential equations. Hybrid automata are a prime example of a formalism which relies on ordinary differential equations (ODE) to specify continuous behaviour. PDDL+ specifies continuous effects using an explicit ODE of the form $\frac{d}{dt}(f(t)) = G(t, f)$, where $G(t, f)$ is a continuously differentiable, or more generally, a Lipschitz continuous function over functional fluents and constants. An initial value $f(t_0)$ is implicitly given, and together with the ODE it defines the *initial value problem* that thanks to assumptions about G has a unique solution, e.g., see (Teschl 2012). We encode PDDL+ process effects using the equivalent integral form $f(t) = \int_{t_0}^{t} G(\tau, f)\, d\tau + f(t_0)$.

Henceforth, we require that all FOL structures interpret the definite integral symbol in the standard way. Let $\mathcal{M}$ be an arbitrary situation calculus structure, μ be an object assignment, $h(\bar{x}, t, s)$ a SC term of sort $\mathbb{R}$ whose free variables are among $\bar{x}, t, s$ (of sorts OBJECT, $\mathbb{R}$, SITUATION, resp.), and let τ_1, τ_2 be terms of sort $\mathbb{R}$. Then we require that

$$\int_{\tau_1}^{\tau_2} h(\bar{x}, t, s)\, dt^{\mathcal{M}}[\mu] = \int_{\tau_1^{\mathcal{M}}[\mu]}^{\tau_2^{\mathcal{M}}[\mu]} h^{\mathcal{M}}(\bar{x}^{\mathcal{M}}[\mu], t, s^{\mathcal{M}}[\mu])\, dt.$$

The modeller must ensure that $h^{\mathcal{M}}(\bar{x}^{\mathcal{M}}[\mu], t, s^{\mathcal{M}}[\mu])$ is a continuous real-valued function defined on the interval $[\tau_1^{\mathcal{M}}[\mu], \tau_2^{\mathcal{M}}[\mu]]$.

A mapping from PDDL+ to SC

Our mapping explicitly captures all PDDL+ features subsumed by the requirements `:adl + :fluents`.

A *PDDL+ planning instance* is a pair $I = (Dom, Prob)$ where $Dom = (\mathcal{T}, \mathcal{C}, \mathcal{F}, \mathcal{R}, \mathcal{A}, \mathcal{E}, \mathcal{P})$ describes the domain dynamics and $Prob = (\mathcal{O}, \mathcal{I}, \mathcal{G})$ describes problem-specific objects, the initial state, and the goal state. A planning instance is formulated according to a formal grammar found in (Fox and Long 2003; 2006). In short, $\mathcal{F}$ is a set of *function signatures*, $\mathcal{R}$ is a set of *predicate signatures*, $\mathcal{O}$ and $\mathcal{C}$ are sets of *objects* which serve as arguments to functions and predicates, $\mathcal{T}$ is a *type hierarchy* over a set of *type names*, $\mathcal{A}$ is a set of *instantaneous action schemas*, $\mathcal{E}$ is a set of *event schemas*, $\mathcal{P}$ is a set of *process schemas*, $\mathcal{I}$ is the *initial state specification*, and $\mathcal{G}$ is a *goal state specification*. Below, we go over these sets in detail.

Names   A PDDL+ instance I uses the set of alphanumeric strings as a namespace for types, objects, predicates, actions, etc. In the grammar of PDDL+, the syntactic category `<name>` refers to any member of this namespace. Our translation uses the same namespace for the vocabulary of BAT.

Objects and Constants   The instance I supplies a finite set $\mathcal{O}$ of *objects* — symbols for distinct primitive entities operated upon — and a finite set of *constants*, which are objects hard-coded into the domain and shared between all planning problems for that domain. In SC, the elements of $\mathcal{O} \cup \mathcal{C}$ correspond to constants with unique name axioms and a domain closure axiom.

Types   PDDL+ has two basic types, `object` and `number`. Type `object` acts as a parent type for domain-specific types of objects. The type `number` is not extensible

and is only used as the return type of functions. A finite set of object types is specified as a part of the type hierarchy $\mathcal{T}$.

In our mapping, the types `object` and `number` are not modelled explicitly and instead correspond to terms of the sorts OBJECT and $\mathbb{R}$, respectively. Each defined type corresponds to a unary predicate of the same name whose argument is of sort OBJECT. In slight abuse of notation, we write $T \in \mathcal{T}$ to denote a primitive type name occurring in $\mathcal{T}$.

The type hierarchy $\mathcal{T}$ is specified by an instance of the syntactic category `<typed list (name)>`, i.e., an expression of the form $(T_1 - TE_1 \ldots T_n - TE_n)$ where each T_i is a primitive type name and each TE_i is either a primitive type name or a structure of the form `(either `$Y_1 \ldots Y_k$`)` where $k \geq 1$ and Y_j are primitive type names. We translate $\mathcal{T}$ to SC as follows. For each $1 \leq i \leq n$, we include an axiom $\forall x(\widehat{T_i} \rightarrow \widehat{TE_i})$, with the mapping function $\widehat{\cdot}$ defined in Table 1. As each object $O \in \mathcal{O} \cup \mathcal{C}$ is supplied with its own type expression TE, we include an axiom $\widehat{TE}(O)$ to associate O with its type.

