ECAI 2025
I. Lynce et al. (Eds.)

© 2025 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA251398

Simple Numeric Planning with Two Variables Is Decidable

Hayyan Helal[a,*] **and Gerhard Lakemeyer**[a]

[a]RWTH Aachen University, Germany

Abstract. It is known that a simple numeric planning problem (SNP) with one numeric variable is decidable but undecidable with three (Helmert 2002). A more recent result (Gnad et. al 2023) showed undecidability for two numeric and one propositional variable. In this paper, we show the decidability of SNP with exactly two numeric variables. For this, we first partition the state space into a finite number of regions and demonstrate the decidability of SNP when restricted to any of these regions. Afterwards, we develop a correct search algorithm that abstracts from these regions by tracking an infinite number of states following an arithmetic progression pattern. Finally, we prove termination of the search and draw conclusions about the reasons for undecidability for general SNP.

1 Introduction

Although the term "planning" was coined to highlight the application, planning, in its classical sense, is concerned with solving a reachability problem over a (possibly infinite) state space and shares key mathematical properties with other reachability and vector-addition problems. In simple numeric planning (SNP) with exactly n numeric variables, states are n-dimensional vectors, effects are vector additions, and preconditions form convex n-dimensional polytopes[1] within that state (vector) space. In other words, any reachable state must be the addition of a (positive) linear combination of the effect vectors of all actions to the initial (vector) state.

We will first study a restricted domain of SNP, where all actions share the same preconditions. By abstracting from the differences in preconditions, we define four types of actions, depending on the direction of the effect vector. Each such type has a specific functionality in our analysis. We will show that the reachable states at which no more actions are applicable with this domain can be partitioned into a finite number of arithmetic progressions (APs), i.e., we can find a finite representation for a (possibly) infinite number of states.

Afterwards, we will partition the state space into a finite number of polygonal convex regions. To do that, we will view the preconditions as lines that halve the state space, and consider the partitions produced by all such lines. Each of the resulting regions could be analyzed as an instance of the restricted domain, i.e., we can define the same four action types for all actions within each region. Similar to the restricted domain, the states that are reachable from a region but not in it could be partitioned into a finite number of APs in certain cases. In other words, we can find a finite representation for all possible ways of exiting any of the regions.

Having done that, we can prove correctness for an algorithm called Arithmetic Progression Mining (APM), which searches over APs instead of single states. In other words, APM will eventually find any reachable goal state. Finally, we discuss pruning methods and show that in any case, even if the goal is not reachable, APM will terminate for any 2-variable SNP.

2 Related Work

Numeric variables allow for an infinite state space, which generally implies undecidability for the reachability problem. Helmert [6] showed that even very restricted planning problems involving numeric variables with additive constant effects and preconditions given by linear comparisons against zero can simulate Abacus programs with a numeric variable for each register. Abacus programs with two registers were already known for their ability to simulate Turing machines [2]. Hence, numeric planning with two numeric and an arbitrary number of propositional variables is undecidable. In that case, the propositional variables are required to store the current command number during the run of an Abacus program, and their number grows w.r.t. the program length.

Later, restricted tasks (RT) [8], simple numeric planning (SNP) [10], and numeric additive planning (NAP) [5] were defined. These are all equivalent but introduce further restrictions to simplify the problem for theoretical studies. After Gnad et. al [4] showed that two numeric and a single propositional variable are enough to make the planning problem undecidable, it became clear that the two-variable case is on the boundary of decidability, potentially very hard or even undecidable.

Concerning the approach, we will be using ideas similar to causal graphs, which were introduced by [9, 1, 3, 7], and adapted later for numeric planning, either by restricting the dependencies between variables [11], or between actions [5]. Specifically, Helal and Lakemeyer [5] coined the term maintainable plans, which will be equivalent to the term "repetition" in our analysis.

3 Preliminaries

For integer numeric variables $V := \{x_1, ..., x_n\}$, we will use the n-dimensional integer vector space $\mathbb{Z}^n$ as a state space. For any such vector, usually denoted $s \in \mathbb{Z}^n$ for state, we assume that $s[x_i]$ is the i'th element of the vector. When dealing with two-dimensional state spaces, we will always set $V := \{x, y\}$, and for any state $s \in \mathbb{Z}^2$, we assume that $s = \langle s[x], s[y] \rangle$.

