Complexity Issues of Interval Relaxed Numeric Planning

Johannes Aldinger and **Robert Mattmüller** and **Moritz Göbelbecker**
Albert-Ludwigs-Universität, Institut für Informatik
79110 Freiburg, Germany
{aldinger,mattmuel,goebelbe}@informatik.uni-freiburg.de

Abstract

Automated planning is a hard problem even in its most basic form as STRIPS planning. We are interested in numeric planning tasks with instantaneous actions, a problem which is not even decidable in general. Relaxation is an approach to simplifying complex problems in order to obtain guidance in the original problem. We present a relaxation approach with intervals for numeric planning and discuss the arising complexity issues.

Introduction

Relaxation is a predominant approach to simplifying planning problems. Solutions of the relaxed planning problem can be used to guide search in the original planning task. The forward propagation heuristic h_{add} (Bonet, Loerincs, and Geffner 1997; Bonet and Geffner 2001) was used in the heuristic search planner that won the first International Planning Competition (IPC 1998) and h_{max} (Bonet and Geffner 1999) is its admissible counterpart. The underlying assumption of a delete relaxation is that propositions which are achieved once during planning can not be invalidated. More recent planning systems are usually not restricted to propositional state variables of the planning problem. Instead they use the SAS^+ formalism (Bäckström and Nebel 1993) which allows for (finite-domain) multi-valued variables. Unlike propositional STRIPS (Fikes and Nilsson 1971), a "delete relaxation" corresponds to variables that can attain a *set* of values at the same time. Extending this concept for numeric planning relaxes the set representation even further. Numeric variables can have infinitely many values which makes it impossible to store all of them. A memory efficient approach is to consider the enclosing interval of all possible values for each numeric variable. The methods to deal with intervals have been subject to the field of interval arithmetic (Young 1931) which has been used in mathematics for decades (Moore, Kearfott, and Cloud 2009) and enables us to deal with intervals in terms of basic numeric operations.

Numeric planning tasks can require actions to be applied multiple times, as setting a numeric variable to a target value can require multiple steps even in relaxed problems. In this paper we provide the foundations for interval relaxed numeric planning.

Related Work

Extending the concept of classical planning heuristics to numeric problems has been done before, albeit only for a subset of numeric tasks. In many relevant real world problems, numeric variables can only be manipulated in a restricted way. The Metric-FF planning system (Hoffmann 2003) tries to convert the planning task into a *linear numeric* task which ensures that variables can "grow" in only one direction. By introducing inverted auxiliary variables for decreasing numeric variables, the concept of a delete relaxation translates into a relaxation where *decrease* effects are considered harmful and higher values of a variable are always beneficial to fulfill the preconditions of actions.

More recently, Coles et al. (2008) investigated an approach based on linear programs. In many relevant real world applications, numeric variables are used to model resources. Delete relaxation heuristics fail to offer guidance on such problems if a *cyclic resource transfer* is possible. As delete relaxations make the assumption that sub-goals stay achieved, a resource transfer can "produce" resources without decreasing them at their original destination. Coles et al. analyze the planning problem for consumers and producers of resources and build a linear program to ensure that resources are not more often consumed than produced or initially available to obtain an informative heuristic.

Basics

In this section we outline numeric planning with instantaneous actions which is expressible in PDDL2.1, layer 2 (Fox and Long 2003). We present an overview over interval arithmetic, the technique we use to extend delete relaxation heuristics to numeric planning. The section closes with a short complexity discussion.

Numeric Planning with Instantaneous Actions

Given a set of variables $\mathcal{V}$ with domains $\text{dom}(v)$ for all $v \in \mathcal{V}$, a *state* s is a mapping of variables v to their respective domains. Throughout the paper, we denote the value of a variable v in a state s by $s(v)$.

A numeric planning task $\Pi = \langle \mathcal{V}_P, \mathcal{V}_N, \mathcal{O}, \mathcal{I}, \mathcal{G} \rangle$ is a 5-tuple where $\mathcal{V}_P$ is a set of propositional variables $\mathsf{v_p}$ with domain $\{$*true*,*false*$\}$. $\mathcal{V}_N$ is a set of numeric variables $\mathsf{v_n}$ with domain $\mathbb{Q}^\infty$ where we abbreviate $\mathbb{Q}^\infty$ for

$\mathbb{Q} \cup \{-\infty, \infty\}$ throughout the paper. $\mathcal{O}$ is a set of operators, $\mathcal{I}$ the initial state and $\mathcal{G}$ is the goal condition. A *numeric expression* $\mathsf{e}_1 \circ \mathsf{e}_2$ is an arithmetic expression with operators $\circ \in \{+, -, \times, \div\}$ and expressions e_1 and e_2 recursively defined over variables $\mathcal{V}_N$ and constants from $\mathbb{Q}$. A *numeric constraint* $\mathrm{con} = (\mathsf{e}_1 \bowtie \mathsf{e}_2)$ compares numeric expressions e_1 and e_2 with $\bowtie \in \{\leq, <, =, \neq\}$. A *condition* is a conjunction of propositions and numeric constraints. A *numeric effect* is a triple $(\mathsf{v_n} \circ\!\!= \mathsf{e})$ where $\mathsf{v_n} \in \mathcal{V}_N$, $\circ\!\!= \in \{:=, +\!\!=, -\!\!=, \times\!\!=, \div\!\!=\}$ and e is a numeric expression. Operators $o \in \mathcal{O}$ are of the form $\langle \mathrm{pre} \rightarrow \mathrm{eff} \rangle$ and consist of a condition pre and a set of effects eff which contains at most one numeric effect for each numeric variable $\mathsf{v_n}$ and at most one truth assignment for each propositional variable $\mathsf{v_p}$.

The semantic of a numeric planning task is straightforward. For constants $c \in \mathbb{Q}, s(c) = c$ by abuse of notation. Numeric expressions $(\mathsf{e}_1 \circ \mathsf{e}_2)$ for $\circ \in \{+, -, \times, \div\}$ are recursively evaluated in state s: $s(\mathsf{e}_1 \circ \mathsf{e}_2) = s(\mathsf{e}_1) \circ s(\mathsf{e}_2)$. A state satisfies a condition $s \models \mathsf{v_p}$ iff $s(\mathsf{v_p}) = \mathit{true}$, where $\mathsf{v_p} \in \mathcal{V}_P$. For numeric constraints, $s \models (\mathsf{e}_1 \bowtie \mathsf{e}_2)$ iff $s(\mathsf{e}_1) \bowtie s(\mathsf{e}_2)$ where $\bowtie \in \{\leq, <, =, \neq\}$, and e_1 and e_2 are expressions. A state satisfies a conjunctive condition $s \models k_1 \wedge k_2$ iff $s \models k_1$ and $s \models k_2$.

An operator $o = \langle \mathrm{pre} \rightarrow \mathrm{eff} \rangle$ is applicable in s iff $s \models \mathrm{pre}$. The successor state $\mathrm{app}_o(s) = s'$ resulting from an application of o is defined as follows, where $\mathrm{eff} = \{\mathrm{eff}_1, \ldots, \mathrm{eff}_n\}$: if eff_i is a numeric effect $\mathsf{v_n} \circ\!\!= \mathsf{e}$ with $\circ\!\!= \in \{+\!\!=, -\!\!=, \times\!\!=, \div\!\!=\}$, $s'(\mathsf{v_n}) = s(\mathsf{v_n}) \circ s(\mathsf{e})$. If eff_i is a numeric effect $\mathsf{v_n} := \mathsf{e}$, then $s'(\mathsf{v_n}) = s(\mathsf{e})$. If eff_i is a propositional effect $\mathsf{v_p} := \mathsf{e_p}$ with $\mathsf{e_p} \in \{\mathit{true}, \mathit{false}\}$, $s'(\mathsf{v_p})$ is the new truth value $\mathsf{e_p}$. Finally, if a variable v does not occur in any effect, then $s'(\mathsf{v}) = s(\mathsf{v})$.