Variables   PDDL+ variables are names prepended with a question mark, used as placeholders for the objects in action/event/process schemas and goal description quantifiers. We map variables to SC variables according to Table 1.

Predicates $\mathcal{R}$ is a non-empty finite set of predicate signatures of the form $(R\ ?x_1 - TE_1 \ldots ?x_n - TE_n)$ where R is the predicate name, $?x_i \ldots ?x_n$ are variable names, and TE_i is a type expression for the i-th argument of R. As before, the type expression TE_i is either the name of a primitive type from $\mathcal{T}$ or a structure of the form `(either `$T_1 \ldots T_k$`)` where $k \geq 1$ and $T_i \in \mathcal{T}$ for $1 \leq i \leq k$. Let $arity(R)$ denote n, the arity of R as specified by $\mathcal{R}$.

In PDDL+, the dynamic nature of predicates is dictated by the effects of actions/events/processes. In SC, we make this distinction formal. We partition $\mathcal{R}$ into the mutually exclusive sets $\mathcal{R}_s$ ("static") and $\mathcal{R}_d$ ("dynamic") such that $\mathcal{R}_d = \{R \in \mathcal{R} \mid R$ appears in effects of some action/event$\}$ and $\mathcal{R}_s = \mathcal{R} \setminus \mathcal{R}_d$. The predicate names in $\mathcal{R}_s$ become SC static (rigid) predicates with the signature $\text{OBJECT}^{arity(R)}$. The symbols in $\mathcal{R}_d$ become SC relational fluents with the signature $\text{OBJECT}^{arity(R)} \times \text{SITUATION}$.

Functions $\mathcal{F}$ is a possibly empty finite set of function signatures of the form $(F\ ?x_1 - TE_1 \ldots ?x_n - TE_n)$ where F is the function name, $?x_1 \ldots ?x_n$ are variable names, and TE_i is the type description of the i-th argument of F as in the case for predicates. Let $arity(F)$ denote the arity of F as specified by $\mathcal{F}$. As with predicates, we partition PDDL+ functions into classes based on how they change. $\mathcal{F}_t$ ("temporal") are exactly those members F of $\mathcal{F}$ such that `(increase (`F` ...) ...)` occurs in some process schema $\mathcal{P}$, i.e., the functions which undergo continuous evolution. The function signatures in $\mathcal{F}_t$ correspond to SC temporal fluents with the signature $\text{OBJECT}^{arity(R)} \times \mathbb{R} \times \text{SITUATION} \mapsto \mathbb{R}$. Additionally, each member of $\mathcal{F}_t$ gives rise to a related fluent F_{init} with the signature $\text{OBJECT}^{arity(R)} \times \text{SITUATION} \mapsto \mathbb{R}$. $\mathcal{F}_d$ ("dynamic") are those members $F \in (\mathcal{F} \setminus \mathcal{F}_t)$ which can be affected by some instantaneous event or action from $\mathcal{A} \cup \mathcal{E}$.

The members of $\mathcal{F}_d$ correspond to SC functional fluents with the signature $\textsc{Object}^{arity(F)} \times \textsc{Situation} \mapsto \mathbb{R}$. Finally, $\mathcal{F}_s = \mathcal{F} \setminus (\mathcal{F}_d \cup \mathcal{F}_t)$ are the inherently static functions which are mapped to SC static functions with the signature $\textsc{Object}^{arity(F)} \mapsto \mathbb{R}$. We write $\widehat{\mathcal{F}}, \widehat{\mathcal{F}}_t$, etc. to denote sets of SC terms which correspond to the respective PDDL+ function signatures.

Goal Descriptions The category `<GD>` (*goal description*) represents a FOL-like statement used to describe an action precondition or a goal condition. The grammar of `<GD>` defines it as an expression constructed from predicate and (in)equality atoms on functions (but not process atoms) using standard logical connectives and quantifiers. Table 1 summarizes the mapping of arbitrary goal descriptions X to the expressions $\widehat{X}$ of situation calculus. It can be easily shown that (a) $\exists t(t \geq start(s) \wedge \widehat{X})$ is a well-formed SC formula uniform in s, and (b) whenever a PDDL+ state and a SC structure with an object assignment agree on the values of all terms and atoms, they interpret X and $\exists t(t \geq start(s) \wedge \widehat{X})$ (respectively) identically.

Table 1: A mapping from PDDL+ to SC expressions. On the right, x is a variable of sort $\textsc{Object}$, y a variable of sort $\mathbb{R}$, t a variable of sort $\mathbb{R}$ (for time), and s is a variable of sort $\textsc{Situation}$. Here and subsequently, $\phi(x/x_i)$ denotes the result of replacing x by x_i in ϕ.

