Numeric Planning with Disjunctive Global Constraints via SMT

Enrico Scala[1] **Miquel Ramirez**[1] **Patrik Haslum**[1,2] **Sylvie Thiebaux**[1,2]
[1]Research School of Computer Science, ANU
[2]National ICT Australia
Canberra ACT 2601, Australia
firstname.lastname@anu.edu.au

Abstract

This paper describes a novel encoding for sequential numeric planning into the problem of determining the satisfiability of a logical theory T. We introduce a novel technique, orthogonal to existing work aiming at producing more succinct encodings that enables the theory solver to *roll up* an unbounded yet finite number of instances of an action into a single plan step, greatly reducing the horizon at which T models valid plans. The technique is then extended to deal with problems featuring *disjunctive global constraints*, in which the state space becomes a non-convex n dimensional polytope. In order to empirically evaluate the encoding, we build a planner, SPRINGROLL, around a state–of–the–art off–the–shelf SMT solver. Experiments on a diverse set of domains are finally reported, and results show the generality and efficiency of the approach.

Introduction

Planning for realistic domains requires expressive models of the world. While classical planning systems scale up w.r.t. problem size, their limited expressiveness remains and makes it difficult to guarantee that solutions conform to specific constraints, a commonplace requisite for plans to control a real world system. In response to this problem, extensions to classical planning languages have been developed (Fox and Long 2003; Smith, Frank, and Cushing 2008), generating highly expressive, and hard to solve, models. Notable advances have been achieved (Coles et al. 2010; Fox, Long, and Magazzeni 2011; Löhr et al. 2012; Ivankovic et al. 2014; Francès and Geffner 2015), although scalability *and* effective support for complex forms of numeric reasoning still remain challenging when dealt with in an integrated fashion (Dornhege et al. 2012). It is our view that simpler forms of planning, such as the models entailed by PDDL 2.1 Level 2, have not received enough attention. We are convinced that scaling up in these simpler models will help overcoming open challenges in hybrid planning, analogously to the way scaling up in classical planning has helped scale up non–deterministic and partially observable planning (Muise, McIlraith, and Beck 2012; Bonet and Geffner 2014).

This work focuses on sequential numeric planning, where dynamics is governed directly and explicitly by action effects, with continuous and discrete state variables involved in conditions that can be expressed both locally, in an action precondition or the goal, and globally by means of sets of disjunctive constraints. State variables can be either *basic*, set by action effects or initial states, or *derived*, with their valuation in a given state inferred from the global constraints they have to satisfy. This added expressiveness allows us to compactly interconnect plan generation with the dynamics of complex systems, such as those of power systems (Piacentini et al. 2013; Ivankovic et al. 2014). While planners supporting global numeric (Ivankovic et al. 2014) or finite-domain (Francès and Geffner 2015) constraints do exist, these constraints are always given in conjunctive form. By handling directly disjunctive constraints we support a very broad class of domains, where the state space is a non–convex structure, such as the ones entailed by the underwater UAV domain discussed by Li (2011).

This paper contains two contributions. The first one is a novel encoding of classical action theories with numeric effects that exploits the expressive power of certain fragments of first–order logic (CP, SMT, etc.) and aims at greatly reducing the planning horizon at which the theory models valid plans. This reduction is achieved by *rolling up* several instances of an action to be executed in one *single* plan step. We provide a theoretical analysis of the conditions that enable rolling up actions, addressing the interaction of the technique with global constraints. This relationship turns out to be non-trivial, and requires to derive *automatically* specific axioms from the description of action effects and constraints, that guarantee that constraints are not violated while rolling up actions. The second contribution is a formal model of planning tasks that unifies the extensions to classical planning already mentioned, and enables to approach in a uniform, fully declarative way a highly diverse and expressive set of domains. In order to implement the above, we build on top of well-known techniques for compiling classical planning into SAT (Kautz and Selman 1999; Rintanen, Heljanko, and Niemelä 2006) and CP (Lopez and Bacchus 2003) and we present a compilation of this model into SMT (Barrett et al. 2008), a highly expressive decision problem with several publicly available, and efficient solvers such as Z3 (de Moura and Bjorner 2008),

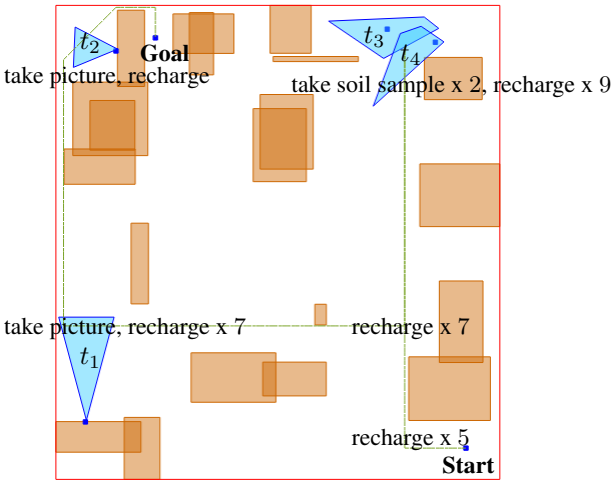

Figure 1: A GEOMETRIC ROVERS example instance, showing the starting and goal locations of the rover, areas where tasks can be performed (blue) and obstacles (orange) and a plan solving the task (green). The red box indicates the bounds of the environment.

which we use off–the–shelf. We build a planner to test our scheme, SPRINGROLL, and measure its performance over several domains previously reported in the literature (Long and Fox 2003; Ivankovic et al. 2014; Francès and Geffner 2015), as well as a novel domain featuring disjunctive global constraints. Results show SPRINGROLL to find plans of length unimaginable with traditional encodings, e.g. on the order of tens of thousands for some domains like COUNTERS (Francès and Geffner 2015).

