Complexity of Interval Relaxed Numeric Planning

Johannes Aldinger, 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 computationally hard even in its most basic form as STRIPS planning. We are interested in numeric planning with instantaneous actions, a problem that is not 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 show that plan existence can be decided in polynomial time for tasks where dependencies between numeric effects are acyclic.

1 Motivation

Automated planning can be used to solve many real world problems where a *goal* is reached by applying *operators* that change the *state* of the world. In classical planning, the world is modeled with Boolean variables. Modeling of physical properties (e.g. velocity) or resources (e.g. fuel level) requires real-valued variables instead. Unlike classical planning, which is PSPACE-complete [4], numeric planning is *undecidable* [7]. Even though completeness of numeric planning algorithms can therefore not be achieved in general, numeric planners can find plans or an assurance that the problem is unsolvable for many practical tasks.

Heuristic search is a predominant approach to solve planning problems. One way to obtain heuristic guidance is to ignore negative interactions between the operators. The underlying assumption of a *delete relaxation* is that propositions which are achieved once during planning can not be invalidated (deleted). More recent planning systems are usually not restricted to propositional state variables. Instead they use the SAS^+ formalism [2], which allows for (finite-domain) multi-valued variables and 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. A memory efficient approach to capture possibly infinitely many values of a numeric variable is to consider the interval that encloses the reached values. The methods to deal with intervals have been studied in the field of interval arithmetic [10], which has been used in mathematics for decades [9] and enables us to deal with intervals in terms of basic numeric operations.

Extending classical delete relaxation heuristics to numeric problems has been done before, albeit only for a subset of numeric tasks, where numeric variables

can only be manipulated in a restricted way. The Metric-FF planning system [8] tries to convert the planning task into a *linear numeric* task, which ensures that variables can "grow" in only one direction. When high values of a variable are beneficial to fulfill the preconditions, *decrease* effects are considered harmful. Another approach to solve *linear* numeric planning problems is to encode numeric variables in a linear program and solve constraints with an LP-solver. Coles et al. [5] analyze the planning problem for consumers and producers of resources to obtain a heuristic that ensures that resources are not more often consumed than produced or initially available. The RANTANPLAN planner [3] uses linear programs in the context of planning as satisfiability modulo theories. In contrast, we are interested in approaching numeric planning supporting all arithmetic base operations by generalizing relaxation heuristics.

2 Basics

In this section, we outline numeric planning with instantaneous actions, which is expressible in PDDL2.1, layer 2 [6]. We describe interval arithmetic, the technique we use to extend delete relaxation heuristics to numeric planning.

2.1 Numeric Planning with Instantaneous Actions

Given a set of variables $\mathcal{V}$ with domains $\mathrm{dom}(\mathsf{v})$ for all $\mathsf{v} \in \mathcal{V}$, 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 state s by $s(\mathsf{v})$. Whenever the state s is not essential, we use the same letter in different fonts to distinguish a variable or expression (sans-serif) and its value or evaluation (italic), e.g. $s(\mathsf{x}) = x$.

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 $\mathbb{Q}^\infty := \mathbb{Q} \cup \{-\infty, \infty\}$. $\mathcal{O}$ is a set of operators, $\mathcal{I}$ the initial state and $\mathcal{G}$ 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 $\mathsf{e_1}$ and $\mathsf{e_2}$ recursively defined over variables $\mathcal{V}_N$ and constants from $\mathbb{Q}$. A *numeric constraint* $(\mathsf{e_1} \bowtie \mathsf{e_2})$ compares numeric expressions $\mathsf{e_1}$ and $\mathsf{e_2}$ 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 $\langle \mathrm{pre} \to \mathrm{eff} \rangle$ and consist of a condition pre and a set of effects $\mathrm{eff} = \{\mathrm{eff}_1, \ldots, \mathrm{eff}_n\}$ containing at most one numeric effect for each numeric variable and at most one truth assignment for each propositional variable.