X	$\widehat{X}$
$?x$	variable x of sort $\textsc{Object}$
$O \in \mathcal{O} \cup \mathcal{C}$	constant O of sort $\textsc{Object}$
numeric literal n	n
$T \in \mathcal{T}$	$T(x)$
$(\texttt{either}\ T_1 \dots T_n)$	$(T_1(x) \vee \dots \vee T_n(x))$
$(R\ b_1 \dots b_n), R \in \mathcal{R}_s$	$R(\widehat{b}_1 \cdots \widehat{b}_n)$
$(R\ b_1 \dots b_n), R \in \mathcal{R}_d$	$R(\widehat{b}_1 \cdots \widehat{b}_n, s)$
$(F\ b_1 \dots b_n), F \in \mathcal{F}_s$	$F(\widehat{b}_1 \cdots \widehat{b}_n)$
$(F\ b_1 \dots b_n), F \in \mathcal{F}_d$	$F(\widehat{b}_1 \cdots \widehat{b}_n, s)$
$(F\ b_1 \dots b_n), F \in \mathcal{F}_t$	$F(\widehat{b}_1 \cdots \widehat{b}_n, t, s)$
$(\texttt{and}\ Y_1 \dots Y_n)$	$(\widehat{Y}_1 \wedge \dots \wedge \widehat{Y}_n)$
$(\texttt{or}\ Y_1 \dots Y_n)$	$(\widehat{Y}_1 \vee \dots \vee \widehat{Y}_n)$
$(\texttt{not}\ Y)$	$\neg(\widehat{Y})$
$(\texttt{imply}\ Y_1\ Y_n)$	$(\widehat{Y}_1) \rightarrow (\widehat{Y}_n)$
$(\texttt{exists}\ (?x_1\text{-}TE_1 \dots ?x_n\text{-}TE_n)\ Y)$	$\exists x_1 \dots \exists x_n$ $(\bigwedge_{1 \leq i \leq n} \widehat{TE}_i(x/x_i) \wedge (\widehat{Y}))$
$(\texttt{forall}\ (?x_1\text{-}TE_1 \dots ?x_n\text{-}TE_n)\ Y)$	$\forall x_1 \dots \forall x_n$ $(\bigwedge_{1 \leq i \leq n} \widehat{TE}_i(x/x_i) \rightarrow (\widehat{Y}))$
$(\texttt{-}\ Y)$	$-(\widehat{Y})$
$(\circ\ Y_1\ Y_2), \circ \in \{+, -, \times, \div, >, <, =, \leq, \geq\}$	$(\widehat{Y}_1 \circ \widehat{Y}_n)$

Actions $\mathcal{A}$ is a set of schemas of the form

```
(:action A
:parameters  (?x_1 - TE_1 ... ?x_n - TE_n)
:precondition Pre_A
:effect Eff_A)
```

where A is the action name, Pre_A is an instance of `<GD>`,

Table 2: An auxiliary mapping from instances of `<effect>` for a PDDL+ action $A(\bar{z})$ to SC expressions. $A(\bar{z}, t)$ is a SC action term corresponding to the PDDL+ action $A(\bar{z})$. Let $RelOp$ be $\{\texttt{increase}, \texttt{decrease}, \texttt{scale-up}, \texttt{scale-down}\}$.

X	$\widetilde{X}$
$(R\ b_1 \dots b_n), R \in \mathcal{R}_d$	$\exists t R(\widehat{b}_1 \cdots \widehat{b}_n, do(A(\bar{z}, t), s))$
$(F\ b_1 \dots b_n), F \in \mathcal{F}_d$	$\exists t F(\widehat{b}_1 \cdots \widehat{b}_n, do(A(\bar{z}, t), s))$
$(F\ b_1 \dots b_n), F \in \mathcal{F}_t$	$\exists t F_{init}(\widehat{b}_1 \cdots \widehat{b}_n, do(A(\bar{z}, t), s))$
$(\texttt{forall}\ (?x_1 \dots ?x_n)\ Y)$	$\forall x_1 \cdots \forall x_n(\widetilde{Y})$
$(\texttt{when}\ Y_1\ Y_2)$	$(\widehat{Y}_1 \rightarrow \widetilde{Y}_2)$
$(\texttt{assign}\ Y_1\ Y_2)$	$\forall y(\widehat{Y}_2 = y \rightarrow \widetilde{Y}_1 = y)$
$(\circ\ Y_1\ Y_2), \circ \in RelOp$	$\forall y(\widehat{Y}_1 \widetilde{\circ} \widehat{Y}_2 = y \rightarrow \widetilde{Y}_1 = y)$
$\circ \in RelOp$	$+/-/\times/\div$
X not appearing above	$\widehat{X}$

and Eff_A is an instance of `<effect>`. According to (McDermott et al. 1998), all variables that occur in a schema must either appear under `:parameters` or be quantified. Each such schema generates a SC action $A(x_1, \dots, x_n, t)$ with the mandatory axiom $time(A(\bar{x}, t)) = t$ and a unique name axiom $A(\bar{u}) = A(\bar{v}) \rightarrow \bar{u} = \bar{v}$. The precondition Pre_A and the parameter types TE_i, when translated according to Table 1, yield a SC precondition axiom

$$Poss(A(x_1, \dots, x_n, t), s) \leftrightarrow \bigwedge_{1 \leq i \leq n} \widehat{TE}_i(x/x_i) \wedge \widehat{Pre}_A.$$

It is clear that the right-hand side of this axiom is uniform in s and contains no free variables except for those shown.

Action effects are specified by the category `<effect>`, but are usually treated as sets of ground atoms, produced by explicit flattening and grounding as per Definition 5 in (Fox and Long 2003). Fortunately, we can avoid grounding in favour of conciseness. Let $\widetilde{\ }$ be a mapping from PDDL+ to SC expressions as defined in Table 2.