* Corresponding Author. Email: helal@kbsg.rwth-aachen.de
[1] Polytopes are the n-dimensional generalization of the two-dimensional polygons and three-dimensional polyhedrons.

3.1 Linear Constraints

Definition 1. *For a set of variables $V := \{x_1, ..., x_n\}$, we define the set of linear constraints over V by $LC(V) := \{c = \langle f_c, h_c\rangle : f_c \in \mathbb{Z}^n, h_c \in \mathbb{Z}\}$. We use the notation $c = [f_c \cdot s \geq h_c]$, where s is assumed to be an n-dimensional integer variable vector, and $f_c \cdot s = \sum_{x \in V} f_c[x]s[x]$ denotes the inner product.*

We define additionally the function $R : LC(V) \to 2^{\mathbb{Z}^n}$ that maps each linear constraint to the state space subset where it is satisfied, i.e., for all $c \in LC(V)$, $R(c) := \{s \in \mathbb{Z}^n : f_c \cdot s \geq h_c\}$.

We extend the definition of R to sets of constraints $C \subseteq LC(V)$ by $R(C) := \bigcap_{c \in C} R(c)$. Notice that for any finite $C \subset LC(V)$, $R(C)$ is a polygonal convex region. It is known from integer linear programming that such linear constraints are closed under negation. Specifically, since only integers are considered, the negation of a linear constraint $c \in LC(V)$ is another linear constraint $\bar{c} \in LC(V)$ defined by $\bar{c} := [f_c \cdot s < h_c] = [(-f_c) \cdot s \geq -h_c + 1] = \langle -f_c, -h_c + 1\rangle$. Finally, we say that two constraints c_1, c_2 are parallel iff there exists $p \in \mathbb{Q} \setminus \{0\}$ s.t. $f_{c_1} = pf_{c_2}$. E.g., for any $c \in LC(V)$, c and $\bar{c}$ are parallel with $f_c = -f_{\bar{c}}$.

3.2 Simple Numeric Planning

The classical definition of simple numeric planning involves both numeric and propositional variables [10]. However, propositional variables can be compiled into numeric variables without causing exponential blowup [5]. Additionally, since our goal in this paper is to prove the decidability of simple numeric planning with two numeric variables and no propositional variables, we do not introduce propositional variables in our definition.

Definition 2. *A simple numeric planning problem (*SNP*) is defined by a tuple (A, V, s_0, G) with $n := |V|$ representing the dimension of the state space $\mathbb{Z}^n$, and A is a finite set of actions given as tuples $a = (\text{pre}_a, \text{eff}_a)$ where $\text{pre}_a \subset LC(V)$ defines a finite set of preconditions for the applicability of a, and $\text{eff}_a : V \to \mathbb{Z}$ defines the (constant) integer value that will be added to each variable when a is applied. Finally, the state $s_0 \in \mathbb{Z}^n$ is an initial state, and $G \subset LC(V)$ is a finite set of goal conditions.*[2]

To ease the notation, we will define for each action $a \in A$ an effect vector $v_a := \langle \text{eff}_a(x_1), ..., \text{eff}_a(x_n)\rangle$. With that: (1) a can be applied at a state $s \in \mathbb{Z}^n$ by adding its effect vector, i.e., $s + v_a$, (2) a state s satisfies the preconditions of a iff $s \in R(\text{pre}_a)$, denoted $s \vDash a$. To conclude the introduction of SNP domains, we define $C_A := \bigcup_{a \in A} \text{pre}_a$ as the set of all linear constraints in a domain.

We will deal with plans using words over the action set A. Let us extend the effect vectors to plans by $v_\varepsilon := \langle 0, .., 0\rangle \in \mathbb{Z}^n$ and $v_{wa} := v_w + v_a$ for $w \in A^*, a \in A$. We can also extend the definition of state satisfaction to plans recursively by assuming the global validity of the empty word, i.e. $s \vDash \varepsilon$ for all $s \in \mathbb{Z}^n$, and define $s \vDash wa$ iff $s \vDash w$ and $s + v_w \vDash a$.

3.3 Reachability

To study reachability, it is important to define paths and the plans corresponding to them first.