Motivating Example

Before beginning with the presentation of the details of our approach, it is useful to consider a reformulation of the well known ROVERS domain from the numeric track of the 3rd International Planning Competition (Long and Fox 2003) that exemplifies the challenges posed by adopting a more true to nature representation of planning problems. Figure 1 illustrates an instance of the reformulated ROVERS domain (GEOMETRIC ROVERS from now on). The rover is required to travel between two given locations on the map, performing some tasks along the way. The rover can move along any of the eight compass directions by a certain amount δ_r, a fraction of the extent of the map. The rover battery gets discharged as it moves around, and can be recharged anywhere by a fixed amount, yet this requires the rover to be stationary. We consider two types of tasks: collecting soil samples from designated areas (the quadrilaterals t_3 and t_4) and taking pictures of specific locations, within a maximum distance and relative angle to the target (the triangles t_1 and t_2). The preconditions of the latter actions require the rover to be *inside* the appropriate polygon, and are given by

$$\bigwedge_i A_i x_r + B_i y_r \leq C_i$$

where A_i, B_i and C_i are the coefficients of the line equation corresponding to the i-th edge of the polygon and x_r, y_r are the rover coordinates. No-go areas are modeled by introducing disjunctive global constraints

$$\bigvee_i A_i x_r + B_i y_r \geq C_i$$

that intuitively assert that the rover needs to be outside of them at every point in time.

Figure 1 shows the plan computed by SPRINGROLL in a schematic form. Lines represent movement, text represents tasks and recharging cycles. The plan (with 370 actions and requiring a planning horizon of 14 steps) is found in about 40 seconds and starts by navigating all the way to the area where tasks t_3 and t_4 can be performed, recharging the battery as necessary, then proceeds to perform t_1 and finishes by performing t_2. The number of time steps depends on the number of turns, on the tasks to be performed and on the maximum battery level.

Background and Formalisation

Our model is mostly based on the characterisation of planning with numeric fluents in the PDDL 2.1 (Fox and Long 2003) specification, with some notable extensions (global constraints, derived state variables), dropping the notion of *undefined values* for numeric fluents and adopting a slightly different, but equivalent characterisation of action preconditions and effects. The planning tasks we consider are formally described as follows:

Definition 1 (Planning Problem)**.** *A planning problem* Π *is a tuple* $\langle X^b, X^d, D, s_0, A, G, \mathcal{C}\rangle$ *where:*

- X^b *and* X^d *are state variables (fluents), split into* basic *and* derived *sets,*
- D *is a function mapping* $x \in X^b \cup X^d$ *to possible values,*
- s_0 *the* initial state*, is an assignment to the variables in* $X^b \cup X^d$*,*
- A *is a set of* actions $a = \langle pre_a, eff_a\rangle$*,*
- G *the* goal*, and* $\mathcal{C}$ *the set of* global constraints*, are both CNF formulas over* $X^b \cup X^d$*.*

The previous definition separates the set of state variables into so-called *basic* and *derived* variables depending on whether they are *directly* affected by actions, or not. Unlike previous work on planning with derived predicates (Thiébaux, Hoffmann, and Nebel 2005), but analogously to a more recent extension (Ivankovic et al. 2014), derived variables in our approach can be numeric and their valuation is defined *intensionally*, meaning that they can have any value as long as they satisfy the set of global constraints $\mathcal{C}$. In the problems considered by this paper D maps variables into either the set $\{\top, \bot\}$ or $\mathbb{Q}$. A state of the system is an assignment for each variable $x \in X^b \cup X^d$ to a value $v \in D(x)$.

Action preconditions pre_a are a conjunction of numeric and propositional conditions c. Each condition c can be either a propositional formula over boolean fluents, or an arithmetic formula over numeric fluents

$$\phi \leq \phi' | \phi < \phi' | \phi = \phi' | \phi > \phi' | \phi \geq \phi'$$

ϕ, ϕ' are linear expressions of the form $\sum_i w_i x_i + k$, where w_i and k are rational constants and x_i is a numeric fluent.

We use $lhs(c)$ to refer to ϕ, $rhs(c)$ to refer to ϕ' and $\chi(c)$ to the set of variables in ϕ and ϕ'.

Action effects eff_a are a set of tuples $e = \langle y, \xi \rangle$, $y \in X^b$ being the *affected* variable and ξ the *effect expression*. *Numeric effects* are s.t. ξ is of the form $\sum_i w_i x_i + k$ and $D(x_i) = D(y), \forall x_i$, *propositional* effects are s.t. ξ is a constant $v \in D(y)$[1]. Semantics of effects are defined below. While propositional effects are *state-independent* (as in STRIPS), the interpretation of the numeric effects is *state-dependent*, and we use $[\xi]^s$ to denote the evaluation of the expression ξ in a given state s. A *numeric effect* where all the coefficients w_i are equal to 0 is considered to be an assignment. We use the short-hand $lhs(e)$ to refer to the affected fluent y, and $rhs(e)$ to refer to the expression ξ denoting the next value for y. With $aff(y)$ we refer to the set of actions having an effect e for which $lhs(e) = y$. We assume that each action affects any variable at most once.

Global constraints impose restrictions on the states that can be reached. Global constraints $\mathcal{C}$ in our model are represented by a conjunctive formula (set), where each conjunct $C \in \mathcal{C}$ can be a condition as for an action precondition, or a *disjunction* of conditions $c \vee c'$, defined over sets of numeric fluents, $\chi(C) \subseteq X^b \cup X^d$. This allows us to account for highly expressive planning formalisms as the one introduced recently by Ivankovic et al (2014).

Semantics.We say that an action a can be applied in a state s whenever $s \models pre_a$. If applicable, the execution of a results in a state s' such that:

- $[y]^{s'} = [\xi]^s$ iff $a \in aff(y)$ and $\langle y, \xi \rangle \in eff_a$
- $[y]^{s'} = [y]^s$ iff $a \notin aff(y), y \in X^b$
- $s' \models C$

A solution for Π, or *valid plan*, is a sequence of actions $\pi = \{a_0, a_1, .., a_{n-1}\}$ for which there exists a sequence of states $\{s_0, ..., s_n\}$ such that $s_0 \models pre_{a_0}$, $s_i \in s_{i-1}[a_{i-1}]$ and $s_{i-1} \models pre_{a_{i-1}}$ for each $1 \leq i \leq n$, and $s_n \models G$. We use $s[a]$ to *intensionally* denote the set of successors states of s using a. Each state $s' \in s[a]$ is s.t. it satisfies the action execution conditions as mentioned above (i.e., action effects, inertia and global constraints), $s[a]$ being empty for invalid transitions. Even though $s[a]$ represents a set of states, actions still have to be interpreted as deterministic transitions, where each successor state is implicitly defined according to the satisfiability of conditions imposed by the global constraints and the resulting course of actions. This handling of derived fluents is indeed less intuitive than the inductive representation used by planning languages such as PDDL 2.2 (Edelkamp and Hoffmann 2004; Thiébaux, Hoffmann, and Nebel 2005). On the other hand, this *deductive* formulation allows us to compactly ramify over infinite sets of indirect action effects.