Theorem 2. *Given an instance Eff_A of `<effect>`, $\widetilde{Eff}_A$ is a SC formula which is logically equivalent to a conjunction of axioms of the following forms:*

$$a = A(\bar{x}, t) \wedge \Psi(\bar{x}, \bar{z}, t, s) \rightarrow P(\bar{x}, \bar{z}, do(a, s)),$$
$$a = A(\bar{x}, t) \wedge \Psi(\bar{x}, \bar{z}, t, s) \rightarrow \neg P(\bar{x}, \bar{z}, do(a, s)),$$
$$a = A(\bar{x}, t) \wedge \Psi(\bar{x}, y, \bar{z}, t, s) \rightarrow F(\bar{x}, \bar{z}, do(a, s)) = y.$$

Such axioms are proper SC effect axioms that can be compiled into SSA in the usual way following Reiter's solution to the frame problem (Reiter 1991). Note that whenever no action or event (see below) affects a function F from $\mathcal{F}_t$, the above method fails to supply the fluent F_{init} with a SSA. Thus, for each $F(\bar{x}, t, s) \in \widehat{\mathcal{F}}_t$ unaffected by actions or events, our BAT must include a trivial SSA $F_{init}(\bar{x}, do(a, s)) \leftrightarrow F(\bar{x}, time(a), s)$.

It can be shown that the PDDL+ notion of an *applicable* action coincides with the SC notion of an *executable* action. Furthermore, whenever a PDDL+ state and a SC structure with an object assignment agree on the values of all terms

and atoms, they also agree on the interpretation of the successor state, as determined by $\mathcal{A}$ and the SSA obtained from the axioms above, respectively.

Events PDDL+ events are described by a set of schemas whose grammar coincides with that of action schemas and are therefore mapped to SC in the same way. However, events have a different semantics in that they occur spontaneously. To capture this semantics, we appeal to Reiter's notion of natural actions and declare each SC action $E(\bar{x}, t)$ which originates from a PDDL+ event as a natural one using the axiom $natural(E(\bar{x}, t))$. Since Reiter defines natural actions only for theories with true concurrency which PDDL+ lacks, we redefine $executable(s)$ accordingly:

$$\forall a, s' \big[do(a, s') \sqsubseteq s \to Poss(a, s') \wedge start(s') \leq time(a) \big] \wedge$$
$$\forall a, a', s' \big[natural(a) \wedge Poss(a, s') \wedge do(a', s') \sqsubseteq s \to$$
$$a = a' \vee (natural(a') \wedge time(a') \leq time(a)) \big].$$

Here, natural actions occur as soon as possible (at the true time instant t), but in some arbitrary order such that no agent action can appear in between two natural actions which occur at the same time instant.

Processes $\mathcal{P}$ is a set of schemas of the form

```
(:process P
:parameters  (?x₁ - TE₁ ... ?xₙ - TEₙ)
:precondition Preₚ
:effect Effₚ)
```

where P is the process name, Pre_P is an instance of `<GD>`, and Eff_P is an instance of `<process-effect>`. Each such schema generates the following SC symbols:

- a relational fluent $P(x_1, \ldots, x_n, s)$ (denoted $\widehat{P}$) and

- actions $begin_P(x_1, \ldots, x_n, t)$ and $end_P(x_1, \ldots, x_n, t)$.

The process-induced relational fluents $P(x_1, \ldots, x_n, s)$ are to compose the SEA contexts which determine how continuous quantities evolve based on which processes are active in the given state. The actions are natural and act as events which initiate and terminate the process in an explicit syntactic representation of the *start-process-stop* model. The following axioms model this behaviour.

$$Poss(begin_P(\bar{x}, t), s) \leftrightarrow \bigwedge_{1 \leq i \leq n} \widehat{TE}_i(x/x_i) \wedge \widehat{Pre}_P,$$
$$Poss(end_P(\bar{x}, t), s) \leftrightarrow \bigwedge_{1 \leq i \leq n} \widehat{TE}_i(x/x_i) \wedge \neg \widehat{Pre}_P,$$
$$P(\bar{x}, do(a, s)) \leftrightarrow$$
$$\exists t(a = begin_P(\bar{x}, t)) \vee P(\bar{x}, s) \wedge \neg \exists t(a = end_P(\bar{x}, t)).$$

Process effects are specified by an instance Eff_P of the category `<process-effect>`. Without loss of generality, we consider the following simplified grammar:

```
<process-effect> ::= (and <p-eff-t>*)
<p-eff-t> ::= (increase <f-head> <f-exp-t>)
<f-head> ::= (<function-symbol> <term>*)
<f-exp-t> ::= (* #t <f-exp>)
```

Crucially, each process effect defined thereby is *relative*, in that it describes but a part of the continuous evolution of a quantity, the part induced by P alone. To achieve axiomatic

causal completeness, our mapping must combine effects due to *all* processes into a universal temporal change axiom and, subsequently, SEA for each continuous fluent.