A plan π is a sequence of actions that leads from $\mathcal{I}$ to a state satisfying $\mathcal{G}$ such that each action is applicable in the state that follows by executing the plan up to that action.

We intend to relax numeric planning with the help of intervals. The next section establishes the foundations of interval arithmetic.

Interval Arithmetic

Interval arithmetic uses an upper and a lower bound to enclose the actual value of a number. Closed intervals $[\underline{x}, \overline{x}] = \{q \in \mathbb{Q}^\infty \mid \underline{x} \leq q \leq \overline{x}\}$ contain all rational numbers (or $\pm\infty$) from $\underline{x}$ to $\overline{x}$. Throughout this paper we refer to the lower bound of an interval x by $\underline{x}$ and to the upper bound by $\overline{x}$. The set $\mathbb{I}_c = \{[\underline{x}, \overline{x}] \mid \underline{x} \leq \overline{x}\}$ contains all closed intervals. Numbers q can be transformed into a degenerate interval $[q, q]$. The basic arithmetic operations in interval arithmetic are given as:

- addition: $[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$,
- subtraction: $[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$,
- multiplication: $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] =$ $\left[\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})\right]$,
- division: $[\underline{x}, \overline{x}] \div [\underline{y}, \overline{y}] =$ $\left[\min\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right), \max\left(\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\right)\right]$

if $0 \notin [\underline{y}, \overline{y}]$. Otherwise, at least one of the bounds diverges to $\pm\infty$. We do not explicate all cases of $\underline{x}, \overline{x}, \underline{y}$ and $\overline{y}$ being positive, negative or zero which determine which of the bounds diverge and refer the interested reader to the literature (Moore, Kearfott, and Cloud 2009).

Analogously we define open bounded intervals $(\underline{x}, \overline{x}) = \{q \in \mathbb{Q}^\infty \mid \underline{x} < q < \overline{x}\}$ and the set of open intervals $\mathbb{I}_o = \{(\underline{x}, \overline{x}) \mid \underline{x} < \overline{x}\}$, as well as half open intervals $[\underline{x}, \overline{x}) = \{q \in \mathbb{Q}^\infty \mid \underline{x} \leq q < \overline{x}\}$ and $(\underline{x}, \overline{x}] = \{q \in \mathbb{Q}^\infty \mid \underline{x} < q \leq \overline{x}\}$ and the respective sets $\mathbb{I}_{co} = \{[\underline{x}, \overline{x}) \mid \underline{x} < \overline{x}\}$ and $\mathbb{I}_{oc} = \{(\underline{x}, \overline{x}] \mid \underline{x} < \overline{x}\}$. Finally the set of mixed bounded intervals is given as $\mathbb{I}_m = \mathbb{I}_c \cup \mathbb{I}_o \cup \mathbb{I}_{oc} \cup \mathbb{I}_{co}$. Open and mixed bounded intervals follow the same arithmetic rules as closed intervals. Whenever open and closed bounds contribute to the new interval bound, the bound is open.

Example 1. The product $(-2, 3] \times [-4, 2)$ is the interval $[-12, 8)$. The lower bound is the result of 3×-4 and the resulting bound is closed because both contributing bounds are closed. The new upper bounds is computed by -2×-4 and the open bound of the left interval determines the "openness" of the resulting bound.

Definition 1. Let $x, y \in \mathbb{I}_m$ be intervals. The convex union $u = x \sqcup y$ is the interval with $\underline{u} = \min(\underline{x}, \underline{y})$ and $\overline{u} = \max(\overline{x}, \overline{y})$. Whether the bounds of u are open or closed depends on whether those of x and y are open or closed.

Definition 1 implicitly adds all values between the intervals to the resulting interval if $x \cap y = \emptyset$.

Complexity

Unlike classical planning, which is PSPACE-complete (Bylander 1994), numeric planning is *undecidable* (Helmert 2002). Even though completeness of numeric planning can therefore not be achieved in general, numeric planners can find plans or an assurance that the problem is unsolvable for many practical problems. Moreover, we will prove in the following section that the relaxed numeric plan existence problem is decidable in polynomial time for *acyclic dependency* tasks, tasks in which the expressions of numeric effects do not depend on the variable they alter.

Delete Relaxation

In this section we discuss natural extensions of delete relaxation to planning with numeric variables.

Motivation

As planning is hard, it is beneficial to consider a simplified problem in order to obtain guidance in the original problem. The delete relaxation of classical planning ignores delete effects, effects that set the truth value of a proposition to *false*. As action preconditions and the goal condition require propositions to evaluate to *true*, delete effects complicate plan search. Finding a relaxed plan on the other hand is possible in polynomial time because relaxed actions do not have delete effects and therefore each action has to be applied at most once. Plans for the original problem are also plans for the corresponding delete relaxed planning task. While finding *any* relaxed plan is possible in polynomial time, finding a shortest relaxed plan is NP-hard (Bylander 1994).

Numeric Relaxation Approaches

The idea behind delete relaxation is that facts that are reached once stay achieved. We will now discuss several ways to extend this concept to numeric planning. Combinations of these approaches are subject of future research.

Enumeration. The number of values that a variable can attain after applying a fixed number of actions is finite. An idea is to store the set of all attained values for each variable. However, the number of attainable values grows exponentially with the number of applied operators. As such it becomes infeasible to maintain the set of possible values quickly.

Example 2. Consider a numeric planning task with initial state $\mathcal{I}(\mathsf{x}) = 0$ and operators $o_1 = \langle \emptyset \rightarrow \{\mathsf{x} \mathrel{+}= 1\}\rangle$ and $o_2 = \langle \emptyset \rightarrow \{\mathsf{x} \div= 2\}\rangle$. Denoting by x_k, $k = 0, \ldots, 3$ the set of possible values of x after k steps, we have: $x_0 = \{0\}$, $x_1 = \{0, 1\}$, $x_2 = \{0, \frac{1}{2}, 1, 2\}$ and $x_3 = \{0, \frac{1}{4}, \frac{1}{2}, 1, 1\frac{1}{2}, 2, 3\}$.

For problems with bounded plan length, the enumeration approach requires space exponential in the bound. Even worse, the enumeration relaxation remains undecidable in general.

Theorem 1. *The numeric plan existence problem in an enumeration relaxation is undecidable.*

Proof sketch. We can basically adopt the proof for numeric planning by Helmert (2002). Formalizing Diophantine equations as planning problem results in a task that is not decidable as solutions have to be integers and the relaxation does not relax this property. ∎

Discretization. In order to restrict the number of possible values from the enumeration approach, multiple values can be aggregated into "buckets", where a representative approximates all values within. These representatives can be treated as multi-valued finite domain variables from classical SAS^+-planning. The state transition has to be defined in such a way that completeness is preserved – plans for the real problem have to act as plans in the relaxed problem. Proper abstractions offer potential for future research.

Higher values are better. Another approach is only feasible on a restricted set of planning tasks. If all preconditions and goals have the form $(\mathsf{x} > c)$ or $(\mathsf{x} \geq c)$ where x is a numeric variable and c a numeric constant, higher values are always beneficial for a variable. Numeric effects are only allowed to alter numeric variables by a positive constant, and therefore, decrease effects are considered harmful. The Metric-FF planning system uses this type of relaxation and Hoffmann (2003) shows that a large class of problems can be compiled into the required linear normal form.

Interval relaxation. An interval which encloses all values that a numeric variable can attain is a memory efficient method. Algebraic base operations are allowed in PDDL and supported by interval arithmetic. Therefore, we will focus on interval relaxation in the following section.

Interval Relaxation

In this section we elaborate on interval relaxation for numeric planning tasks. We will discuss the complexity of the plan existence problem for the presented semantics. We identify a class of tasks with *acyclic dependencies* between variables for which we can generate interval relaxed plans in polynomial time.

The interval relaxation of a numeric planning task differs only marginally from the original task description on a syntactic level. Propositional variables can now be both true and false at the same time and numeric variables are mapped to closed intervals.