From Planning to SMT

Even if the contributions presented in the next two Sections are independent of specific solvers and modeling languages, so far as the fragment of first-order logic they capture is expressive enough, we have chosen to present them in the context of SMT. The reasons for doing so are two. First, SAT is a framework well-known by the planning community, and we consider SMT to be a parsimonious extension of SAT. SMT is the decision problem for logical formulas w.r.t. combinations of background theories expressed in classical first-order logic plus equality. Amongst the many background theories discussed in the literature, of specific interest and relevance to this paper are the theory of arithmetic over real and integer numbers (Barrett et al. 2008). Second, the SMT community uses a standardised set of tools for modeling problems, and those provide a clear declarative interface between the planning and satisfiability components, facilitating experimentation with existing SMT solvers and adoption of the state–of–the–art as it is pushed forward.

The reduction we are going to present is a generalisation of the boolean SAT encoding (Kautz and Selman 1999). Concerning the classical numeric part it builds on the pioneering work of TM-LPSAT (Shin and Davis 2005), hereby extended to handle global numeric constraints over derived variables and, as we will see, to account for a novel plan execution semantic that allows for the same action to be executed a finite but unbounded number of times in a given plan step. The set of valid plans for the planning problem $\Pi = \langle X^b, X^d, D, s_0, A, G, \mathcal{C} \rangle$ with horizon N is encoded by the theory $T(\Pi, N)$ where, for each time step $i \in \{0,\ldots,N\}$, SMT functions model state variables and the actions (up to $N-1$) to execute. We partition A into two sets, $\mathcal{N}_A$ and $\mathcal{P}_A$. $\mathcal{N}_A$ is the subset of *numerically interesting* actions, namely, those actions a with eff_a featuring at least one effect of the form $\langle y, \sum_i w_i x_i + k \rangle$, whereas $\mathcal{P}_A$ is defined as $A \setminus \mathcal{N}_A$ and denotes the set of actions with no numeric effects. We use $\phi[x/x^i]$ to denote the substitution of each occurrence of all variables $x \in X^b \cup X^d$ in formula ϕ by their corresponding time-indexed function x^i. $T(\Pi, N)$ is then built upon three sets of functions, for every time step: (1) *boolean* or *real* functions x^i, $x \in X^b \cup X^d$, $0 \leq i \leq N$, denoting *values of state variables* at time step i, (2) *integer* functions b^i, $b \in \mathcal{N}_A$,$0 \leq i \leq N-1$, number of instances of an action $b \in \mathcal{N}_A$ executed at time step i, and (3) *boolean* functions a^i, $a \in \mathcal{P}_A$,$0 \leq i \leq N-1$, denoting whether actions $a \in \mathcal{P}_A$ are executed at time step i. $T(\Pi, N)$ axioms, except those accounting for action roll–ups and their interaction with global constraints, which are introduced in the next two sections, are:

A1. **Init**: $x^0 = v$ for $x \in X^b$ and $s_0 \models x = v$, $v \in D(x)$.

A2. **Actions** : For $i = 0, 1, \ldots, N-1$, $a \in \mathcal{P}_A$

- $a^i \Rightarrow pre_a[x/x^i]$
- $a^i \Rightarrow y^{i+1} = v$, for each $\langle y, v \rangle \in eff_a$,

and for actions $b \in \mathcal{N}_A$, $b^i \geq 0$ and

- $b^i = 1 \Rightarrow pre_b[x/x^i]$
- $b^i = 1 \Rightarrow y^{i+1} = \chi$, for each $\langle y, \xi \rangle \in eff_b$

A3. **Frame**: for each $x^i \in X^b$

$$x^i \neq x^{i+1} \Rightarrow \bigvee_{a \in aff(x), a \in \mathcal{P}_A} a^i \vee \bigvee_{b \in aff(x), b \in \mathcal{N}_A} b^i > 0$$

A4. **Interferences**: If a, d interfere, for $i = 0, \ldots, N-1$

[1] Note that this formulation subsumes STRIPS effects. It suffices to consider the domain of y made up of $\{\top, \bot\}$

- $\neg a^i \vee \neg d^i$, $a, d \in \mathcal{P}_A$,
- $\neg a^i \vee d^i = 0$, $a \in \mathcal{P}_A$, $d \in \mathcal{N}_A$,
- $a^i = 0 \vee d^i = 0$, $a, d \in \mathcal{N}_A$,

A5. **Global Constraints**: $C_j[x/x^i]$, for $0 \leq i \leq N$, $C_j \in \mathcal{C}$.
A6. **Goal**: $G[x/x^N]$

Action interference is established according to PDDL 2.1 (Fox and Long 2003), but extended to account for global constraints as follows:

Definition 2 (Interference via Global Constraints)**.** *We say that actions a and b interfere via global constraint C if there are effects e_a and e_b s.t. $lhs(e_a)$ and $lhs(e_b)$ both appearing in C.*

We acknowledge that weaker – yet sound – notions of action interference may exist, but the present ones suffice to guarantee the validity of plans, and are sufficient to enable the execution of several actions in the same plan step whilst guaranteeing the existence of a valid linear plan as per ∀–*step* semantics (Rintanen, Heljanko, and Niemelä 2006).

Rolling up the Repeated Execution of an Action into a Single Plan Step

A seldom discussed consequence of extending STRIPS planning to encompass numeric effects, or conditional effects (Pednault 1986), is that actions are no longer idempotent operations. That is, when doing an action twice in a row, the effects of the action can be different the second time. These *state dependent* effects make numeric planning more involved than STRIPS, yet allow for very compact grounded action descriptions. In this Section we present a novel technique to encode sub–optimal linear plans, that acknowledges this feature and exploits it by allowing to *roll up* many instances of a given action with state dependent *numeric* effects so they are executed in a *single* plan step, subject to constraints that guarantee the validity of the actual plan they entail. We next formalise the two conditions that enable an action to be given this treatment, and then we deal with the difficulties presented by maintaining the truth of preconditions and global constraints through the repeated execution of such actions.