Recall that $\mathcal{F}_t$ are exactly the symbols that instantiate `<f-head>` in `<p-eff-t>` above. Consider the set

$$Terms_P = \{\widehat{X} \mid (\texttt{increase } X \texttt{ (...)}) \in Eff_P\}.$$

of all function terms affected by process P. For each $\widehat{X} \in Terms_P$ where the function name is F, let $\theta_{\widehat{X}, P}$ be an arbitrary one-to-one variable substitution from the object variables of $\widehat{P}$ to the set $\{x_1, \ldots, x_{arity(P)}\}$ such that $\widehat{X}\theta_{\widehat{X}, P} = F(x_1, \ldots, x_{arity(F)}, t, s)$. For each $\widehat{X} \in Terms_P$, consider the set $Parts_{\widehat{X}, P}$ of tuples $(\widehat{P}\theta_{\widehat{X}, P}, \widehat{Y}\theta_{\widehat{X}, P})$ such that $(\texttt{increase } X \texttt{ (* #t } Y\texttt{)})$ occurs in Eff_P. Each pair in this set describes a context and a continuous effect that it induces on $\widehat{X}$, normalized wrt the arguments of the term $\widehat{X}$. For each such tuple $T = (P(\bar{z}, s), \phi(\bar{z}', t, s)) \in Parts_{\widehat{X}, P}$, according to the grammar, we have $\bar{z}' \subseteq \bar{z}$ and $\bar{x} \subseteq \bar{z}$. Let $\bar{w}$ denote the variables contained in $\bar{z}$ but not in $\bar{x}$. Let $Gnd(T)$ denote the set of all tuples which can be obtained from the tuple T by simultaneously replacing all variables $\bar{w}$ with constants from $\mathcal{O} \cup \mathcal{C}$ in all possible ways (thereby eliminating $\bar{w}$). Let $GndParts_{\widehat{X}, P} = \{Gnd(T) \mid T \in Parts_{\widehat{X}, P}\}$. Finally, for each $F \in \mathcal{F}_t$, we combine all such tuples due to all processes which affect F in a single set:

$$B_F = \bigcup_{P \in \mathcal{P}, \widehat{X} \in Terms_F} GndParts_{\widehat{X}, P}.$$

We can now generate the complete TCA in normal form for the fluent F, where 2^{B_F} is the power set of B_F:

$$\bigvee_{b_F \in 2^{B_F}} \Big[\bigwedge_{\langle X, Y \rangle \in b_F} (X) \wedge \bigwedge_{\langle X, Y \rangle \in B_F \setminus b_F} (\neg X) \wedge$$
$$y = F_{init}(\bar{x}, s) + \int_{start(s)}^{t} \sum_{\langle X, Y \rangle \in b_F} Y(t/t') \, dt' \Big]$$
$$\to F(\bar{x}, t, s) = y.$$

By construction, exactly one context of this TCA is satisfied in any given FOL structure. Thus, the TCA is causally complete and can be transformed into a correct SEA by simply making the implication bidirectional.

Durative Actions As shown in Appendix D of (Fox and Long 2006), durative actions can be seen in PDDL+ as syntactic sugar for a combination of two agent actions, an event, and a process, all of which we capture above. Thus, indirectly, the mapping supports the requirement `:durative-actions`.

Timed Initial Intervals Timed initial intervals (TIL) first appeared in PDDL2.2 (Edelkamp and Hoffmann 2004) and are not a part of the original PDDL+ proposal, but commonly occur in practice. They offer a means to concisely express a special kind of exogenous events which are destined to occur at prespecified times. We capture these using special-purpose natural actions—one per time point—and add the obvious effect axioms to the SSA of the fluents involved. Thus, our mapping also supports `:timed-initial-intervals`.

Initial State The PDDL+ initial state is specified by a set of ground atoms and equalities between ground terms and numbers. These are trivially translated to SC using Table 1 while substituting S_0 for the situation term, 0 for the time argument, and asserting that $start(S_0) = 0$.

Goal State A PDDL+ goal state is an arbitrary instance of `<GD>` already handled by the mapping, yielding a formula $Goal(s, t)$. To ensure that all timed initial intervals get a chance to occur, we use the time T of the latest TIL (or $T = start(S_0)$ if no TILs are given) to constrain the goal formula: $\exists s \exists t(t \geq start(s) \land t \geq T \land Goal(s, t))$.

This completes the translation and yields a hybrid BAT $\mathcal{D}$ which concisely characterizes both a planning instance and its Tarskian semantics. Consequently, it can be directly tested for consistency and satisfiability via automated theorem proving. The PDDL+ notion of a plan corresponds to a SC ground situation, i.e., a sequence of ground actions, which are already annotated with timestamps by virtue of having a temporal argument. The planning task is then defined as finding a ground situation term σ over the vocabulary of $\mathcal{D}$ such that $\mathcal{D} \models executable(\sigma) \land \exists t(t \geq start(\sigma) \land t \geq T \land Goal(\sigma, t))$. Here, $executable(\sigma)$ ensures that all preconditions, including those of natural actions, are met.