[2] Notice that SNP can be equivalently defined over rationals $\mathbb{Q}$. However, a multiplication with the least common multiplier of all denominators allows us to define an equivalent integer-valued SNP for any rational domain [5]

Definition 3. *We define a path P as a sequence of states $(s_0, ..., s_{|P|})$ s.t. for each $i \in \{1, ..., |P|\}$, there exists an action $a_i \in A$ s.t. $s_{i-1} \vDash a_i$ and $s_i = s_{i-1} + v_{a_i}$. We call the word $A(P) := a_1 a_2 ... a_{|P|} \in A^*$ the plan corresponding to P, and say that P is the path that follows the plan $A(P)$ from s_0.*

We say that there exists a path between two states s_1, s_2, if there exists a path $P := (t_0, ..., t_{|P|})$ s.t. $t_0 = s_1$ and $t_{|P|} = s_2$. The notation $P[s_1, s_2]$ denotes a shortest path between s_1 and s_2 if one exists. Otherwise, it is not defined.

Definition 4. *Given an initial state $s_0 \in \mathbb{Z}^n$, the set of reachable states from s_0 is defined as*

$$\mathbf{Reach}(s_0) := \{s \in \mathbb{Z}^n \text{ s.t. } P[s_0, s] \text{ exists}\}$$

We will also need a restricted form of reachability for our analysis. Given a region $R \subseteq \mathbb{Z}^n$, if $s_0 \in R$, we define $\mathbf{Reach}_R(s_0)$ as the set of all states reachable by paths $P = (s_0, ..., s_{|P|})$ with all action applications happening in R, i.e., $s_i \in R$ for all $i \in \{0, ..., |P| - 1\}$. If $s_0 \notin R$, then, $\mathbf{Reach}_R(s_0) := \{s_0\}$.

This concludes the introduction of SNP as a problem. Specifically, an SNP (A, V, s_0, G) is solvable iff $\mathbf{Reach}(s_0) \cap R(G)$ is non-empty.

4 A Restricted Domain

We will first tackle a restricted version of a 2-variable SNP $(A, \{x, y\}, s_0, G)$, where all actions are applicable at exactly one region $\mathbb{N}^2 = R(\{[x \geq 0], [y \geq 0]\})$. In other words, we assume that $\text{pre}_a = C_A = \{[x \geq 0], [y \geq 0]\}$ for all $a \in A$. Notice that $\mathbf{Reach}(s_0) = \mathbf{Reach}_{\mathbb{N}^2}(s_0)$ in this case. We will show later that the state space can be partitioned into a finite number of subsets, where reachability, when restricted to any of these subsets, behaves similarly to this restricted version.

4.1 Action Functionality

Let us first study the effect vectors of an action in the restricted domain. We will say that two actions $a, b \in A$ are parallel iff their effect vectors are, i.e., $v_a = pv_b$ for some $p \in \mathbb{Q} \setminus \{0\}$. We will define four action types based on their effect vector. We always assume that no action $a \in A$ has a zero effect vector, i.e., $v_a[x] = v_a[y] = 0$, as these are useless actions.

Definition 5. *For an action $a \in A$, we define the types:*

	$v_a[x]$	$v_a[y]$
a is a repetition	≥ 0	≥ 0
a is a co-repetition	≤ 0	≤ 0
a is a y-crossing	≥ 0	< 0
a is an x-crossing	< 0	≥ 0

We call a a crossing if it is an x-crossing or a y-crossing.

In Fig. 1, both a, b are y-crossings but b' is an x-crossing.

To study each action type, let us fix a state $s \in \mathbb{N}^2$: (1) For a repetition $r \in A$, we have $s \vDash r^i$ for all $i \in \mathbb{N}$, where r^i denotes the plan with r repeated i times, i.e., $r^0 := \varepsilon$ and $r^i = r^{i-1}r$. In other words, a can be "repeated" arbitrarily often from any state in $\mathbb{N}^2$. (2) A co-repetition $r' \in A$ behaves similarly when co-reachability is considered, i.e., $s - iv_{r'} \vDash r'$ for all $i \in \mathbb{N}$. (3) On the other hand, for a y-crossing $a \in A$, we get $(s + v_a)[y] < s[y]$, i.e., by applying

a enough times, the constraint $[y \geq 0]$ will be eventually "crossed". In other words, $i \in \mathbb{N}$ exists s.t. $s \vDash a^i$ and $(s + a^i)[y] < 0$. (4) Similarly, x-crossings eventually lead to crossing $[x \geq 0]$.