We use $ind(x) = \{y \in X^d \mid c \in C, x, y \in \chi(c)\}$ to denote the set of variables indirectly affected by changing x. A first condition for an action to be rolled up is that its effects are not self-interfering. More formally:

Definition 3 (Self-interfering numeric effects)**.** *An action a is said to feature self-interfering numeric effects whenever at least one of the following conditions hold:*

1. *for some effect $e \in eff_a$ there is an effect $e' \in eff_a$, $e \neq e'$ s.t. $(lhs(e') \cup ind(lhs(e'))) \cap rhs(e) \neq \emptyset$*
2. *there is an effect $e \in eff_a$ s.t. $rhs(e) \cap ind(lhs(e)) \neq \emptyset$*

An action b having non-interfering effects has the property that the value of an affected numeric fluent after m executions of b can be expressed in closed form, and therefore eligible to have multiple executions rolled up.

Theorem 1. *Let b be an action with no self-interfering effects, and a numeric effect on y of the form $\alpha y + \sum_i w_i x_i + k$. The value of fluent y after executing m times action b is given by the equation*

$$f_y^a(m) = \alpha^m y + \sum_{r=0}^{m} \alpha^r (\sum_i w_i x_i + k) \qquad (1)$$

where when $\alpha > 0$ the function $f_y^b(m)$ is monotonic in m.

Proof sketch. By induction over m, observe that it suffices to substitute recursively y by the expression of the effect, to obtain Equation 1 □

In order to guarantee the generation of valid plans when allowing an action b to be rolled up, we need first to ensure that pre_b is satisfied throughout every execution. This is achieved by introducing a set of axioms that are *derived automatically* when b is found to satisfy the eligibility criteria given above. These are obtained by extending the numeric regression operator presented in a recent work (Scala 2013) to account for the rolling up of actions and compiling it into our theory $T(\Pi, N)$. Given an action b with monotonic effects affecting the set of fluents Y, and given a condition $c \in pre_b$ s.t. $\chi(c) = X \cup Y$, the regression of c through m repetitions of action b is the following:

$$c^{r(b,m)} \equiv \sum_{y \in Y} w_y f_y^b(m) + \sum_{x \in X} w_x x + k \{\geq, >, =\} 0 \qquad (2)$$

where w_y, w_x are the coefficients associated with fluents x and y in c. Equation 2 amounts to rewriting the precondition c so as to account for the effect of each repeated application of b over each fluent in c. The second condition an action b needs to comply with in order to admit rolling up execution is that the left-hand-side of $c^{r(b,m)}$ is a *monotone* function over m, and the sufficient conditions for that to be the case follow

Proposition 2. *$c^{r(b,m)}$ in Equation 2 is a* monotone *function if at least one of the three following conditions holds for b:*

1. *$\forall y \in Y$, $f_y^b(m)$ is* linear
2. *$|Y| = 1$ and $\alpha > 0$.*
3. *$\forall y \in Y$, $f_y^b(m)$ is* monotonically increasing (decreasing).

At this point in the discussion, we find it necessary to make some important remarks. First, whenever $0 < \alpha < 1$ or $\alpha > 1$, $f_y^b(m)$ becomes an *exponential* function. In this case, satisfiability of $T(\Pi, N)$ is only decidable when approximated within a tolerance parameter δ (Gao, Avigad, and Clarke 2012), and requires specialised solvers. Alternatively, the SMT framework can be abandoned altogether, adopting CP or Mixed-Integer Non-linear Programming (MINLP) languages and solvers instead. Second, when $\alpha < 0$, $f_y^b(m)$ oscillates and becomes non-monotonic. In that case action rolling up can only be applicable after reformulating the action model, breaking up action b into actions d_k, so that $f_y^b(m)$ is approximated piecewise by monotone functions $f_y^{d_k}(m)$. Last, we note that cases $\alpha = 1$ and $\alpha = 0$ cover a very general class of dynamics, where rates of change are described by linear or constant equations.

Definition 4 (Rolling up eligibility)**.** *An action b without self-interfering effects is said to be* eligible for rolling up *whenever it complies with Proposition 2, and for each propositional precondition $x = k$ in $pre(b)$, $x \notin aff(b)$.*

Once established all of the above for action b, the following Proposition formalises the guarantee that rolling up b *necessarily* results in valid plans

Proposition 3. *Let b be an action eligible for rolling up, and let s' be a state such that $s' \models y = f_y^b(m)$ for some $m > 0$. If for all $c \in pre(b)$, $c^{r(b,m-1)}$ and pre_b are satisfied in s, then the transition between states s and s' is valid.*

Proof sketch. Let $[lhs(c^{r(b,0)})]^s \geq 0$ and $[lhs(c^{r(b,k)})]^s \geq 0$. For the sake of the argument, assume that there exists i, $0 < i < k$ s.t. $[lhs(c^{r(b,i)})]^s < 0$. This can only be true whenever the right hand side of Equation 2 is *not* a monotone function, which is in turn only possible when b does not comply with Proposition 2. □

In the absence of global constraints, satisfiability of the following axioms suffices to enforce Proposition 3:
A7. **Rolled-up Preconditions and Effects**

$$b^i > 1 \Rightarrow \bigwedge_{c \in pre_b} c^{r(b,b^i-1)}[x/x^i] \wedge pre_b[x/x^i] \quad (3)$$

$$b^i > 1 \Rightarrow \bigwedge_{\langle y, \sum_i w_i x_i + k \rangle \in eff_b} y^{i+1} = f_y^b(b^i)[x/x^i] \quad (4)$$

$$b^i > 1 \Rightarrow \bigwedge_{\langle y, v \rangle \in eff_b} y^{i+1} = v \quad (5)$$

$$b^i \leq 1, \forall b \text{ not eligible for rolling up} \quad (6)$$

These axioms ensure that the accumulated effects do not violate c until (possibly) right after the last execution is done. Equation 5 ensures the propagation of *state independent* effects.