Example. For illustration, consider a representative fragment of the Car domain from (Piotrowski et al. 2016), which models the movement of a car. The domain symbols are mapped as follows (we shorten the names for brevity).

$$
\begin{aligned}
\texttt{running} &\mapsto run(s) & \in \mathcal{F}_d \\
\texttt{goal\_reached} &\mapsto goalr(s) & \in \mathcal{F}_d \\
\texttt{engineBlown} &\mapsto engb(s) & \in \mathcal{F}_d \\
\texttt{accelerate} &\mapsto accel(t) & \in \mathcal{A} \\
\texttt{decelerate} &\mapsto decel(t) & \in \mathcal{A} \\
\texttt{stop} &\mapsto stop(t) & \in \mathcal{A} \\
\texttt{engineExplode} &\mapsto expl(t) & \in \mathcal{E} \\
\texttt{a} &\mapsto A(s) & \in \mathcal{F}_d \\
\texttt{up\_limit} &\mapsto up\_limit & \in \mathcal{F}_s \\
\texttt{down\_limit} &\mapsto down\_limit & \in \mathcal{F}_s \\
\texttt{moving} &\mapsto m(s) & \in \mathcal{P} \\
\texttt{windResistance} &\mapsto w(s) & \in \mathcal{P} \\
\texttt{v} &\mapsto V(t, s) & \in \mathcal{F}_t \\
& V_{init}(s) & \text{auxiliary} \\
\texttt{d} &\mapsto D(t, s) & \in \mathcal{F}_t \\
& D_{init}(s) & \text{auxiliary}
\end{aligned}
$$

Additionally, the mapping extends the vocabulary with the natural actions $begin_m(t)$, $end_m(t)$, $begin_w(t)$, $end_w(t)$ which initiate and terminate the processes. The action preconditions are as follows:

$$
\begin{aligned}
Poss(accel(t), s) &\leftrightarrow run(s) \land A(s) < up\_limit, \\
Poss(decel(t), s) &\leftrightarrow run(s) \land A(s) > down\_limit, \\
Poss(stop(t), s) &\leftrightarrow V(t, s) = 0 \land D(t, s) \geq 30, \\
Poss(expl(t), s) &\leftrightarrow run(s) \land A(s) \geq 1 \land V(t, s) \geq 100,
\end{aligned}
$$

$$
\begin{aligned}
Poss(begin_m(t), s) &\leftrightarrow run(s), \\
Poss(end_m(t), s) &\leftrightarrow \neg run(s), \\
Poss(begin_w(t), s) &\leftrightarrow run(s) \land V(t, s) \geq 50, \\
Poss(end_w(t), s) &\leftrightarrow \neg(run(s) \land V(t, s) \geq 50).
\end{aligned}
$$

The latter four actions are natural and are asserted as such, e.g., by the axiom $natural(end_w(t))$.

Relational atemporal predicates get the SSA compiled from action effects and implicit frame axioms:

$$
\begin{aligned}
run(do(a, s)) &\leftrightarrow run(s) \land \neg \exists t[a = expl(t)], \\
goalr(do(a, s)) &\leftrightarrow \exists t[a = stop(t)] \lor goalr(s), \\
engb(do(a, s)) &\leftrightarrow \exists t[a = expl(t)] \lor engb(s).
\end{aligned}
$$

Process fluents are also atemporal and so are described similarly, e.g. $m(do(a, s)) \leftrightarrow \exists t[a = begin_m(t)] \lor m(s) \land \neg \exists t[a = end_m(t)]$.

The action effects for $\texttt{a} \mapsto A(s)$ yield SC effect axioms

$$
\begin{aligned}
\exists t[a = accel(t) \land y = A(s) + 1] &\to A(do(a, s)) = y, \\
\exists t[a = decel(t) \land y = A(s) - 1] &\to A(do(a, s)) = y, \\
\exists t[a = explode(t) \land y = 0] &\to A(do(a, s)) = y,
\end{aligned}
$$

which are likewise compiled into a SSA

$$
A(do(a, s)) = y \leftrightarrow \phi(y, a, s) \lor y = A(s) \land \neg \exists y'\, \phi(y', a, s),
$$

where $\phi(y, a, s)$ is the disjunction of the left-hand sides of the effect axioms.

Although acceleration, velocity, and distance are all continuous quantities, only the latter two undergo continuous evolution in the Car domain. Velocity changes trivially across situations:

$$
V_{init}(do(a, s)) = y \leftrightarrow V(time(a), s) = y,
$$

but non-trivially within situations. In the SEA, the four Boolean combinations of $m(s)$ and $w(s)$ constitute the discrete contexts, and the corresponding evolutions are arithmetic combinations of evolutions due to each active process:

$$
\begin{aligned}
V(t, s) = y \leftrightarrow \exists v_0 . v_0 = V_{init}(s) \land \\
[\neg m(s) \land \neg w(s) \land y = v_0 \\
\lor m(s) \land \neg w(s) \land y = v_0 + \int_{start(s)}^{t} A(s)\, dt' \\
\lor \neg m(s) \land w(s) \land y = v_0 - \int_{start(s)}^{t} \frac{(v(t', s) - 50)^2}{10}\, dt' \\
\lor m(s) \land w(s) \land y = v_0 + \int_{start(s)}^{t} A(s) - \frac{(v(t', s) - 50)^2}{10}\, dt'].
\end{aligned}
$$

The distance is axiomatized similarly by a SSA for the atemporal fluent $D_{init}(s)$ and a SEA for the temporal fluent $D(t, s)$.