Definition 2. Let Π be a numeric planning task. The interval delete relaxation $\Pi^+ = \langle \mathcal{V}_P^+, \mathcal{V}_N^+, \mathcal{O}^+, \mathcal{I}^+, \mathcal{G}^+\rangle$ of Π is a 5-tuple where $\mathcal{V}_P^+$ are the propositional variables from Π with the domains replaced by $\mathrm{dom}(\mathsf{v_p}) = \{\mathit{true}, \mathit{false}, \mathit{both}\}$ and $\mathcal{V}_N^+$ are the numeric variables with the domains replaced by closed intervals $\mathrm{dom}(\mathsf{v_n}) = \mathbb{I}_c$ for all $\mathsf{v_n} \in \mathcal{V}_N^+$. The initial state $\mathcal{I}^+$ is derived from $\mathcal{I}$ by replacing numbers $\mathcal{I}(\mathsf{v_n})$ with degenerate intervals $\mathcal{I}^+(\mathsf{v_n}) = [\mathcal{I}(\mathsf{v_n}), \mathcal{I}(\mathsf{v_n})]$ and $\mathcal{I}^+(\mathsf{v_p}) = \mathcal{I}(\mathsf{v_p})$. $\mathcal{G}^+$ is the goal condition.

The semantic of Π^+ draws on interval arithmetic. *Numeric expressions* are defined recursively: let e_1 and e_2 be numeric expressions. The interpretation of a constant expression is $s^+(c) = [c, c]$ and compound expressions are interpreted as $s^+(\mathsf{e}_1 \circ \mathsf{e}_2) = s^+(\mathsf{e}_1) \circ s^+(\mathsf{e}_2)$ for $\circ \in \{+, -, \times, \div\}$ where "$\circ$" now operates on intervals. For (goal and operator) conditions, the relaxed semantic is defined as follows: let $\mathsf{v_p} \in \mathcal{V}_P^+$ be a propositional variable, then $s^+ \vDash \mathsf{v_p}$ iff $s^+(\mathsf{v_p}) \in \{\mathit{true}, \mathit{both}\}$. For numeric constraints let e_1 and e_2 be numeric expressions, and $\bowtie \in \{<, \leq, =, \neq\}$ a comparison operator. Then $s^+ \vDash (\mathsf{e}_1 \bowtie \mathsf{e}_2)$ iff $\exists q_1 \in s^+(\mathsf{e}_1), \exists q_2 \in s^+(\mathsf{e}_2)$ with $q_1 \bowtie q_2$. This implies that two intervals can be "greater" and "less" than each other at the same time.

The semantic of numeric effects $\mathsf{v_n} \circ= \mathsf{e}$ is relaxed twice: $\mathsf{v_n}$ keeps its old value and gains all values up to the new value which is an interval in the relaxation. The state $\mathrm{app}_o^+(s^+) = s'^+$ resulting from an application of o with effect $\mathrm{eff} \in \{\mathrm{eff}_1, \ldots, \mathrm{eff}_n\}$ is then $s'^+(\mathsf{v_n}) = s^+(\mathsf{v_n}) \sqcup (s^+(\mathsf{v_n}) \circ s^+(\mathsf{e}))$ if eff is a numeric effect. As we use the convex union from Definition 1, $s'^+(\mathsf{v_n})$ contains all values between the old value of $\mathsf{v_n}$ and the evaluated expression $(s^+(\mathsf{v_n}) \circ s^+(\mathsf{e}))$. For propositional effects, $s'^+(\mathsf{v_p}) = \mathit{both}$ if the effect changes the truth value $\mathrm{eff}_i(\mathsf{v_p}) \neq s^+(\mathsf{v_p})$ of $\mathsf{v_p}$, and $s'^+(\mathsf{v_p}) = s^+(\mathsf{v_p})$ otherwise. Again, $s'^+(\mathsf{v}) = s^+(\mathsf{v})$ if v occurs in no effect.

Example 3. Applying $o = \langle \emptyset \rightarrow \{\mathsf{x} \times= \mathsf{e}\}\rangle$ in a state mapping $\mathsf{x} \mapsto [8, 10]$ and $\mathsf{e} \mapsto [-\frac{1}{2}, \frac{1}{2}]$ leads to a state $s'(\mathsf{x}) = [8, 10] \sqcup ([8, 10] \times [-\frac{1}{2}, \frac{1}{2}]) = [8, 10] \sqcup [-5, 5] = [-5, 10]$.

Interval Relaxation Complexity

For the classical relaxed planning problem, a relaxed plan can be found by applying all applicable operators in parallel until a fix-point is reached. As no effect can destroy a condition in the relaxed task, the number of operators in the planning task restricts the required number of iterations until a fix-point is reached. The task is solvable if the goal condition holds in the resulting state. A serialized plan can be obtained by ordering actions from the same parallel layer arbitrarily.

We employ a similar method for interval relaxed numeric planning. We have to approach the challenge that numeric operators can have to be applied arbitrarily often. An idea is to transform the planning task into a semi-symbolic representation which captures repeated application of operators with numeric effects. We define interval relaxed and *repetition relaxed* planning tasks which we refer to as *repetition relaxed* for short. In repetition relaxed planning tasks we simulate the behavior of applying numeric effects arbitrarily often *independently*. As we will see later, the independence assumption is not justified for numeric effects $\mathsf{v_n} \circ\!= \mathsf{e}$ where the expression of the assignment e depends on the affected variable $\mathsf{v_n}$. We show that an adaptation of the algorithm from classical relaxed planning can be used to find plans for repetition relaxed planning tasks with *acyclic dependencies*, where the variables in e do not depend on $\mathsf{v_n}$.

Repetition relaxed planning tasks use mixed bounded intervals, intervals whose bounds can either be open or closed, to capture the attainable values of a numeric variable. We are interested in the behavior of numeric effects in the limit. We use different fonts to distinguish a variable and its value e.g. $s(\mathsf{x}) = x$ in the following, whenever the state s is not essential. If an operator o has an additive effect $\mathsf{x} \pm\!= \mathsf{e}$ for $\pm\!= \in \{+\!=, -\!=\}$ which can extend a bound of x once, it can extend that bound to any value by applying o multiple times. The result of applying an additive effect arbitrarily often in a state s only depends on whether e can be negative, zero or positive. The behavior of multiplicative effects $\circledast\!= \in \{\times\!=, \div\!=\}$ is slightly more complex. Multiplicative effects $\mathsf{x} \circledast\!= \mathsf{e}$ can contract or expand depending on whether e contains elements with absolute value greater one and switch signs if e negative which results in up to seven different behaviors of e.

Definition 3. Let Π^+ be an interval relaxed planning task. An (interval and) repetition relaxed planning task of Π^+ is a 5-tuple $\Pi^\# = \langle V_P^+, \mathcal{V}_N^\#, \mathcal{O}^\#, \mathcal{I}^\#, \mathcal{G}^\# \rangle$ with propositional variables V_P^+ from Π^+. The domains of numeric variables $\mathrm{dom}(\mathsf{v_n}) = \mathbb{I}_m$ for $\mathsf{v_n} \in \mathcal{V}_N^\#$ are extended to mixed-bounded intervals. The initial state $\mathcal{I}^\#(\mathsf{v_p}) = \mathcal{I}^+(\mathsf{v_p})$ assigns the same truth value from $\mathcal{I}^+$ to each propositional variable $\mathsf{v_p}$ and each numeric variable $\mathsf{v_n}$ is initialized to the same closed degenerate interval $\mathcal{I}^\#(\mathsf{v_n}) = \mathcal{I}^+(\mathsf{v_n})$.