Handling Global Constraints

In the presence of global constraints $\mathcal{C}$, the validity of plans when rolling an action a depends on each intermediate state s' satisfying every $C_i \in \mathcal{C}$, or in other words, each transition made by the sequence of actions being rolled up needs to be *valid*. For global constraints C_i involving a single condition, handling global constraints is a straightforward variation on axiom 7 introduced in the previous Section:
A8. **Rolled-up actions and global conjunctive constraints**

$$b^i > 1 \Rightarrow C^{r(b,b^i)}[x/x^i],\ C \in \mathcal{C} \quad (7)$$

where we retain the regression of b and drop the term involving the precondition.

Handling properly the case when $C_i = c_1 \vee \ldots \vee c_n$ is not trivial, since every disjunct c_j can be potentially relevant to an action b, and this relevance depends on the number of instances of the action to be rolled up. In Figure 2 we can see that for the sequence β^1,...,β^8 one disjunct is true (below line c_3) all along the sequence. This is the case when global constraints and action effects are known to be aligned with the axis – as is the case in Figure 2. When constraints are oriented along an arbitrary rotation axis or actions allow changes in the state that are not parallel or orthogonal to that rotation axis, it gets complicated. For instance, when we try to roll up α^1, α^2 and α^3 together, no constraint c_j remains true in every intermediate state, precluding rolling up

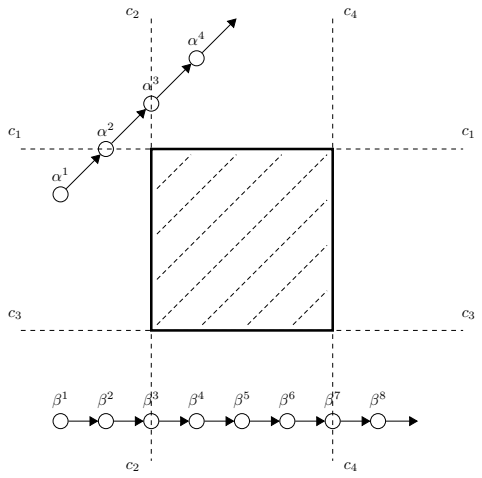

Figure 2: Rectangle in the center represents an obstacle as modeled in GEOMETRIC ROVERS: c_0,...,c_3 are the four disjuncts in the constraint disallowing actions to end inside the obstacle. Two executable sequences of instances of actions α and β, namely, α^1,..,α^4 and β^1,...,β^8 are shown to illustrate how actions interact with disjuncts c_j. See text for discussion.

together those three actions. On the other hand, c_1 is always true for the intermediate states generated by the sequence α^2,α^3,α^4, and rolling up becomes possible.

In order to ensure the consistency of the implicit trajectory conveyed by a sequence of rolled up actions w.r.t. the global constraints, each disjunct c_j in a global constraint C_i is substituted with a conjunction, each conjunct asserting the truth of c_j *before* and *after* the sequence is executed:

$$\mathcal{R}_{C_i}^{b,m} = (c_1 \wedge c_1^{r(b,m)}) \vee \ldots \vee (c_n \wedge c_n^{r(b,m)})$$

This results in a *new*, *automatically generated* constraint $\mathcal{R}_{C_i}^{b,m}$ for each action b and disjunctive constraint C_i. This constraint implicitly creates the commitment that the cumulative effects of the rolled up action, given by Equation 2, need to be consistent with at least one disjunct of constraint C_i. The last set of axioms to be added to our theory $T(\Pi, N)$ follow directly from Theorem 3:
A9. **Rolled-up actions and global disjunctive constraints**

$$b^i > 1 \Rightarrow \mathcal{R}_{C_k}^{b,b^i}[x/x^i],\ C_k \in \mathcal{C} \quad (8)$$

The above guarantees that whenever an action is rolled up at plan step t, the sequence of actions is *sound*, that is, can be executed. But it does not capture *all possible* ways to roll up actions. Going back to the example in Figure 2, there is no reason in principle for not rolling up α^1,...,α^4. Yet the reformulated constraint

$$(c_1 \wedge c_1^{r(\alpha,4)}) \vee \ldots \vee (c_4 \wedge c_4^{r(\alpha,4)})$$

is false, as well as any instance of axiom A9, for $b = \alpha$ and $m = 4$. If the sequence α^1,...,α^4 is part of the valid plan selected by the solver, we will have α^1 at plan step t, and the rolled up sequence α^2,...,α^4 at plan step $t+1$, or alternatively, α^1,α^2 in plan step t, and α^3,α^4 in $t+1$. Either way, we will end up doing an additional call to the solver.

Empirical Evaluation

In order to gauge the efficiency of the encoding presented in the previous sections, we have build the planner SPRINGROLL, implemented in JAVA, that generates $T(\Pi, N)$ automatically from a PDDL 2.1 description of Π, trivially extending the language to represent global constraints, and invokes the SMT solver Z3 (de Moura and Bjorner 2008) in an off-the-shelf fashion. The algorithm to search for plans is that of Kautz and Selman (1999). Namely, we generate theories $T(\Pi, k)$, $k = 0,1,2,\ldots$, until $T(\Pi, k)$ is SAT. We acknowledge existing work that speeds up notably planning times by either tweaking solver variable selection heuristics and clause learning schemes, using smart strategies for the search of planning horizon k for which $T(\Pi, k)$ models a plan (Rintanen 2012), as well as aiming at the derivation of sets of clauses that implement *deductive lower bounds* (Geffner 2004) from the analysis of unsatisfiability proofs (Suda 2014). But we leave the investigation of which heuristics and strategies work best with the kind of axioms we are generating to future work, and focus instead on measuring the *inherent* efficiency of the encoding and our choice in the representation of derived fluents and compare it with state–of–the–art heuristic search planners, when such a system is available.

In order to evaluate SPRINGROLL we have tested it over a diverse set of benchmarks, discussed next, on an 8–core i7-4770@3.40Ghz machine using Ubuntu 14.04. We limited run–time of planners to 500 seconds and memory usage to 2 GBytes, unless otherwise noted.

Benchmarks

The first set of benchmarks we consider are those of Ivankovic et al. (2014). These allow us to test the effectiveness of SPRINGROLL when dealing with global numeric constraints that involve both basic and derived variables. The Power Supply Restoration (PSR) domain, where the number of global constraints affected by action execution depends on the state, is taken off–the–shelf. In Hydraulic Blocks World (HBW) some constraints have been directly encoded as action effects (e.g., the weight of a box on a piston). Equilibrium of forces among the pistons has been modeled using global constraints in a way that is equivalent to the *switched constraints* discussed by Ivankovic et al. (2014).