The semantics of a numeric planning task is straightforward. For constants $c \in \mathbb{Q}$, $s(c) = c$. Numeric expressions $(\mathsf{e_1} \circ \mathsf{e_2})$ for $\circ \in \{+, -, \times, \div\}$ are recursively evaluated in state s: $s(\mathsf{e_1} \circ \mathsf{e_2}) = s(\mathsf{e_1}) \circ s(\mathsf{e_2})$. Satisfaction of conditions in a state s is defined as follows: for propositional variables $\mathsf{v_p} \in \mathcal{V}_P$, $s \vDash \mathsf{v_p}$ iff $s(\mathsf{v_p}) = true$, for numeric constraints $(\mathsf{e_1} \bowtie \mathsf{e_2})$, with expressions $\mathsf{e_1}$, $\mathsf{e_2}$ and $\bowtie \in \{\leq, <, =, \neq\}$,

$s \vDash (\mathsf{e}_1 \bowtie \mathsf{e}_2)$ iff $s(\mathsf{e}_1) \bowtie s(\mathsf{e}_2)$ and finally, for conjunctive conditions $s \vDash p_1 \wedge p_2$ iff $s \vDash p_1$ and $s \vDash p_2$.

An operator $o = \langle \text{pre} \rightarrow \text{eff} \rangle$ is applicable in s iff $s \vDash \text{pre}$ and if for none of its numeric effects a division by zero occurs. The successor state $\text{app}_o(s) = s'$ resulting from an application of o is defined as follows: if $\text{eff}_i \in \{\text{eff}_1, \ldots, \text{eff}_n\}$ is a numeric effect $\mathsf{v_n} \circ\!= \mathsf{e}$ with $\circ\!= \in \{+=, -=, \times=, \div=\}$, then $s'(\mathsf{v_n}) = s(\mathsf{v_n}) \circ s(\mathsf{e})$. 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 $\mathsf{e_p} \in \{\mathit{true}, \mathit{false}\}$, then $s'(\mathsf{v_p})$ 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 recalls the foundations of interval arithmetic.

2.2 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 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 identified with the degenerate interval $[q, q]$. In interval arithmetic, the basic operations 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: $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] = \left[\min(\underline{xy}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}), \max(\underline{xy}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy})\right]$,
- 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, zero or negative, which determine which of the bounds diverge and refer to the literature [9].

Analogously we define open 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.

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.*

If $x \cap y = \emptyset$, u also includes intermediate values not present in either x or y.

3 Delete Relaxation

In this section, we discuss extensions of delete relaxation to numeric planning. The idea behind delete relaxation is that values of a variable that are achieved once remain achieved. We discuss several ways to extend this concept to numeric planning.

Accumulation Semantics. In the accumulation semantics, instead of *changing* their values, variables *accumulate* all values achieved so far. The number of accumulated values after k parallel steps is finite, but generally exponential in k. Therefore, it quickly becomes infeasible to maintain the set of possible values, as can be seen in the task with $o_1 = \langle \emptyset \to \{\mathsf{x} \mathrel{+}= 1\}\rangle$, $o_2 = \langle \emptyset \to \{\mathsf{x} \mathrel{\div}= 2\}\rangle$ and $\mathcal{I}(\mathsf{x}) = 0$. Denoting by x_k, $k = 0, \ldots, 3$, the possible values of x after k parallel steps, we get $x_0 = \{0\}$, $x_1 = \{0, 1\}$, $x_2 = \{0, \frac{1}{2}, 1, 2\}$ and $x_3 = \{0, \frac{1}{4}, \frac{1}{2}, 1, \frac{3}{2}, 2, 3\}$. Besides this observation for *bounded* plan existence, one can also show that *unbounded* plan existence wrt. the accumulation semantics is still undecidable. To see this, we can adapt the undecidability proof for numeric planning by Helmert [7]. A reduction of the search for solutions to Diophantine equations to numeric planning wrt. the accumulation semantics shows that the latter problem is undecidable, since solutions to Diophantine equations have to be integers and the delete relaxation does not relax this property.