Again, the relaxation does not change much on a syntactical level. The main difference lies in the semantic of numeric effects. The semantic of *numeric expressions* can be transferred directly from the interval relaxation as interval arithmetic operations are also defined for mixed bounded intervals. The interpretation of a numeric expression is given as $s^\#(\mathsf{e}_1 \circ \mathsf{e}_2) = s^\#(\mathsf{e}_1) \circ s^\#(\mathsf{e}_2)$ for expressions e_1 and e_2 and $\circ \in \{+, -, \times, \div\}$. The semantic of conditions is again $s^\# \models \mathsf{v_p}$ iff $s^\#(\mathsf{v_p}) \in \{\mathit{true}, \mathit{both}\}$ for propositions $\mathsf{v_p} \in \mathcal{V}_P^\#$. For numeric constraints $\mathsf{e}_1 \bowtie \mathsf{e}_2$ where e_1 and e_2 are expressions and comparison operator $\bowtie \in \{<, \leq, =, \neq\}$, $s^\# \models (\mathsf{e}_1 \bowtie \mathsf{e}_2)$ iff $\exists q_1 \in s^\#(\mathsf{e}_1), \exists q_2 \in s^\#(\mathsf{e}_2)$ with $q_1 \bowtie q_2$.

The semantic of *numeric effects* captures the repeated application of actions. We first define the *repetition relaxed* semantic of $x \circ\!= e$ for intervals x and e with $\circ\!= \in \{:=, +\!=, -\!=, \times\!=, \div\!=\}$. Let $x_0 = x$ and $x_{i+1} = x_i \sqcup (x_i \circ e)$ for $i \geq 0$ where $(x : e)$ is defined as e for assign effects. Let $\mathrm{succ}_\circ(x, e) = \bigcup_{i=0}^{\infty} x_i$. We are interested in the result of applying an operator arbitrarily often individually for each effect, where the interval e is fixed even if the expression e depends on x. As $x_{i+1} \supseteq x_i$ by definition of the convex union and because all x_i are convex, the resulting set $\mathrm{succ}_\circ(x, e)$ is an interval. However, open bounded intervals can be generated by the limit value consideration. The state $\mathrm{app}_o^\#(s^\#) = s'^\#$ resulting from an application of o with effect $\mathrm{eff} = \{\mathrm{eff}_1, \dots, \mathrm{eff}_n\}$ is then again $s'^\#(\mathsf{v}) = s^\#(\mathsf{v})$ if v occurs in no effect, $s'^\#(\mathsf{v_p}) = \mathit{both}$ if $\mathrm{eff}_i(\mathsf{v_p}) \neq s^\#(\mathsf{v_p})$ is a propositional effect which changes the truth value of $\mathsf{v_p}$ and $s'^\#(\mathsf{v_p}) = s^\#(\mathsf{v_p})$ otherwise. For numeric effects $\mathrm{eff}_i = (\mathsf{v_n} \circ\!= \mathsf{e})$, $s'^\#(\mathsf{v_n}) = \mathrm{succ}_\circ(s^\#(\mathsf{v_n}), s^\#(\mathsf{e}))$.

Fixing expressions e of numeric effects $\mathsf{v_n} \circ\!= \mathsf{e}$ to the interval e they evaluate to in the previous state is beneficial to compute the successor, as changes in the assignment (which can be an arbitrary arithmetic expression) do not have to be considered immediately. The repetition relaxation $\Pi^\#$ of a planning task relaxes Π^+ further and plans for Π^+ are still plans for $\Pi^\#$. The reason is that each operator application can only extend the interval of affected numeric variables more than before. Evaluating the expression in the successor state $s'^\#(\mathsf{e})$ can only extend the interval $s^\#(\mathsf{e})$.

We want to use the fix-point algorithm which applies all operators of a planning task in parallel until a fix-point is reached to find a repetition relaxed plan. The successors $\mathrm{succ}_\circ(x, e)$ of numeric effects are defined by the limit $\bigcup_{i=0}^{\infty} x_i$ and we are interested in determining the result of such an effect in constant time. The result only depends on which of up to 21 symbolic *behavior classes* are covered by x and e. The seven *behavior classes* for e are $\mathcal{B}_e = \{(-\infty, -1), \{-1\}, (-1, 0), \{0\}, (0, 1), \{1\}, (1, \infty)\}$, and for x they are $\mathcal{B}_x = \{(-\infty, 0), \{0\}, (0, \infty)\}$. We decompose e and x into the hit behavior classes where $e \cap \tilde{e} \neq \emptyset$ for a behavior class $\tilde{e} \in \mathcal{B}_e$ and $x \cap \tilde{x} \neq \emptyset$ for a behavior class $\tilde{x} \in \mathcal{B}_x$, respectively. Table 1 contains partial behaviors $\mathcal{T}_\circ(x, e)$ for $\circ\!= \in \{+\!=, -\!=, \times\!=, \div\!=\}$ where $\mathcal{T}_\circ(x, e)$ is only defined if $x \subseteq \tilde{x} \in \mathcal{B}_x$ and $e \subseteq \tilde{e} \in \mathcal{B}_e$ and $\mathcal{T}_\circ(x, e)$ is the table entry with column $\tilde{x}$ and row $\tilde{e}$ in the table with the corresponding $\circ\!=$ operator. We use "indeterminate" parentheses $(\!\lfloor \underline{\cdot}, \overline{\cdot} \rceil\!)$ to denote intervals whose openness is determined by the terms

$+=$		$\tilde{e}$		
		$(-\infty, 0)$	$\{0\}$	$(0, \infty)$
$\tilde{x}$	$(-\infty, \infty)$	$(-\infty, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \infty)$

$-=$		$\tilde{e}$		
		$(-\infty, 0)$	$\{0\}$	$(0, \infty)$
$\tilde{x}$	$(-\infty, \infty)$	$\llparenthesis\underline{x}, \infty)$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$(-\infty, \overline{x}\rrparenthesis$

$\times=$		$\tilde{e}$						
		$(-\infty, -1)$	$\{-1\}$	$(-1, 0)$	$\{0\}$	$(0, 1)$	$\{1\}$	$(1, \infty)$
$\tilde{x}$	$(-\infty, 0)$	$(-\infty, \infty)$	$\llparenthesis\underline{x}, -\underline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \underline{x}\times\underline{e}\rrparenthesis$	$\llparenthesis\underline{x}, 0]$	$\llparenthesis\underline{x}, 0)$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$(-\infty, \overline{x}\rrparenthesis$
	$\{0\}$	$[0, 0]$						
	$(0, \infty)$	$(-\infty, \infty)$	$\llparenthesis-\overline{x}, \overline{x}\rrparenthesis$	$\llparenthesis\overline{x}\times\underline{e}, \overline{x}\rrparenthesis$	$[0, \overline{x}\rrparenthesis$	$(0, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \infty)$

$\div=$		$\tilde{e}$						
		$(-\infty, -1)$	$\{-1\}$	$(-1, 0)$	$\{0\}$	$(0, 1)$	$\{1\}$	$(1, \infty)$
$\tilde{x}$	$(-\infty, 0)$	$\llparenthesis\underline{x}, \underline{x}\div\overline{e}\rrparenthesis$	$\llparenthesis\underline{x}, -\underline{x}\rrparenthesis$	$(-\infty, \infty)$	undefined	$(-\infty, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$\llparenthesis\underline{x}, 0)$
	$\{0\}$	$[0, 0]$			undefined	$[0, 0]$		
	$(0, \infty)$	$\llparenthesis\overline{x}\div\overline{e}, \overline{x}\rrparenthesis$	$\llparenthesis-\overline{x}, \overline{x}\rrparenthesis$	$(-\infty, \infty)$	undefined	$\llparenthesis\underline{x}, \infty)$	$\llparenthesis\underline{x}, \overline{x}\rrparenthesis$	$(0, \overline{x}\rrparenthesis$

Table 1: Partial behaviors for numeric effects

contributing to it. For assignment effects $:=$ we do not need a table as the behavior is equal for all classes, and $\mathcal{T}_{:}(x, e) = \llparenthesis\min(\underline{x}, \underline{e}), \max(\overline{x}, \overline{e})\rrparenthesis$.