The second set of benchmarks is taken from Francès and Geffner (2015). This includes a number of domains that take numeric planning out of the comfort zone by introducing challenging numeric preconditions and goals, as well as state constraints (Ferrer-Mestres, Frances, and Geffner 2015). In the domains GARDENING, PUSHING STONES and GROUPING we have replaced the graph representation of the navigation grid with locations represented by tuples $(x, y) \in \mathbb{N}^2$. This alternative formulation does not change the set of valid and optimal plans. We also generated additional larger instances, increasing the value of the relevant parameters offered by Francès and Geffner's (2015) problem generators. Along with these we also considered the domains from the numeric track of the 3rd International Planning Competition (Long and Fox 2003) minus the SETTLERS domain[2] For the ZENO TRAVEL domain we generated a set of instances with increasingly tighter bounds on the amount of fuel allowed to be spent, resulting in a set of numerically challenging instances.

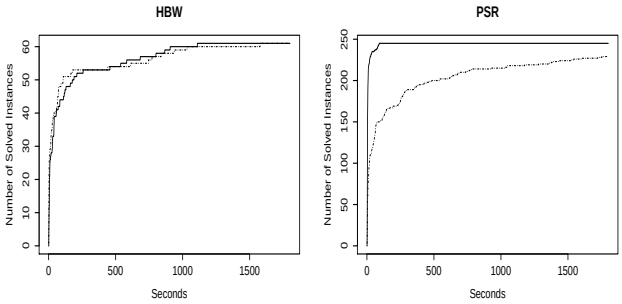

Figure 3: Coverage vs Timeout (secs) curve for SPRINGROLL (continuous line) and HSHP (dashed line) over HBW (on the left) and PSR (on the right).

The third and last set consist of hand-crafted instances of GEOMETRIC ROVERS with increasing numbers of obstacles (up to 20) and tasks to be performed (up to 5). Besides that we generated 20 instances without tasks and unlimited battery charge, with the number of obstacles ranging between 5 and 100. In the instances with more than 50 obstacles the percentage of map area covered by them is greater than 70%. Such problems are considered to be challenging by recent works on motion planning (Plaku 2013).

In all these benchmarks the value of α in Equation 1 is either 0 or 1.

Results

Ivankovic et al.'s (2014) domains. In Figure 3 the coverage of SPRINGROLL is compared with that of the heuristic search hybrid[3] planner (HSHP) proposed by Ivankovic et al. (2014). For both domains, SPRINGROLL was run allowing only one action per time step so as to find optimal plans, as it is the case for HSHP. On HBW both planners solved 61 out of 72 solvable instances (where plans are up to 20 actions), and spent similar average run-times: 135 secs for SPRINGROLL and 137 secs for HSHP. On the PSR instances, a substantial gap separates SPRINGROLL from HSHP. SPRINGROLL coverage is greater (245 vs 229), solving all instances in under 100 seconds, much faster than HSHP. The latter is on average 54 times slower than SPRINGROLL. The HSHP heuristic is very expensive to compute in general, and on the larger instances of the benchmark, it is not very accurate evaluating 10-100 times more nodes than are expanded. Plans are very short, and even without rolling up actions, SPRINGROLL is very efficient.

Francès and Geffner's (2015) domains. The comparison between SPRINGROLL, the latest version available of FS0[4]

[2]This domain required non-trivial reformulation of its instances, as it relies on the notion of *undefined* numeric fluents to account for objects being dynamically created.

[3]Since it deals with mixed discrete–continuous state variables.

[4]https://bitbucket.org/gfrances/fs0

		Coverage			Plan length			Average Time		
	I	Springroll	FS0	FF	Springroll	FS0	FF	Springroll	FS0	FF
GARDENING	51	24	41	**51**	**89.9**	91.6	266.4	83.2	43.17	**1**
PUSHING STONES	324	**87**	59	NA	**42**	70.5	NA	117.3	**113**	NA
GROUPING	192	**192**	124	22	43.5	**32.8**	41	**0.1**	43.7	21.8
COUNTERS	35	**35**	12	9	130.4	**118.4**	**118.4**	**0.1**	12.6	142.5
COUNTERS-RND	105	**105**	36	30	209.9	**180.6**	192.7	**0.1**	50	10.4
COUNTERS-INV	35	**35**	12	6	69.3	**61**	64.3	**0.1**	6.6	30.4

Table 1: Coverage, average plan length and time (in seconds) of FS0 SPRINGROLL and MetricFF on the domains reported by (Francès and Geffner 2015).

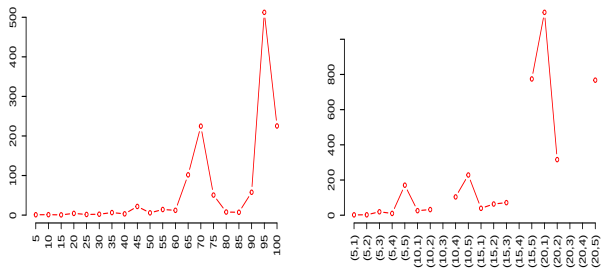

Figure 4: Run-times of GEOMETRIC ROVERS with tasks and limited battery (right) and without (left). The y-axis shows run-time in seconds, the x-axis is the number of obstacles (or obstacles and tasks, on the right). Missing data points denote a time out.

implementing state constraints, and MetricFF (Hoffmann 2003) is reported in Table 1. With the exception of PUSHING STONES and GROUPING all planners use the same formulation of the given domain[5]. MetricFF is not tested on PUSHING STONES since it does not support state constraints. We tested SPRINGROLL without generating axioms 7–9, therefore disabling action roll up entirely. The coverage of the planner resulting from using axioms 1–6 alone is an insignificant fraction of that attained by SPRINGROLL with rolling up and the other planners. For instance, SPRINGROLL generating only axioms 1–6 solves just 15 instances, out of 170, over all COUNTERS variants.