Accumulation Semantics for Positive Tasks. One possible approach to dealing with the exploding number of accumulated values is the restriction to tasks where higher values are always better. Then, instead of storing *all values* a variable has attained so far, it is sufficient to store (an upper bound on) the *highest value*. A sufficient criterion for this is that all preconditions and goals have the form $(\mathsf{x} > c)$ or $(\mathsf{x} \geq c)$, where x is a numeric variable and c a constant, and that all numeric effects only add or subtract a positive constant to or from a variable. The Metric-FF planner uses this type of relaxation, and Hoffmann [8] shows that a large class of problems can be compiled into the required linear normal form.

Interval Relaxation. One can handle a larger class of tasks with higher precision by only making the assumption that one of the *extreme* values, the highest or lowest, is best. This necessitates keeping track of two values for each variable, a lower bound $\underline{x}$ and an upper bound $\overline{x}$. As long as preconditions and goals are comparisons of variables to constants and effects only add or subtract constants, it is insubstantial whether one considers the values between $\underline{x}$ and $\overline{x}$ reached or not. By considering the entire interval reached in a relaxed sense, one can handle even more expressive tasks. In particular, when allowing divisions in effects, besides "higher values are better" and "lower values are better", one also has the objective "values closer to c are better" for constants c. Then, if c is in the interval between $\underline{x}$ and $\overline{x}$, one may assume that values arbitrarily close to c can be reached, whereas otherwise, one can assess the proximity to c achieved so far. Since the algebraic base operations that are allowed in PDDL are also supported by interval arithmetic, we consider interval relaxation a viable approach and focus on it in the following section.

4 Interval Relaxation

This section elaborates on interval relaxation for numeric planning tasks. We discuss the complexity of the plan existence problem for the presented semantics.

The interval relaxation of a numeric planning task differs only marginally from the original task description on a syntactic level. The domains of variables are change and propositional variables can now be both true and false at the same time whereas 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* $\mathrm{dom}(\mathsf{v_p}) = \{true, false, both\}$ *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 $\mathcal{I}^+(\mathsf{v_p}) = \mathcal{I}(\mathsf{v_p})$. Operators $\mathcal{O}^+ = \mathcal{O}$ and the goal condition $\mathcal{G}^+ = \mathcal{G}$ remain unchanged.*

The semantics of Π^+ draws on interval arithmetic. *Numeric expressions* are defined recursively. Constant expressions are interpreted as $s^+(c) = [c, c]$ and compound expressions $\mathsf{e_1}$ and $\mathsf{e_2}$ 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 semantics is defined as follows: let $\mathsf{v_p} \in \mathcal{V}_P^+$ be a propositional variable, then $s^+ \vDash \mathsf{v_p}$ iff $s^+(\mathsf{v_p}) \in \{true, both\}$. For numeric constraints let $\mathsf{e_1}$ and $\mathsf{e_2}$ 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 once.

The "values remain achieved" idea for numeric effects $\mathsf{v_n} \circ\!= \mathsf{e}$ is obtained by the semantics in that $\mathsf{v_n}$ 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 is then $s'^+(\mathsf{v_n}) = s^+(\mathsf{v_n}) \sqcup (s^+(\mathsf{v_n}) \circ s^+(\mathsf{e}))$ if $\mathrm{eff}_i \in \{\mathrm{eff}_1, \ldots, \mathrm{eff}_n\}$ 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, $s'^+(\mathsf{v_p}) = both$ 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. A relaxed plan is defined in the obvious way.

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

To compute relaxed plans, we can proceed as in classical planning: there, a relaxed plan can be found by iteratively applying all applicable operators to the current relaxed state in parallel, and terminating if either a fix-point is reached or the current relaxed state satisfies the goal condition. As no effect can invalidate any condition in the relaxed task, the number of iterations is restricted by the number of operators in the task. A serialized plan can be obtained by ordering parallel actions arbitrarily.