Theorem 2. *The partial behaviors $\mathcal{T}_{\circ}(x, e)$ are equal to $\mathrm{succ}_{\circ}(x, e)$ for $x \subseteq \tilde{x} \in \mathcal{B}_x$ and $e \subseteq \tilde{e} \in \mathcal{B}_e$.*

We prove Theorem 2 exemplarily for two of the less obvious entries in Table 1. The proofs for the remaining cases can be done similarly.

Proof for multiplication, $x \subseteq \tilde{x} = (0, \infty)$ and $e \subseteq \tilde{e} = (0, 1)$: We have to show that $\mathrm{succ}_{\times}(x, e) = (0, \overline{x}\rrparenthesis$.

"$\subseteq$": In order to prove $\mathrm{succ}_{\times}(x, e) \subseteq (0, \overline{x}\rrparenthesis$, we show that for every element $q \in \mathrm{succ}_{\times}(x, e) = \bigcup_{i=0}^{\infty} x_i$ there exists an index $k \in \mathbb{N}$ with $q \in x_k = \llparenthesis\underline{x_k}, \overline{x_k}\rrparenthesis$ and $x_k \subseteq (0, \overline{x}\rrparenthesis$. We prove this subset relation separately for each bound of x_k.
Lower bound: We show $\underline{x_k} > 0$ for all $k \in \mathbb{N}$ by induction. The base case $\underline{x_0} > 0$ holds as $x_0 = x \subseteq (0, \infty)$. Inductively $\underline{x_{i+1}} = \underline{x_i \sqcup (x_i \times e)} = \min(\underline{x_i}, \underline{x_i} \times \underline{e}, \underline{x_i} \times \overline{e}, \overline{x_i} \times \underline{e}, \overline{x_i} \times \overline{e})$. As $x \subseteq (0, \infty)$ and $e \subseteq (0, 1)$ are both positive, the result is positive as well. The minimum is obtained for $\underline{x_i} \times \underline{e}$ because e is a contraction. Thus, for all k it holds that $q \geq \underline{x_k} > 0$.
Upper bound: Again, we show $\overline{x_k} \leq \overline{x}$ for all $k \in \mathbb{N}$ by induction. The base case holds because $\overline{x_0} = \overline{x}$ with interval open/closed as for $\overline{x}$. The upper bound does not change in the inductive step and we have $\overline{x_{i+1}} = \overline{x_i}$ because $\overline{x_{i+1}} \geq \overline{x_i}$ by definition of the convex union and $\overline{x_{i+1}} \leq \overline{x_i}$ because $\overline{x_{i+1}} = \overline{x_i \times e}$ which is smaller than $\overline{x_i}$ because $0 < \underline{e} \leq \overline{e} < 1$ is a contraction. The upper bound of the interval remains open/closed as $\overline{x}$ as well for $\overline{x_{i+1}}$. Thus, for every q it holds that $q \leq \overline{x_k} \leq \overline{x}$. Together with the lower bound $0 < q$ we can conclude that $q \in (0, \overline{x}\rrparenthesis$.

"$\supseteq$": Now we have to show the converse direction $\mathrm{succ}_{\times}(x, e) \supseteq (0, \overline{x}\rrparenthesis$. Let $q \in (0, \overline{x}\rrparenthesis$. We have to show that $q \in \mathrm{succ}_{\times}(x, e)$. As $\overline{x_k} = \overline{x}$ for all $k \in \mathbb{N}$ we only have to show that there exists a $k \in \mathbb{N}$ with $q \in x_k = \llparenthesis\underline{x_k}, \overline{x_k}\rrparenthesis$ because $x_{i+1} \supseteq x_i$ and therefore $q \in \mathrm{succ}_{\times}(x, e) = \bigcup_{i=0}^{\infty} x_i$. Such a k exists because to obtain $\underline{x} \times \underline{e}^k < q$ respectively $\underline{e}^k < q \div \underline{x}$. Building the logarithm alters the inequality because $\underline{e} < 0$: $\log_{\underline{e}}(\underline{e}^k) > \log_{\underline{e}}(q \div \underline{x})$. Therefore, $q \in \mathrm{succ}_{\times}(x, e)$ for $k \geq \lceil \log_{\underline{e}}(q \div \underline{x}) \rceil$. $\square$

Proof for division, $x \subseteq \tilde{x} = (-\infty, 0)$ and $e \subseteq \tilde{e} = (-\infty, -1)$: We have to show that $\mathrm{succ}_{\div}(x, e) = \llparenthesis\underline{x}, \underline{x} \div \overline{e}\rrparenthesis$.

"$\subseteq$": We prove $\mathrm{succ}_{\div}(x, e) \subseteq \llparenthesis\underline{x}, \underline{x} \div \overline{e}\rrparenthesis$ again by showing that for every $q \in \mathrm{succ}_{\div}(x, e) = \bigcup_{i=0}^{\infty} x_i$ there exists an index $k \in \mathbb{N}$ with $q \in x_k = \llparenthesis\underline{x_k}, \overline{x_k}\rrparenthesis$. We show inductively that $x_k \subseteq \llparenthesis\underline{x}, \underline{x} \div \overline{e}\rrparenthesis$ for all $k \in \mathbb{N}$.
Base case: For $q \in x_0 = \llparenthesis\underline{x}, \overline{x}\rrparenthesis$ with bounds open/closed as in x, it is easy to show that also $q \in \llparenthesis\underline{x}, \underline{x} \div \overline{e}\rrparenthesis$ because from $x \subseteq (-\infty, 0)$ and $e \subseteq (-\infty, -1)$ we know that $\underline{x}$, $\overline{x}$, $\underline{e}$ and $\overline{e}$ are all negative and therefore $\underline{x} \div \overline{e} > 0 > \overline{x}$ so the upper bound on the right hand side is always greater than the upper bound of $\overline{x_0}$ and we have $\underline{x} = \underline{x_0} \leq q \leq \overline{x_0} < \underline{x} \div \overline{e}$.
Inductive step, lower bound: We have to prove that the lower bound $\underline{x_{i+1}} \geq \underline{x}$, with the induction hypothesis that $\underline{x_i} \geq \underline{x}$ holds. The new bound $\underline{x_{i+1}} = \underline{x_i \sqcup (x_i \div e)} = \min(\underline{x_i}, \underline{x_i} \div \underline{e}, \underline{x_i} \div \overline{e}, \overline{x_i} \div \underline{e}, \overline{x_i} \div \overline{e})$ is attained at $\underline{x_i}$ because $-\infty < \underline{e} < \overline{e} < 0$ and $\underline{x_i}$ are negative making the results $\underline{x_i} \div \underline{e}$ and $\underline{x_i} \div \overline{e}$ positive and obviously greater than 0 and as such also greater than $\underline{x_i}$. If the upper bound $\overline{x_i}$ is also negative, the minimum for $\underline{x_{i+1}}$ is clearly attained by $\underline{x_i}$ but even if x_i is positive, it is bounded by $0 \leq \overline{x_i} \leq (\underline{x} \div \overline{e})$. Because the division by $e \subseteq (-\infty, -1)$ is a contraction, the highest absolute value is attained by dividing by $\overline{e}$ but still $(\underline{x} \div \overline{e}) \div \overline{e} \geq (\overline{x_i} \div \overline{e}) \geq \underline{x_i}$. Therefore, the minimum of $\underline{x_{i+1}} = \underline{x_i}$. The lower bound remains open/closed for all k and $q \geq \underline{x_{i+1}} = \underline{x_i} = \underline{x_0} = \underline{x}$.
Inductive step, upper bound: We now have to show that $\overline{x_{i+1}} \leq \underline{x} \div \overline{e}$. The new bound $\overline{x_{i+1}}$ is computed as $\overline{x_i \sqcup (x_i \div e)} = \max(\overline{x_i}, \underline{x_i} \div \underline{e}, \underline{x_i} \div \overline{e}, \overline{x_i} \div \underline{e}, \overline{x_i} \div \overline{e})$. The maximum is attained at $\underline{x_i} \div \overline{e}$ (and at $\overline{x_i}$ if they are equal). The reasoning is as follows: all elements of e are negative and if $\underline{x_i}$ and $\overline{x_i}$ are both negative, $\underline{x_i}$ has the higher absolute value. If $\overline{x_i}$ is positive, the division by a negative number will not contribute to a higher upper bound. As $\underline{e} < \overline{e} < -1$ is a contraction, the highest value is achieved for $\underline{x} \div \overline{e}$ with bounds closed if the bounds corresponding to $\underline{x}$ and to $\overline{e}$ are both closed, and open otherwise. With $\underline{x_i} > \underline{x}$ by induction hypothesis, we can therefore conclude

that $\overline{x_{i+1}} \leq \underline{x_i} \div \overline{e} \leq \underline{x} \div \overline{e}$. Therefore, $q \leq \overline{x_k} \leq \underline{x} \div \overline{e}$. Together with the lower bound $\underline{x} \leq \underline{x_k} \leq q$ we can conclude that $q \in (\!|\underline{x}, \underline{x} \div \overline{e}|\!)$.