SPRINGROLL does best on the GROUPING and COUNTERS domains. In these, all actions are eligible for being rolled up, and on top of that, actions seldom interfere with one another. Interestingly, rolling up actions tends to produce longer plans than FS0 on these two domains. Since there are no explicit upper bounds on the values that functions b^i can take, the plans SPRINGROLL obtains in COUNTERS set the counters to arbitrarily high or small values achieving the same goal condition with more actions than FS0 plans do. On the domain PUSHING STONES – a simplified version of SOKOBAN with no fixed obstacles – SPRINGROLL's coverage is 50% higher than FS0's. Actions interfere often and serialise plans, yet this is balanced by rolling up of actions, that keeps the planning horizon manageable for a substantial number of instances. In GARDENING FS0 and MetricFF coverage doubles that of SPRINGROLL, and MetricFF solves all the instances. Solving GARDENING requires finding plans with length linear on the number of plants to be watered, and actions doing so interfere with every other action. As in PUSHING STONES this implicitly serialises plans and increases the planning horizon for which valid plans exist, yet in this case rolling up actions does not allow SPRINGROLL to keep the pace with the heuristic search planners.

Long and Fox's (2003) benchmarks. We compared MetricFF with SPRINGROLL over the 3rd IPC benchmarks. Across all the tested domains, MetricFF clearly performs better than SPRINGROLL solving 76 out of 125 total instances while SPRINGROLL only manages to solve 28 instances. A notable exception is for the ROVERS domain, where both MetricFF and SPRINGROLL solve 12 out of 20 instances. Here, MetricFF Enforced Hill Climbing did not perform particularly well due to numerous numeric deadends, feature that does not affect SPRINGROLL. For this class of benchmarks, results are in general consistent with the experimental results reported by previous works (Bofill, Espasa, and Villaret 2015; Hoffman et al. 2007). The instances in the benchmarks do not have very tightly constrained goals, sometimes making the numeric structure almost irrelevant and most action effects are propositional, ruling out action roll ups. In order to verify the former observation, we took the instances from the ZENO TRAVEL domains and introduced a hard constraint on the amount of fuel that valid plans could spend. With this constraint set so that plans used 90% of the fuel used by MetricFF plans on the original instances, MetricFF solves half as many problems as it did, 10 out of 23, while SPRINGROLL solves 6 out of 23.

GEOMETRIC ROVERS. Our experiments on the GEOMETRIC ROVERS benchmarks are reported in Figure 4. Run-times increase with the number of obstacles, all of them aligned with the axis, and tasks (see graph on the right of Figure 4). The longest plans in these instances consist of up to 370 actions, a number significantly higher than that of plans typically feasible for SAT or SMT planners. When limiting run-times to $1,800$ seconds, SPRINGROLL solves all the instances (20) without tasks and unlimited battery, and 16 out of 20 of the instances with tasks and limited battery. Similarly as for the GARDENING domain discussed above, actions achieving tasks interfere with moving and recharging, so the planning horizon increases with the number of tasks to be performed (and SPRINGROLL's performance decreases accordingly).

We also tested the planner on benchmarks where obstacles were oriented around an arbitrary rotation axis. The time allowed to the planner in this case was $3,600$ seconds. With these settings, SPRINGROLL solves problems with as many

[5]In those domains, the inequality $loc(o) \neq loc(o')$ used to denote compactly that o and o' should occupy different locations, becomes $x(o) \neq x(o') \vee y(o) \neq y(o')$. At the moment of writing this, FS0 does not support the $\vee$ logical connective and only *binary* relations over pairs of fluents can be defined intensionally.

as 5 obstacles in *tens of seconds*, that is one order of magnitude slower than when obstacles are all aligned with the axis. The biggest instance solved features 15 obstacles, in slightly over $2,000$ seconds. This inability to scale up has two direct causes. First, as discussed in the previous Section, our axioms do not guarantee that SPRINGROLL rolls up as many instances of the same action together as it would be possible, and we have to deal with more *unsatisfiable* intermediate theories $T(\Pi, i)$. Second, when obstacles are arbitrarily oriented, $\mathcal{R}^{b,m}_{C_i}$ involves both state variables x and y and the resulting multivariate disjuncts hurt further solver performance. We conjecture that more decisions are required to propagate implied atoms, i.e. a disjunct of $\mathcal{R}^{b,m}_{C_i}$ being true or false.

Related Work

Compilations of planning over states with numeric variables into SMT have been reported in the past (Wolfman and Weld 1999; Shin and Davis 2005; Hoffman et al. 2007; Bofill, Espasa, and Villaret 2015), yet have not proved so far a very popular approach, in part due to the reported inefficiency of early SMT solvers (Hoffman et al. 2007). We demonstrate that with an appropriate encoding that exploits the expressive power of SMT and looks at the fine structure of the domain dynamics as modeled by action effects, planning as SMT can be competitive with state–of–the–art heuristic search planners.

The way we handle the interaction of preconditions and global constraints with action rolling up is related to the *zero-crossing axioms* discussed by Shin and Davis (2005). TM-LPSAT encodes into a set of constraints the necessary conditions requiring to insert a new time point, namely, that a given constraint C_i's truth value changes between two given time points the planner has already committed to. In this work, axioms A7–A9 describe sufficient conditions for which execution of actions can proceed in a *continuous* way without interfering with preconditions or global constraints. In other words, TM-LPSAT looks at when the derivative of $C_i(t)$, the function describing the values of constraint C_i left–hand side change over time, is zero. Our axioms allow rolling up whenever actions changes on variables relevant to C_i are such that the derivative of $C_i(t)$ is always positive or negative, over an interval defined as an arbitrary number m of discrete time steps of duration ∂t.

To the best of our knowledge, the only other planner handling unrestricted numeric effects in actions *and* global constraints is KONGMING (Li 2011). The KONGMING plan graph and its encoding as a MINLP is very similar to our theories $T(\Pi, N)$, yet not equivalent in the same way as Kautz & Selman encoding of planning tasks into SAT is equivalent to GRAPHPLAN (Geffner 2004). Their approach to the problem of ensuring that continuing, repeating change complies with global constraints is by constructing over–approximations of valid plans, or *flow tubes* (Hofmann and Williams 2006). Each of these define implicitly many trajectories which are known to be consistent with the global constraints, in roughly the same way as our axioms A8 and A9 do. Flow tubes as defined in KONGMING over-approximate the set of valid trajectories, while our axioms represent a subset of the possible feasible trajectories within a given plan step.