For interval relaxed numeric planning, we have to take into account that numeric operators may have to be applied arbitrarily often. Our idea is to transform the planning task into a semi-symbolic representation that captures repeated application of operators with numeric effects. We define *repetition relaxed* planning tasks, where 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 e depends on the affected variable $\mathsf{v_n}$. However, we can find plans for tasks with *acyclic dependencies* in polynomial time with our approach.

Repetition relaxed planning tasks use mixed-bounded intervals to capture the attainable values of a numeric variable. We are interested in the behavior of numeric effects in the limit. If an operator o has an additive effect $\mathsf{x} \pm\!= \mathsf{e}$ for $\pm\!= \in \{+\!=, -\!=\}$ that 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 $\divideontimes\!= \in \{\times\!=, \div\!=\}$ is slightly more complex. Multiplicative effects $\mathsf{x} \divideontimes\!= \mathsf{e}$ can *contract* or *expand* depending on whether e contains elements with absolute value greater one and switch signs if e contains negative elements, resulting in up to seven different behaviors of e.

Definition 3. *Let Π^+ be an interval relaxed planning task. The* repetition relaxation *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 $\mathcal{I}^\#$ maps variables to the same truth (for propositional variables $\mathsf{v_p}$) respectively the same closed degenerate interval (for numeric variables $\mathsf{v_n}$) from $\mathcal{I}^+$. Operators $\mathcal{O}^\# = \mathcal{O}^+$ and the goal condition $\mathcal{G}^\# = \mathcal{G}^+$ remain unchanged.*

Again, the relaxations differ mainly in the semantics of numeric effects. The semantics of *numeric expressions* is 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 $\mathsf{e_1}$ and $\mathsf{e_2}$ and $\circ \in \{+, -, \times, \div\}$. The semantics of conditions is $s^\# \vDash \mathsf{v_p}$ iff $s^\#(\mathsf{v_p}) \in \{\mathit{true}, \mathit{both}\}$ for propositions $\mathsf{v_p} \in \mathcal{V}_P^\#$. For numeric constraints, where $\mathsf{e_1}$ and $\mathsf{e_2}$ are expressions and $\bowtie \in \{<, \leq, =, \neq\}$ is a comparison operator, $s^\# \vDash (\mathsf{e_1} \bowtie \mathsf{e_2})$ iff $\exists q_1 \in s^\#(e_1)$, $\exists y \in s^\#(e_2)$ with $q_1 \bowtie q_2$.

The semantics of *numeric effects* captures the repeated application of actions. We first define the *repetition relaxed* semantics of $x \circ\!= e$ for intervals x and e with $\circ\!= \in \{:\!=, +\!=, -\!=, \times\!=, \div\!=\}$. Let $x_0 = x$ and $x_{k+1} = x_k \sqcup (x_k \circ e)$ for $k \geq 0$ where $(x : e)$ is defined as e for assign effects. Let $\mathrm{succ}_\circ(x, e) = \bigsqcup_{k=0}^{\infty} x_k$. 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 $x_{k+1} \supseteq x_k$ by definition of the convex union and because all x_k are convex, the resulting set $\mathrm{succ}_\circ(x, e)$ is an interval. However, open intervals can be generated in the limit. The state $\mathrm{app}_o^\#(s^\#) = s'^\#$ resulting from an application of o with effect $\mathrm{eff} = \{\mathrm{eff}_1, \ldots, \mathrm{eff}_n\}$ is again $s'^\#(\mathsf{v}) = s^\#(\mathsf{v})$ if v

Table 1. Partial behaviors for numeric effects

$+=$		$\tilde{e}$		
		$(-\infty, 0)$	$\{0\}$	$(0, \infty)$
$\tilde{x}$	$(-\infty, \infty)$	$(-\infty, \overline{x}\rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \infty)$