"$\supseteq$": We have to show that $\mathrm{succ}_\div(x, e) \supseteq (\!|\underline{x}, \underline{x} \div \overline{e}|\!)$. Let $q \in (\!|\underline{x}, \underline{x} \div \overline{e}|\!)$. We have to show that it then also follows that $q \in \mathrm{succ}_\div(x, e)$. As $\underline{x_k} = \underline{x}$ for all $k \in \mathbb{N}$ we only have to show that there exists a $k \in \mathbb{N}$ with $q \in x_k = (\!|\underline{x_k}, \overline{x_k}|\!)$ because $x_{i+1} \subseteq x_i$ and therefore $q \in \mathrm{succ}_\div(x, e) = \bigcup_{i=0}^{\infty} x_i$. The maximum is obtained after $k = 1$ steps because the maximum to compute $\overline{x_{i+1}} = \underline{x_i} \div \overline{e}$ only depends on the lower bound $\underline{x_i}$ which equals $\underline{x}$ for all $k \geq 0$. □

With such a decomposition, numeric effects can now be computed in constant time. Unfortunately, the union of the partial behaviors of an effect does not equal the semantic of a successor.

Hypothesis 1. The successor $\mathrm{succ}_\circ(x, e)$ of an effect $\mathsf{x} \circ\!\!= \mathsf{e}$ is the union of the successors obtained by decomposition of the effect into behavior classes, i.e. $\bigcup_{\tilde{x} \in \mathcal{B}_x, \tilde{e} \in \mathcal{B}_e} \mathrm{succ}_\circ(x \cap \tilde{x}, e \cap \tilde{e}) = \mathrm{succ}_\circ(x, e)$ where $\mathrm{succ}_\circ(\emptyset, e) = \mathrm{succ}_\circ(x, \emptyset) = \emptyset$.

Hypothesis 1 does not hold in general, as the following example illustrates. The successor can grow into behavior classes which were not covered by the decomposition:

Example 4. Let $o = \langle \emptyset \rightarrow \{\mathsf{x} \times\!\!= \mathsf{e}\} \rangle$ have a numeric effect on x in a state where $x = [1, 4]$ and $e = [-\frac{1}{2}, 2]$. The successor $\mathrm{succ}_\times(x, e)$ is $(-\infty, \infty)$. However, the partial behaviors of the decomposition are $\mathrm{succ}_\times(x, [-\frac{1}{2}, 0)) = [-2, 4]$, $\mathrm{succ}_\times(x, [0, 0]) = [0, 4]$, $\mathrm{succ}_\times(x, (0, 1)) = (0, 4]$, $\mathrm{succ}_\times(x, [1, 1]) = [1, 4]$ and $\mathrm{succ}_\times(x, (1, 2]) = [1, \infty)$. With the union $\mathrm{succ}_\times(x \cap \tilde{x}, e \cap \tilde{e}) = [-2, \infty)$ which differs from $\mathrm{succ}_\times(x, e) = (-\infty, \infty)$.

However, the number of behavior classes is restricted, and therefore, new behavior classes can only be hit a restricted number of times. The hypothesis can therefore be fixed by including the partial behaviors $\mathcal{T}_\circ(x, e)$ of the classes attained by x in a nested fix-point iteration: Let $x_0 = x$ and $x_{j+1} = \bigcup_{\tilde{x} \in \mathcal{B}_x, \tilde{e} \in \mathcal{B}_e} \mathrm{succ}_\circ(x_j \cap \tilde{x}, e \cap \tilde{e})$ with $\mathrm{succ}_\circ(\emptyset, e) = \mathrm{succ}_\circ(x, \emptyset) = \emptyset$ for $j \geq 0$. Let $\widetilde{\mathrm{succ}}_\circ(x, e) = \bigcup_{j=0}^{\infty} x_j$. Now, the newly attained behavior classes become part of the decomposition in the next iteration.

Example 5. Recall Example 4 starting with $x_0 = x = [1, 4]$ where the successor $\mathrm{succ}_\times(x_0 \cap \tilde{x}, e \cap \tilde{e}) = [-2, \infty)$. Building the decomposition over the newly achieved behavior classes with $x_1 = [-2, \infty)$ and $e = [-\frac{1}{2}, 2]$ contains among others $\mathrm{succ}_\times([-2, 0), (1, 2]) = (-\infty, 0)$. The union still contains partial behaviors which set the upper bound to ∞ and therefore $\mathrm{succ}_\times(x_1 \cap \tilde{x}, e \cap \tilde{e}) = (-\infty, \infty)$. Now, a fix-point is reached and $\widetilde{\mathrm{succ}}_\circ(x, e) = \mathrm{succ}_\circ(x, e)$.

Lemma 1. The union of decomposed successors $\widetilde{\mathrm{succ}}_\circ(x, e)$ converges after at most 21 steps.

Proof sketch. The number of behaviors in each class is restricted to $|\mathcal{B}_x| = 3$ and $|\mathcal{B}_e| = 7$. Most partial behaviors $\mathcal{T}_\circ(x, e)$ either set a new bound to a certain value (0 or $\pm\infty$), or leave a bound of x unchanged. The only unsafe cases are multiplications or divisions of a bound with -1 or e. However, none of these cases is problematic because e is fixed: $\mathcal{T}_\times(x, e)$ with $x \subseteq \tilde{x} = (-\infty, 0)$ and $e \subseteq \tilde{e} = (-1, 0)$ sets a new upper bound $\underline{x} \times \underline{e} > 0$. However, for all classes $\mathcal{T}_\times(x, e)$ with $x \subseteq \tilde{x} = (0, \infty)$, the upper bound is either set to ∞ or it remains the same. Therefore no problematic interactions occur. The same reasoning holds for $\mathcal{T}_\times(x, e)$ with $x \subseteq \tilde{x} = (0, \infty)$ and $e \subseteq \tilde{e} = (-1, 0)$ as well as $\mathcal{T}_\div(x, e)$ with $e \subseteq \tilde{e} = (-\infty, -1)$. ∎

The feasibility of a decomposition can therefore be reformulated to the following Theorem:

Theorem 3. *The successor* $\mathrm{succ}_\circ(x, e)$ *of an effect* $x \circ\!\!= e$ *is the fix-point of the convex union of the successors obtained by decomposition of the effect into behavior classes, i.e.* $\widetilde{\mathrm{succ}}_\circ(x, e) = \mathrm{succ}_\circ(x, e)$.

Proof sketch. It should be evident that $\widetilde{\mathrm{succ}}_\circ(x, e) \subseteq \mathrm{succ}_\circ(x, e)$. In the first iteration of $\widetilde{\mathrm{succ}}_\circ(x, e)$ all partial behaviors $\mathrm{succ}_\circ(x \cap \tilde{x}, e \cap \tilde{e})$ are operations on subsets of x and e. As interval arithmetic is well defined, an arithmetic operation on a interval x will therefore always subsume the interval resulting from the same operation of a sub-interval $x' \subseteq x$. During each iteration of $\widetilde{\mathrm{succ}}_\circ(x, e)$, the decomposition can only grow to behavior classes that were part of $\mathrm{succ}_\circ(x, e)$ in the first place.