Last, our encoding aims at producing more succinct encodings of planning tasks, as is the case for $\exists$–step semantics (Rintanen, Heljanko, and Niemelä 2006). While the former exploits both the lack or weakness of causal dependencies between actions in a valid plan, we exploit the observation that actions with numeric state–dependent effects are not idempotent in general. In both works though, this is achieved by adding additional constraints, challenging theory solvers in interesting ways (Rintanen 2012).

Discussion & Future Work

This paper presents a novel encoding of expressive numeric planning with disjunctive global constraints that, under some well-defined conditions, produces very compact theories $T(\Pi, N)$ modeling valid plans. We instantiate the approach for the SMT modeling and computational framework and show that it can compete with the state–of–the–art in non–trivial interesting numeric planning problems, *without further optimisations*.

While numeric arguments in action schemata could seem as an alternative to rolling up, we rather see that the contributions in this paper would be solving the problems posed by selecting values for such arguments and implementing the transition function for such actions when effects interact with preconditions and global constraints. Handling global disjunctive constraints is crucial for the GEOMETRIC ROVERS domain, and rolling up the execution of actions in the same plan step reduces the horizon N at which $T(\Pi, N)$ models a valid plan by an order of magnitude (e.g. a plan with over 300 actions requires horizon $N \approx 10$). GEOMETRIC ROVERS is just a concrete example of planning tasks that require to find a *path* between two arbitrary points in the n-manifold resulting from the intersection of an arbitrary number of *open* n-manifolds representing the points that satisfy global constraints. Our work assumes that such paths (plans) can be described in a piecewise form by concatenating monotone functions, i.e. the accumulated effects of rolled up actions.

As immediate future work, besides investigating the impact of recent orthogonal optimisations and other heuristic techniques in the efficiency of SMT solvers on the theories we generate, we aim at developing better heuristic estimators (Löhr et al. 2012) for hybrid planning that exploit the ability to roll up many discrete changes into a plan step.

Acknowledgments

This research is supported by the ARC Discovery Project "Robust AI Planning for Hybrid Systems" (DP140104219) and NICTA. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. We would also like to thank the anonymous reviewers and Alban Grastien for their constructive and helpful comments.

References

Barrett, C.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2008. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press. 737–797.

Bofill, M.; Espasa, J.; and Villaret, M. 2015. The RANTANPLAN planner: System description. In *Proc. of Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*, 1–10.

Bonet, B., and Geffner, H. 2014. Belief tracking for planning with sensing: Width, complexity and approximations. *Journal of Artificial Intelligence Research* 50:923–970.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proc. of ICAPS*, 42–49.

de Moura, L., and Bjorner, N. 2008. Z3: An efficient SMT solver. *Lecture Notes in Computer Science* 4963:337–340.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2012. Semantic attachments for domain-independent planning systems. In *Towards Service Robots for Everyday Environments*. 99–115.

Edelkamp, S., and Hoffmann, J. 2004. Pddl 2.2: The language for the classical part of the 4th international planning competition. Technical report, Albert-Ludwigs-Universität Freiburg, Institüt for Informatik.

Ferrer-Mestres, J.; Frances, G.; and Geffner, H. 2015. Planning with state constraints and its application to combined task and motion planning. In *Proc. of Workshop on Planning and Robotics (PLANROB)*, 13–22.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Fox, M.; Long, D.; and Magazzeni, D. 2011. Automatic construction of efficient multiple battery usage policies. In *Proc. of IJCAI*, 2620–2625.

Francès, G., and Geffner, H. 2015. Modeling and computation in planning: Better heuristics from more expressive languages. In *Proc. of ICAPS*, 70–78.

Gao, S.; Avigad, J.; and Clarke, E. M. 2012. δ-complete decision procedures for satisfiability over the reals. In *Automated Reasoning*. Springer. 286–300.

Geffner, H. 2004. Planning Graphs and Knowledge Compilation. In *Proc. of Principles of Knowledge Representation and Reasoning (KR)*.

Hoffman, J.; Gomes, C.; Selman, B.; and Kautz, H. 2007. SAT encodings of state-space reachability problems in numeric domains. In *Proc. of IJCAI*, 1918–1923.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.

Hofmann, A., and Williams, B. C. 2006. Robust execution of temporally flexible plans for bipedal walking devices. In *Proc. of ICAPS*, 386–389.

Ivankovic, F.; Haslum, P.; Thiébaux, S.; Shivashankar, V.; and Nau, D. S. 2014. Optimal planning with global numerical constraints. In *Proc. of ICAPS*, 145–153.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and Graph-based planning. In Dean, T., ed., *Proc. of IJCAI*, 318–327. Morgan Kaufmann.

Li, H. X. 2011. *Kongming: A Generative Planner for Hybrid Systems with Temporally Extended Goals*. Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Massachussets Institute of Technology.

Löhr, J.; Eyerich, P.; Keller, T.; and Nebel, B. 2012. A planning based framework for controlling hybrid systems. In *Proc. of ICAPS*, 164–171.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 1–59.

Lopez, A., and Bacchus, F. 2003. Generalizing graphplan by formulating planning as a csp. In *Proc. of IJCAI*, 954–960.

Muise, C.; McIlraith, S. A.; and Beck, J. C. 2012. Improved Non-deterministic Planning by Exploiting State Relevance. In *Proc. of ICAPS*.

Pednault, E. P. D. 1986. Formulating multiagent, dynamic-world problems in the classical planning framework. In *Reasoning about actions and plans*. 47–82.

Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2013. Combining a temporal planner with an external solver for the power balancing problem in an electricity network. In *Proc. of ICAPS*.

Plaku, E. 2013. Robot motion planning with dynamics as hybrid search. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1415–1421.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence Journal* 170(12-13):1031–1080.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artificial Intelligence Journal* 193:45–86.

Scala, E. 2013. Numeric kernel for reasoning about plans involving numeric fluents. In *AI*IA 2013: Advances in Artificial Intelligence*, 263–275.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence Journal* 166(1):194–253.

Smith, D. E.; Frank, J.; and Cushing, W. 2008. The ANML language. In *The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Suda, M. 2014. Property directed reachability for automated lanning. *Journal of Artificial Intelligence Research* 50(1):265–319.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In defense of PDDL axioms. *Artificial Intelligence Journal* 168(1-2):38–69.

Wolfman, S. A., and Weld, D. S. 1999. The lpsat engine & its application to resource planning. In *Proc. of IJCAI*, 310–317.