$-=$		$\tilde{e}$		
		$(-\infty, 0)$	$\{0\}$	$(0, \infty)$
$\tilde{x}$	$(-\infty, \infty)$	$\lbrack\!\lbrack \underline{x}, \infty)$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$(-\infty, \overline{x}\rbrack\!\rbrack$

$\times=$		$\tilde{e}$						
		$(-\infty, -1)$	$\{-1\}$	$(-1, 0)$	$\{0\}$	$(0, 1)$	$\{1\}$	$(1, \infty)$
$\tilde{x}$	$(-\infty, 0)$	$(-\infty, \infty)$	$\lbrack\!\lbrack \underline{x}, -\underline{x} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \underline{x} \times \underline{\mathsf{e}} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, 0]$	$\lbrack\!\lbrack \underline{x}, 0)$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$(-\infty, \overline{x}\rbrack\!\rbrack$
	$\{0\}$	$[0, 0]$						
	$(0, \infty)$	$(-\infty, \infty)$	$(\!\lbrack -\overline{x}, \overline{x} \rbrack\!\rbrack$	$\lbrack\!\lbrack \overline{x} \times \overline{\mathsf{e}}, \overline{x} \rbrack\!\rbrack$	$[0, \overline{x}\rbrack\!\rbrack$	$(0, \overline{x}\rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \infty)$

$\div=$		$\tilde{e}$						
		$(-\infty, -1)$	$\{-1\}$	$(-1, 0)$	$\{0\}$	$(0, 1)$	$\{1\}$	$(1, \infty)$
$\tilde{x}$	$(-\infty, 0)$	$\lbrack\!\lbrack \underline{x}, \underline{x} \div \overline{\mathsf{e}} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, -\underline{x} \rbrack\!\rbrack$	$(-\infty, \infty)$	undefined	$(-\infty, \overline{x}\rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$\lbrack\!\lbrack \underline{x}, 0)$
	$\{0\}$	$[0, 0]$			undefined	$[0, 0]$		
	$(0, \infty)$	$\lbrack\!\lbrack \overline{x} \div \overline{\mathsf{e}}, \overline{x} \rbrack\!\rbrack$	$(\!\lbrack -\overline{x}, \overline{x} \rbrack\!\rbrack$	$(-\infty, \infty)$	undefined	$\lbrack\!\lbrack \underline{x}, \infty)$	$\lbrack\!\lbrack \underline{x}, \overline{x} \rbrack\!\rbrack$	$(0, \overline{x}\rbrack\!\rbrack$

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 changing the truth value of $\mathsf{v_p}$ and $s'^{\#}(\mathsf{v_p}) = s^{\#}(\mathsf{v_p})$ otherwise. For numeric effects $\mathrm{eff}_i = (\mathsf{v_n} \circ\!= \mathsf{e})$, $s'^{\#}(\mathsf{v_n}) = \mathrm{succ}_\circ(s^{\#}(\mathsf{v_n}), s^{\#}(\mathsf{e}))$. Repetition relaxed plans are defined in the obvious way.

Fixing expressions e of numeric effects 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.

We want to use the fix-point algorithm that 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 *behavior classes* for x are $\mathcal{B}_x = \{(-\infty, 0), \{0\}, (0, \infty)\}$, and for e they are $\mathcal{B}_e = \{(-\infty, -1), \{-1\}, (-1, 0), \{0\}, (0, 1), \{1\}, (1, \infty)\}$. We decompose e and x into the behavior classes they hit, i.e, 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 $(\!\lbrack \cdot, \overline{\cdot} \rbrack\!\rbrack$ to denote intervals whose openness is determined by the contributing terms. For assignment effects $:=$ we do not need a table as $\mathcal{T}_:(x, e) = (\!\lbrack \min(\underline{x}, \underline{\mathsf{e}}), \max(\overline{x}, \overline{\mathsf{e}}) \rbrack\!\rbrack$ for all classes. Division by zero is undefined for degenerate intervals $[0, 0]$. Otherwise, the union over partial behaviors can ignore "undefined" entries. If a division by zero would occur in the original problem, the causing operator is not applicable. Therefore, actual division by zero neither occurs in the relaxation.