The converse direction $\widetilde{\mathrm{succ}}_\circ(x, e) \supseteq \mathrm{succ}_\circ(x, e)$ is shown by contradiction. Let $q \in \mathrm{succ}_\circ(x, e)$ but not in $\widetilde{\mathrm{succ}}_\circ(x, e)$. Both successor functions are defined recursively starting with $x_0 = x$. Therefore $q \notin x_0$, and there has to be a $k > 0$ in $\mathrm{succ}_\circ(x, e)$ with $k_{k+1} = x_k \sqcup (x_k \circ e)$ so that $x_k \subset \widetilde{\mathrm{succ}}_\circ(x, e)$ but $x_{k+1} \not\subset \widetilde{\mathrm{succ}}_\circ(x, e)$. After k steps, the bound of the successors extended beyond the decomposition $\widetilde{\mathrm{succ}}_\circ(x, e)$ for the first time. Obviously, the new bound does not originate in x_k but the new interval x_{k+1} is obtained from $(x_k \circ e)$. The resulting interval depends on $\underline{x_k}$, $\overline{x_k}$, $\underline{e}$, $\overline{e}$ and in case of division also on whether $0 \in e$. Each combination of these extreme bounds is contained in one partial behavior $\mathcal{T}_\circ(x_k, e)$. If $(x_k \circ e)$ hits a new behavior class or extends the bounds within a behavior class, this is a contradiction to $\widetilde{\mathrm{succ}}_\circ(x, e)$ being a fix-point. If $(x_k \circ e)$ stays within a behavior, this is a contradiction to $\mathcal{T}_\circ(x_k, e)$ being well defined (Theorem 2). Thus, such a k cannot be found, and therefore, it is impossible for $q \in \mathrm{succ}_\circ(x, e)$ but not $q \in \widetilde{\mathrm{succ}}_\circ(x, e)$. ∎

With the help of the decomposed successor $\widetilde{\mathrm{succ}}_\circ(x, e)$ we can compute the result of applying an operator $\mathrm{app}_o^{\#}$ with the repetition relaxed semantic in constant time. This allows us to use the parallel fix-point algorithm from the classical case analogously: apply all applicable operators in parallel until a fix-point is reached. If the algorithm terminates, the plan is indeed a plan.

Theorem 4. *The parallel fix-point algorithm for repetition relaxed planning is correct, i.e. if the algorithm outputs an alleged plan, it is indeed a plan for* $\Pi^{\#}$.

Proof. Operators are only applied if the precondition is fulfilled. □

Unfortunately, the algorithm does not necessarily terminate. In the definition of the semantic of a repetition relaxed planning task, we fix the effect e even if it depends on x. However, this implicit independence assumption is not justified. Inspecting the entries in Table 1 reveals critical entries (marked in red) for multiplicative effects which contract x and flip the arithmetic sign at the same time. The same is true for assignment effects where $\mathcal{T}_{:}(x,e) = (\!|\min(\underline{x},\underline{e}),\max(\overline{x},\overline{e})|\!)$. In these cases, the new value of x can have a different behavior, if e also depends on x. As e can change when x changes, the algorithm does not necessarily terminate.

Example 6. Let $x = [-1,-1]$ and $o = \langle \emptyset \to \{\mathsf{x} \times\!= \mathsf{e}\}\rangle$ with $\mathsf{e} = -\frac{\mathsf{x}+1}{2}$. The goal is $\mathcal{G} = \{\mathsf{x} \geq 1\}$.

Applying the operator arbitrarily often according to the repetition semantic yields the following progression for k operator applications:

k	x	e
0	$[-1,-1]$	$[0,0]$
1	$[-1,0]$	$[-0.5,0]$
2	$[-1,0.5]$	$[-0.75,0]$
3	$[-1,0.75]$	$[-0.875,0]$
4	$[-1,0.875]$	$[-0.9375,0]$
5	$[-1,0.9375]$	$[-0.96875,0]$
	$\vdots$	$\vdots$

Obviously, interval x does *not* only change a restricted number of times, so the fix-point algorithm for interval relaxed numeric planning will not terminate.

If we succeeded in directly computing the fix-point to which the intervals converge with a symbolic interval we could continue the fix-point algorithm from here. In Example 6 we could continue if we would set $x = [-1,1)$ and $e = (-1,0]$. Unfortunately, the authors did not succeed in finding a general approach to do so (or to prove that such a general approach does not exist). Instead, we will now restrict the problem to planning tasks where the aforementioned problem does not occur. The problem in Example 6 is that e depends on x. Thus, we will restrict planning tasks to contain only effects where the assigned expression is independent from the affected variable. We will then show that such planning tasks are solvable in polynomial time.

Definition 4. A numeric variable v_1 is *directly dependent* on a numeric variable v_2 in task Π if there exists an $o \in \mathcal{O}$ with a numeric effect $\mathsf{v}_1 \circ\!= \mathsf{e}$ so that e contains v_2.

Note that a variable can be *directly dependent* on itself. Also, the definition of *direct dependence* does not consider operator applicability.

Definition 5. A planning task Π is an *acyclic dependency* task, if the *direct dependency* relation is acyclic.

Theorem 5. *The parallel fix-point algorithm for repetition relaxed planning terminates for* acyclic dependency *tasks.*

Proof. As the planning task has *acyclic dependencies*, the *direct dependency* relation induces a topology. Let a *phase* of the algorithm be a sequence of parallel operator applications, where no new operator becomes applicable. During each *phase*, we consider numeric effects in topological order concerning the dependency graph. Let $V_N^{\#l} \subseteq V_N^{\#}$ be the variables in dependency layer l. We iterate over the layers $k \geq 0$ of the topology assuming that a fix-point is reached for all variables $V_N^{\#k}$. Variables $V_N^{\#k+1}$ only depend on variables $V_N^{\#l}$ with $0 \leq l \leq k$ or on constants. A fix-point is reached for all those variables by induction hypothesis. Inductively, we can assume that the expressions of numeric effects which alter the variables of layer $V_N^{\#k+1}$ are fixed. Therefore, the successor $\mathrm{succ}_\circ(x,e)$ of an effect $(\mathsf{x} \circ\!= \mathsf{e})$ with $x = s^{\#}(\mathsf{x})$ and $e = s^{\#}(\mathsf{e})$ does not change the variable more than once (or more than 21 times, if we also consider the intermediate variable updates of the nested fix-point iteration from Lemma 1).

The number of *phases* is restricted, too, with the same argument as for the fix-point algorithm in the classical case. No precondition can be invalidated once it holds, and during each phase at least one operator which was not applicable before must become applicable. The number of phases is therefore restricted to the number of operators in the planning task. $\square$

Theorem 6. *The fix-point algorithm for repetition relaxed planning is complete for* acyclic dependency *tasks.*

Proof. We prove completeness by contradiction and show that it is impossible that the algorithm terminates and reports unsolvable although a plan exists. Now assume there is a plan, but the algorithm terminates and reports unsolvable. Therefore, a satisfiable condition must have been unsatisfied. For propositional conditions, this is impossible, as $s^{\#}(\mathsf{v_p}) \vDash \mathsf{v_p}$ if $\mathsf{v_p} \in \{\textit{true}, \textit{both}\}$ and no effect can set a propositional variable to *false*. Additionally, all operators are applied as soon as they are applicable. Thus, without loss of generality, a satisfiable numeric constraint was not achieved by the algorithm. This implies that a numeric effect $(\mathsf{v_n} \circ\!= \mathsf{e})$ would have been able to assign a value to a variable which was not reached by our algorithm. Therefore, the successor defined by the semantic $\mathrm{succ}_\circ(s^{\#}(\mathsf{v_n}), s^{\#}(\mathsf{e}))$ has to be different from the successor computed by the algorithm $\widetilde{\mathrm{succ}}_\circ(s^{\#}(\mathsf{v_n}), s^{\#}(\mathsf{e}))$ which is impossible for numeric tasks with Theorem 3, a contradiction. $\square$

Until now we have an algorithm which can compute parallel plans for repetition relaxed planning tasks in polynomial time for acyclic dependency tasks. As intervals can only grow by applying an operator, the plan can be serialized by applying parallel operators from the same layer in an arbitrary order. Beneficial effects may make the application of some operators unnecessary, but it cannot harm conditions.