Since the process m is triggered by the initial state (not shown), every executable situation begins with the natural action $begin_m(0)$. For example, the goal $\exists t(t \geq start(\sigma) \land v(t, \sigma) \geq 10)$ is achieved, among others, by the plan $\sigma = do([begin_m(0), accelerate(2)], S_0)$ with t bound to 12.

Discussion

We proposed an extension of situation calculus which aligns closely with the widely accepted way of representing continuous change in hybrid systems. We then leveraged it to provide a Tarskian semantics to PDDL+. While hybrid automata have a solid foundation in logic (Nerode 2007) and notwithstanding the progress toward properly mapping PDDL+ to hybrid automata (Bogomolov et al. 2014), the PDDL+ semantics, due its focus on grounding, invokes HA in ways which obscure the connection between PDDL+, HA, and logic, putting more emphasis on fine-grained transition system semantics. The lifted semantics proposed in the present paper allows us to sidestep most of the special purpose semantic devices such as happenings and time slippage and remain at a high level of abstraction.

In principle, the proposed hybrid BATs support significantly richer domains than can be specified by PDDL+. For example, SC is bound by neither the closed world nor the domain closure assumption, allowing for domains where objects can be brought in and out of existence. Likewise, hybrid BATs support dynamic systems described by partial DE which are inexpressible by current PDDL+ process effects. At the same time, Tarskian semantics provides for direct verification of various interesting meta-properties of a planning instance, from the essential properties such as satisfiability (due to the relative satisfiability property of hybrid BATs) to testing for mutually exclusive events or establishing a particular relation between executable subsitutions. Moreover, a SC axiomatization makes formal causal analysis (Batusov and Soutchanski 2018) applicable to PDDL+ domains.

Notably, PDDL+ does not have a semantics for truly concurrent events, allowing only interleaved concurrency. (Reiter 2001) points out examples where interleaved concurrency in SC is not appropriate and develops a theory of true concurrency. In PDDL+, if two events were to occur at the same instant, they would be arbitrarily forced into an order with possibly undesirable consequences. (Fox and Long 2006) sidestep the problem by considering only *event-deterministic* instances, i.e., those where such nondeterministic choice of order is of no consequence. Although our mapping stays true to the notion of interleaved concurrency for the sake of providing correct semantics, it is equipped to lift this limitation.

Since SC makes all assumptions explicit, it bears one notable syntactic difference to source PDDL+. In particular, the meta-theoretic causal completeness assumption of PDDL+ allows for a very concise description of *relative* process effects. SC affords no such luxury and must list all permutations of all process fluents which may affect a temporal fluent F in the state evolution axiom for F. For a worst-case example, consider the road traffic domain of (Vallati et al. 2016). Each road intersection has a set of traffic lights characterized by the process $FlowGreen(r_1, r_2, i)$ parametrized by the relevant road segments r_1, r_2 and the intersection i. If the road network of a city contains hundreds of intersections, the length of a permutation of all groundings of $FlowGreen$ becomes unmanageable for all practical purposes. However, if one notes (or derives via automated theorem-proving) that only a small fraction of road segments may ever be affected by a particular grounding of $FlowGreen$ (i.e., the maximum number of roads that meet at any one intersection), the SEA is reduced to a manageable size. This appears to be a case of a *sparse predicate* (Franco et al. 2018) which presents a serious difficulty to PDDL+ planners as well, except it manifests in the explosion of search spaces rather than axioms.

Relevant and Future Work

A variety of approaches have been applied to PDDL+ planning. (Della Penna et al. 2009) put forth UPMurphi, an optimal planner based on model checking and blind forward search. They pioneer the discretise-and-validate approach, a method of planning in a discrete approximation of the continuous domain. In case studies (Vallati et al. 2016; Fox, Long, and Magazzeni 2012), UPMurphi solved challenging problems by relying on domain-specific heuristics. (Bryce et al. 2015), SMTPlan+ (Cashmore et al. 2016) and DiNo (Piotrowski et al. 2016) are the first state-of-the art non-linear planners with full coverage of PDDL+. SMTPlan+ operates by encoding a PDDL+ instance as an SMT problem, inspired by prior successes of non-linear SMT solvers in hybrid systems. Like prior work on PDDL using SAT, this approach reaches state-of-the-art performance by leveraging an existing advanced SMT solver. DiNo builds on and outperforms UPMurphi using a domain-independent heuristic. (Balduccini, Magazzeni, and Maratea 2016) implement a prototype planner using Constraint Answer Set Programming. They achieve noteworthy results, showing that logic programming is a viable approach to PDDL+ planning, especially with domain-specific declarative heuristics. (To et al. 2017) introduce MTP, a planner for a new formalism, which is a hybrid of SMT and a metric temporal logic which subsumes PDDL+. Despite allowing goals with rich temporal properties, MTP outperforms SMTPlan+ on standard and new domains. MTP employs a heuristic based on achieving partial goals, which are inexpressible in PDDL+.