Theorem 1. *The partial behaviors* $\mathcal{T}_\circ(x,e)$ *equal* $\mathrm{succ}_\circ(x,e)$ *for* $x \subseteq \tilde{x} \in \mathcal{B}_x$ *and* $e \subseteq \tilde{e} \in \mathcal{B}_e$. □

Theorem 1 is shown exemplarily for "×=", $\tilde{x} = (0,\infty), \tilde{e} = (0,1)$ and for "÷=", $\tilde{x} = (-\infty,0)$ and $\tilde{e} = (-\infty,-1)$ in the workshop version of this paper [1]. For each $q \in \mathcal{T}_\circ(x,e)$, we obtain a number of iterated assignments sufficient to reach q as a byproduct of the proof.

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 succeeding interval according to the semantics.

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

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

Example 2. Let $o = \langle \emptyset \to \{\mathtt{x} \times{=} \mathtt{e}\}\rangle$ have an effect on $\mathtt{x}$ in a state with $x = [1,4]$ and $e = [-\frac{1}{2},2]$. The partial behaviors 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 finally $\mathrm{succ}_\times(x,(1,2]) = [1,\infty)$. However, the union $\bigcup_{\tilde{x}\in\mathcal{B}_x,\tilde{e}\in\mathcal{B}_e} \mathrm{succ}_\times(x\cap\tilde{x}, e\cap\tilde{e})$ $= [-2,\infty)$, differs from $\mathrm{succ}_\times(x,e) = (-\infty,\infty)$.

However, the number of behavior classes is restricted, and therefore, new classes can only be hit a restricted number of times. We correct the hypothesis by including the partial behaviors $\mathcal{T}_\circ(x,e)$ of the classes hit 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, newly attained behavior classes become part of the decomposition in the next iteration.

Example 3. Recall Example 2 starting with $x_0 = x = [1,4]$ where the successor $\mathrm{succ}_\times(x_0\cap\tilde{x}, e\cap\tilde{e})$ equals $[-2,\infty)$. The decomposition over the newly achieved behavior classes with $x_1 = [-2,\infty)$ and $e = [-\frac{1}{2},2]$ contains among others the successor $\mathrm{succ}_\times([-2,0),(1,2]) = (-\infty,0)$. The union still contains partial behaviors that set the upper bound to ∞, so $\mathrm{succ}_\times(x_1\cap\tilde{x}, e\cap\tilde{e}) = (-\infty,\infty)$. Now, a fix-point is reached and $\widetilde{\mathrm{succ}}_\circ(x,e) = \mathrm{succ}_\circ(x,e)$.

Lemma 1. *The sequence of* x_j *converges to* $\widetilde{\mathrm{succ}}_\circ(x,e)$ *after at most* 3 *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 $e \subseteq \tilde{e} = (-\infty,-1)$. As e remains fixed x can change at most 3 times. □

We reformulate the feasibility of the decomposition to the following Theorem:

Theorem 2. *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. To see that $\widetilde{\mathrm{succ}}_\circ(x,e) \subseteq \mathrm{succ}_\circ(x,e)$ consider the first iteration determining $\widetilde{\mathrm{succ}}_\circ(x,e)$: all partial behaviors $\mathrm{succ}_\circ(x \cap \tilde{x}, e \cap \tilde{e})$ are operations on subsets of x and e. As we use the convex union for effects, $\mathrm{succ}_\circ(x,e)$ is monotone in both arguments i.e. $x_1 \subseteq x_2 \wedge e_1 \subseteq e_2 \Rightarrow \mathrm{succ}_\circ(x_1,e_1) \subseteq \mathrm{succ}_\circ(x_2,e_2)$. During each iteration determining $\widetilde{\mathrm{succ}}_\circ(x,e)$, 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, $q \notin x_0$. Let $x_0, x_1, \ldots$ be the sequence of intervals defining $\mathrm{succ}_\circ(x,e)$. There has to be a $k > 0$ with $x_{k+1} = x_k \sqcup (x_k \circ e)$ 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 successor is 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 1). 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 semantics 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.