We are interested in plans for the interval relaxation without the symbolic description of numeric variables. We will now show that we can derive interval relaxed plans π^+ from repetition relaxed plans $\pi^{\#}$.

Theorem 7. *Plans for the repetition relaxation correspond to plans for the interval relaxation.*

Proof sketch. A serialization of a repetition relaxed plan $\pi^{\#} = \langle o_1, o_2, \ldots, o_n\rangle$ where $0 < i < n$ are the operators applied in the i-th step where the same operator can be applied in multiple steps. We seek to find a repetition constant k_i for each operator in order to satisfy the constraints from interval relaxed planning corresponding to those of the repetition relaxed planning plan. However, repetition relaxed tasks operate on mixed bounded intervals $\mathbb{I}_m$ whereas interval relaxed tasks are restricted to closed intervals. Thus, we have to explicate the interval bounds as well. The repetition relaxed fix-point algorithm is split into *phases*, were during each phase the same operators are applicable. Within each phase, the operators are applied in parallel at most 21 times for each variable (if all variables have different topology levels in the dependence graph). In order to determine the repetition constants k_i, we look at each constraint $[\underline{a}, \overline{a}] \bowtie [\underline{b}, \overline{b}]$. By definition of $\bowtie$, there exist q_a and q_b in the respective intervals so that $q_a \bowtie q_b$. Let q_a and q_b be such numbers which satisfy the constraint $a \bowtie b$ where a and b are intervals which are obtained by evaluating the corresponding expressions $s^{\#}(\mathsf{e_a})$ and $s^{\#}(\mathsf{e_b})$. We investigate each expression individually. For each expression we have a target value q. For the constraints above the expressions are $\mathsf{e_a}$ and $\mathsf{e_b}$ and the corresponding target values q_a and q_b. Unless the expression is a variable, the target value has to be obtained recursively from the expressions $\mathsf{e_1} \circ \mathsf{e_2}$.

Example 7. Let $x = [0, 1)$, $y = [0, 1)$ and $z = (1.7, 3]$ be the symbolic values of variables x, y and z with a condition $\mathsf{x{+}y} > \mathsf{z}$. From $\mathsf{e_a} = \mathsf{x{+}y} \mapsto [0, 2)$ and $\mathsf{e_b} = \mathsf{z}$ we choose an arbitrary $q_a = 1.9 \in s^{\#}(\mathsf{e_a})$ an arbitrary $q_b = 1.8 \in s^{\#}(\mathsf{e_b})$ from within the expression intervals so that the constraint is satisfied. Now we have to recursively find appropriate q_x and q_y in the sub-expressions. A leeway of $2 - 1.9 = 0.1$ can be distributed arbitrarily to the target values of the sub-expressions. We could for example continue with target values 0.95 for x and y each.

We can choose arbitrary target values for the sub-expressions within a leeway of feasible choices. Eventually, all expressions induce target values for the numeric variables. This can induce multiple different target values for each variable where only the most extreme target values have to be considered (an interval including the most extreme target values will also include intermediate target values). All target values originate from a repetition relaxed symbolic state, so they are indeed reachable. In the repetition relaxed plan, each operator which has a numeric effect on a variable with a target value achieved the symbolic value for this variable with a partial behavior from Table 1. For each operator, the constant k is now computed by solving $x \pm k \cdot e = q$ for additive effects and $x \divideontimes e^k = q$ for multiplicative effects. The k is therefore the same k from the proof of Theorem 2. Each operator then has to be applied n times, where n is the sum over all k_p in the phases of the algorithm, where k_p is the maximum number of applications required for that operator in that phase. ■

Theorem 8. *The problem to generate an interval relaxed numeric plan is in* P *for tasks with* acyclic dependencies*.*

Proof. The fix-point algorithm for repetition relaxed planning tasks is correct (Theorem 4) and complete (Theorem 6) and it terminates in polynomial time (Theorem 5). Therefore, generating a repetition relaxed plan $\pi^{\#}$ is possible in polynomial time. An interval relaxed plan π^{+} can be constructed from $\pi^{\#}$ (Theorem 7) in polynomial time. □

The definition of a relaxation is *adequate* (Hoffmann 2003) if it is *admissible*, i.e. any plan π for the original task Π is also a relaxed plan for Π^+, if it offers *basic informedness*, i.e. the empty plan is a plan for Π iff it is a plan for Π^+ and finally the plan existence problem for the relaxation is in P.

Theorem 9. *The* interval relaxation *is adequate for* acyclic dependency *tasks.*

Proof. Admissibility. After each step of the plan π, if propositional variables of the relaxed state differ from the original state, they assign to *both* which cannot invalidate any (goal or operator) conditions. The original value of numeric variables is contained in the interval of the relaxed state. As comparison constraints are defined with the relaxed semantic that a constraint holds if it holds for any pair of elements from the two intervals, admissibility follows directly.

Basic informedness. No (goal or operator) conditions are dropped from the task. Relaxed numeric variables are mapped to degenerate intervals which only contain one element. Therefore, conditions in the original task $x \bowtie y$ correspond to interval constraints $[x, x] \bowtie [y, y]$ which are satisfied iff they are satisfied in the relaxed task.

Polynomiality. As a corollary to Theorem 8, we can also conclude that interval relaxed numeric plan existence is in P for tasks with *acyclic dependencies*.

The interval relaxation is admissible and offers basic informedness. For *acyclic dependency* tasks, the plan existence problem can be decided in polynomial time. Thus, the *interval relaxation* is adequate. □

The proposed relaxation advances the state of the art even though the adequacy of interval relaxation was only shown for a restricted set of tasks. However, the requirement of acyclic dependency for numeric expressions is a strict generalization of expressions e being required to be constant, which is required for other state-of-the-art approaches e.g. (Hoffmann 2003). On the practical side, many interesting planning problems are restricted to constant expressions.

Conclusion and Future Work

We presented interval algebra as a means to carry the concept of a delete relaxation from classical to numeric planning. We proved that this relaxation is adequate for *acyclic dependency tasks*, tasks where the expressions of numeric effects do not depend on the affected variable. The complexity of the approach for arbitrary interval relaxed planning problems remains an open research issue though. It is imaginable that a clever approach can find the fix-point of arbitrary operator application in polynomial time.

In the future, we intend to adapt the most iconic heuristics from classical planning, h_{max}, h_{add} and h_{FF} to the interval relaxation framework.

Acknowledgments

This work was partly supported by the DFG as part of the SFB/TR 14 AVACS.

References

Bäckström, C., and Nebel, B. 1993. Complexity results for SAS$^+$ planning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993)*.

Bonet, B., and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Proceedings of the 5th European Conference on Planning (ECP 1999)*, 360–372.

Bonet, B., and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A Robust and Fast Action Selection Mechanism for Planning. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI 1997/ IAAI 1997)*, 714–719.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence 69 (AI 1994)* 165–204.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains. In *Proceedings of the 20th International Conference on Automated Planning and Search (ICAPS 2008)*.

Fikes, R., and Nilsson, N. J. 1971. STRIPS: a New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence 2 (AI 1971)* 189–208.

Fox, M., and Long, D. 2003. PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research 20 (JAIR 2003)* 61–124.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 303–312.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables. *Journal of Artificial Intelligence Research 20 (JAIR 2003)* 291–341.

Moore, R. E.; Kearfott, R. B.; and Cloud, M. J. 2009. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics.

Young, R. C. 1931. The Algebra of Many-valued Quantities. *Mathematische Annalen 104* 260–290.