All this work points to the fact that PDDL+ planning is significantly more involved than the purely discrete case and might benefit from moving further away from grounding. Recent research focuses on ways of reducing the difficulty by improving modelling techniques (Denenberg and Coles 2018) or automatically reducing the arity of predicates with large numbers of possible groundings (Franco et al. 2018). Although generally less tractable than the PDDL family of languages, situation calculus has been used successfully for planning purposes in (Finzi et al. 2000). There, the focus of the `wspdf` planner was to exploit the expressiveness of SC to extend practical planning to open worlds. We posit that native SC planning based on theorem proving holds promise for PDDL+ domains as well. We imagine that an effective SC planner can be constructed by supplementing `wspdf` with a non-linear optimizer such as IPopt (Wächter and Biegler 2006), resulting in an iteratively deepening forward search interleaved with solving initial value problems. Tractability of theorem proving can be achieved by considering *bounded* hybrid BATs along the lines of (De Giacomo, Lespérance, and Patrizi 2016). The present paper can be considered as a step in this direction.

References

Balduccini, M.; Magazzeni, D.; and Maratea, M. 2016. PDDL+ planning via constraint answer set programming. *arXiv preprint arXiv:1609.00030*.

Batusov, V., and Soutchanski, M. 2018. Situation calculus semantics for actual causality. In McIlraith, S. A., and Weinberger, K. Q., eds., *Proceedings of AAAI-2018*, 1744–1752.

Batusov, V.; De Giacomo, G.; and Soutchanski, M. 2018. Hybrid temporal situation calculus. *CoRR* abs/1807.04861.

Boddy, M. S. 2003. Imperfect match: PDDL 2.1 and real applications. *Journal of Artificial Intelligence Research* 20:133–137.

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as model checking in hybrid domains. In Brodley, C. E., and Stone, P., eds., *Proceedings of AAAI-2014*, 2228–2234.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. SMT-based nonlinear PDDL+ planning. In *AAAI*, 3247–3253. AAAI Press.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A compilation of the full PDDL+ language into SMT. In Coles, A. J.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of ICAPS-2016*, 79–87.

Claßen, J.; Hu, Y.; and Lakemeyer, G. 2007. A situation-calculus semantics for an expressive fragment of PDDL. In *Proceedings of AAAI-2007*, 956–961.

De Giacomo, G.; Lespérance, Y.; and Patrizi, F. 2016. Bounded situation calculus action theories. *Artif. Intell.* 237:172–203.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A tool for universal planning on PDDL+ problems. In Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds., *Proceedings of ICAPS-2009*.

Denenberg, E., and Coles, A. 2018. Modelling Sequences of Processes in PDDL+ for Efficient Problem Solving. In *Proceedings of KEPS-2018*, 70–77.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classical part of the 4th international planning competition. *4th International Planning Competition (IPC'04), at ICAPS'04*.

Finzi, A.; Pirri, F.; Reiter, R.; et al. 2000. Open world planning in the situation calculus. In *AAAI/IAAI*, 754–760.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based policies for efficient multiple battery load management. *Journal of Artificial Intelligence Research* 44:335–382.

Franco, S.; Vallati, M.; Lindsay, A.; and McCluskey, T. L. 2018. Improving Planning Performance in PDDL+ Domains via Automatic Predicates Reformulation. In *Proceedings of KEPS-2018, Delft, the Netherlands, 2018*, 6–10.

Lakemeyer, G., and Levesque, H. J. 2004. Situations, si! situation terms, no! In Dubois, D.; Welty, C. A.; and Williams, M., eds., *Proceedings of of KR-2004*, 516–526.

Lifschitz, V. 1987. On the semantics of STRIPS. In *Proceeding of the 1986 Workshop: Reasoning About Actions & Plans*. Elsevier. 1–9.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL–the planning domain definition language. Technical report, Yale University, CVC TR98003/DCS TR1165.

McDermott, D. 2003. PDDL2. 1–The Art of the Possible? Commentary on Fox and Long. *Journal of Artificial Intelligence Research* 20:145–148.

Nerode, A. 2007. Logic and control. In Cooper, S. B.; Löwe, B.; and Sorbi, A., eds., *Computation and Logic in the Real World, CiE 2007, Proceedings*, volume 4497 of *Lecture Notes in Computer Science*, 585–597. Springer.

Pinto, J. A. 1994. *Temporal reasoning in the situation calculus.* Ph.D. Dissertation, University of Toronto.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic planning for PDDL+ domains. In *AAAI Workshop: Planning for Hybrid Systems*, volume WS-16-12 of *AAAI Workshops*.

Pirri, F., and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *Journal of the ACM (JACM)* 46(3):325–361.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy* 27:359–380.

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press Cambridge.

Ternovskaia, E. 1994. Interval situation calculus. In Schwind, C., ed., *Proc. of the ECAI Workshop W5 on Logic and Change*, 153–164. Amsterdam: 11th ECAI, Aug 8-12.

Teschl, G. 2012. *Ordinary Differential Equations and Dynamical Systems*. Graduate studies in mathematics. AMS, https://www.mat.univie.ac.at/~gerald/ftp/book-ode/ode.pdf.

To, S. T.; Johnson, B.; Roberts, M.; and Aha, D. W. 2017. A new approach to temporal planning with rich metric temporal properties. In *Proceedings of ICAPS-2017*, 288–296.

Vallati, M.; Magazzeni, D.; De Schutter, B.; Chrpa, L.; and McCluskey, T. L. 2016. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *Proceedings of AAAI-2016*, 3188–3194.

Wächter, A., and Biegler, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106(1):25–57.