Theorem 3. *The parallel fix-point algorithm for repetition relaxed planning is sound, 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 semantics 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 bold gray) for multiplicative effects that contract x and flip the arithmetic sign at the same time. The same is true for assignment effects $\mathcal{T}_\cdot(x,e) = (\!|\min(\underline{x}, \underline{e}), \max(\overline{x}, \overline{e})|\!)$. 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 4. Let $o = \langle \emptyset \to \{\mathsf{x} \mathrel{\times}= \mathsf{e}\}\rangle$ in a state with $x = [-1, -1]$, $\mathsf{e} = -\frac{\mathsf{x}+1}{2}$ and goal $\mathcal{G} = \{\mathsf{x} \geq 1\}$. Applying the operator arbitrarily often according to the repetition semantics yields the progression for k operator applications depicted to the right. 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.

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	$[-1, 0.875]$	$[-0.9375, 0]$
	$\vdots$	$\vdots$

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 4 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 restrict the problem to planning tasks where the aforementioned problem does not occur. The problem in Example 4 is that e depends on x. Thus, we restrict planning tasks to contain only effects where the assigned expression is independent from the affected variable. We 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*. A planning task with acyclic* direct dependency *relation is an* acyclic dependency *task.*

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

Theorem 4. *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 that 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 3 times, if we also consider the variable updates of the nested fix-point iteration from Lemma 1).

The number of *phases* is restricted, with the same argument as for the fix-point algorithm in the classical case. Preconditions cannot be invalidated and during each phase at least one previously inapplicable operator must become applicable. The number of phases is therefore restricted to the number of operators in the planning task. □

Theorem 5. *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. All operators are applied as soon as they are applicable, so a satisfiable condition must have been unsatisfied. For propositional conditions, this is impossible, as $s^\#(\mathsf{v_p}) \vDash \mathsf{v_p}$ if $\mathsf{v_p} \in \{\mathit{true}, \mathit{both}\}$ and no effect can set propositional variables to *false*. Therefore, a satisfiable numeric constraint was not achieved by the algorithm. This implies that an effect $(\mathsf{v_n} \circ\!= \mathsf{e})$ would have been able to assign a value to a variable that was not reached by our algorithm. Therefore, the successor defined by the semantics $\mathrm{succ}_\circ(s^\#(\mathsf{v_n}), s^\#(\mathsf{e}))$ 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 2, a contradiction. □

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 6. *Let $\langle o_1, o_2, \ldots, o_n\rangle$ be a repetition relaxed plan. Then, there exist $k_1, \ldots, k_n \in \mathbb{N}^+$ such that $\langle o_1^{k_1}, o_2^{k_2}, \ldots, o_n^{k_n}\rangle$ is an interval relaxed plan, where $o_i^{k_i}$ denotes a sequence of k_i repetitions of operator o_i.* □

A proof of Theorem 6 can be found in the workshop version of this paper [1]. The idea is to explicate the number of operator applications from a plan obtained by the fix-point algorithm for repetition relaxed planning. For each numeric constraint we determine a target value in the open intervals, which is sufficient to satisfy that constraint. The number of required operator applications k_i to reach the target value is obtained from the proof of Theorem 1.

Example 5. 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 $\mathsf{x}{+}\mathsf{y} > \mathsf{z}$. From $\mathsf{e_a} = \mathsf{x}{+}\mathsf{y} \mapsto [0, 2)$ and $\mathsf{e_b} = \mathsf{z}$ we choose an arbitrary $q_a = 1.9 \in s^\#(\mathsf{e_a})$ and an arbitrary $q_b = 1.8 \in s^\#(\mathsf{e_b})$ from within the expression intervals so that the constraint is satisfied. 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 to the target values of the sub-expressions. We could continue with target values 0.95 for x and y each. Let $x = [0, 1)$ be obtained from a repetition relaxed operator $o_1 = \langle \emptyset \to \{\mathsf{x} := (\mathsf{a} + 1)\}\rangle$, $a = [-1, -1]$, which induces a target value of -0.05 for a. Let now $o_2 = \langle \emptyset \to \{\mathsf{a} \div\!= 2\}\rangle$ be the operator that manipulated a in the repetition relaxed plan. The explicated number of operator applications for o_2 is obtained by solving $-0.05 = -1 \div 2^k$ so $k \approx 4.3$ and o_2 has to be applied 5 times.

Theorem 7. *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 sound (Theorem 3), complete (Theorem 5) and terminates in polynomial time (Theorem 4). Thus, generating a repetition relaxed plan $\pi^\#$ is in P. An interval relaxed plan π^+ can be constructed from $\pi^\#$ (Theorem 6) in polynomial time. □

The definition of a relaxation is *adequate* [8] 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 8. *The* interval relaxation *is adequate for* acyclic dependency *tasks.*

Proof. Admissibility: After each step of the original plan π, the propositional variables are either equal in the relaxed and in the original state, or they assign to *both*, which cannot invalidate any (goal or operator) conditions. For numeric variables, the value of the original task is contained in the mapped interval. Admissibility follows from the semantics of comparison constraints that hold if they do for any pair of elements from the two intervals. *Basic informedness:* No (goal or operator) conditions are dropped from the task. Relaxed numeric variables are mapped to degenerate intervals that 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 7, we can also conclude that interval relaxed numeric plan existence is in P for tasks with *acyclic dependencies*. □

5 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 proposed relaxation advances the state of the art even though adequacy of interval relaxation was only shown for the restricted set of *acyclic dependency* tasks. However, the requirement of acyclic dependency for numeric expressions is a proper generalization of expressions e being required to be constant, a requirement for other state-of-the-art approaches, e.g. [8], which is met in many practically relevant problems. The complexity of the approach for arbitrary interval relaxed planning problems remains an open research issue, though. It is imaginable that the fixpoint reached by arbitrary operator repetitions can be found in polynomial time.

In the future, we intend to adapt the well-known heuristics from classical planning, $h_{\max}$, h_{add} and h_{FF}, to the interval relaxation framework.

Acknowledgements: This work was supported by the DFG grant EXC1086 BrainLinks-BrainTools and the DFG grant SFB/TR 14 AVACS to the University of Freiburg, Germany.

References

1. Aldinger, J., Mattmüller, R., Göbelbecker, M.: Complexity Issues of Interval Relaxed Numeric Planning. In: Proceedings of the ICAPS-15 Workshop on Heuristics and Search for Domain-independent Planning (HSDIP 2015). pp. 4–12 (2015)
2. Bäckström, C., Nebel, B.: Complexity results for $\mathtt{SAS}^+$ planning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI 1993). pp. 1430–1435 (1993)
3. Bofill, M., Arxer, J.E., Villaret, M.: The RANTANPLAN Planner: System Description. In: Proceedings of the ICAPS-15 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAPS 2015). pp. 1–10 (2015)
4. Bylander, T.: The Computational Complexity of Propositional `STRIPS` Planning. Artificial Intelligence 69 pp. 165–204 (1994)
5. Coles, A., Fox, M., Long, D., Smith, A.: 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). pp. 52–59 (2008)
6. Fox, M., Long, D.: `PDDL2.1` : An Extension to `PDDL` for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research 20 (JAIR) pp. 61–124 (2003)
7. Helmert, M.: 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). pp. 303–312 (2002)
8. Hoffmann, J.: The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables. Journal of Artificial Intelligence Research 20 (JAIR) pp. 291–341 (2003)
9. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. Society for Industrial and Applied Mathematics (2009)
10. Young, R.C.: The Algebra of Many-valued Quantities. Mathematische Annalen 104 pp. 260–290 (1931)