The type of any constructed plan $w \in A^*$ can be determined by considering the overall effect vector of that plan $v_w \in \mathbb{Z}^2$. Therefore, we can infer the existence of new types by analyzing the available combinations of action types, as we will show in the next lemma.

Lemma 1. *If x- and y-crossings that are unparallel to each other exist, then a repetition or a co-repetition with a non-zero effect vector can be constructed.*

Proof. Let a be a y-crossing and b an x-crossing. We construct a plan $w := a^{|v_b[y]|}b^{|v_a[y]|}$. The effect vector of w is

$$v_w = |v_b[y]|v_a + |v_a[y]|v_b = v_b[y]v_a - v_a[y]v_b$$

Therefore, $v_w[y] = 0$. If $v_w[x] > 0$, w is a repetition. If $v_w[x] < 0$, w is a co-repetition. If $v_w[x] = 0$, then, a is parallel to b. □

In Fig. 1, since a is a y-crossing and b' is an x-crossing, and a is not parallel to b', we can construct a repetition $w := aab' \in \{a, b'\}^*$. Notice that the direction of the effect vector of $w = aab'$ is positive, i.e., $(s + v_w)[x] \geq s[x]$ and $(s + v_w)[y] \geq s[y]$ for all $s \in \mathbb{Z}^2$. If w is valid from s, it must be valid from $s + iv_w$ for all $i \in \mathbb{N}$. On the other hand, any combination of b, b' has either a zero effect vector or is a crossing because they are parallel.

We will show later that the existence of a repetition or co-repetition ensures a significant simplification of the problem. Therefore, we can focus for now on action sets containing crossings with a similar direction only, w.l.o.g. y-crossings.

4.2 Projection

Let us first measure the maximal number of valid applications of an action from a state s, which is equivalent to the minimal number of actions applications from s after which a is no longer applicable.

Definition 6. *Let $a \in A$ and $s \in \mathbb{N}^2$. We define the number of valid applications of a from s as*

$$\#_s[a] := \min\{i \in \mathbb{N} \cup \{\infty\} : s + iv_a \notin \mathbb{N}^2\}$$

With this definition, we can see that for all actions $a \in A$ and states $s \in \mathbb{Z}^2$, we have:

- If $s \notin \mathbb{N}^2$, $\#_s[a] = 0$.
- Else if a is a repetition, then, $\#_s[a] = \infty$.
- Else if a is a y-crossing, then, $\#_s[a] = \left\lceil \frac{s[y]+1}{-v_a[y]} \right\rceil$.

Similarly, for an x-crossing with x instead of y. With that, we can define projection formally.

Definition 7. *The projection of a crossing $a \in A$ from $s \in \mathbb{N}^2$ is defined by $\mathbf{Proj}(s; a) := s + \#_s[a]v_a$. The path $P[s, \mathbf{Proj}(s; a)] = (s, s + v_a, s + 2v_a, ..., s + \#_s[a]v_a)$ is called a direct path.*

In Fig. 1, the red vectors denoting the path $P[\langle 0, 4\rangle, \langle 15, -1\rangle]$ is a direct path from $s := \langle 0, 4\rangle$ using the action a. Notice that $\mathbf{Proj}(s; a) = \langle 15, -1\rangle$ is the projection of s using a.

As said before, applying a y- crossing often enough, leads to crossing the linear constraint $[y \geq 0]$. We denote the state where this crossing happens as a y-exit. Formally, the y-exits are defined by:

$$y\text{-}\mathbf{EX} := \bigcup_{s \in \mathbb{N}^2} \Big(\mathbf{Reach}(s) \setminus R([y \geq 0])\Big)$$

Let $\mu := -\min_{a \in A} v_a[y]$ represent the maximal negative effect on y. Notice that for any state $s \in y\text{-}\mathbf{EX}$, there exists a state $t \in \mathbb{N}^2$ and $a \in A$ s.t. $s + v_a = t$, i.e., $(s - t)[y] \leq \mu$. Therefore, we can partition the exits into a finite number of lines by $y\text{-}\mathbf{EX} = \bigcup_{i=1}^{\mu} L_i$, where $L_i := \{s \in y\text{-}\mathbf{EX} : s[y] = -i\}$. We call $L_1, ..., L_\mu$ the y-exiting lines. We will show in the next subsection that the reachable states in y-$\mathbf{EX}$ can be represented with a finite number of arithmetic progressions.

4.3 Arithmetic Progressions

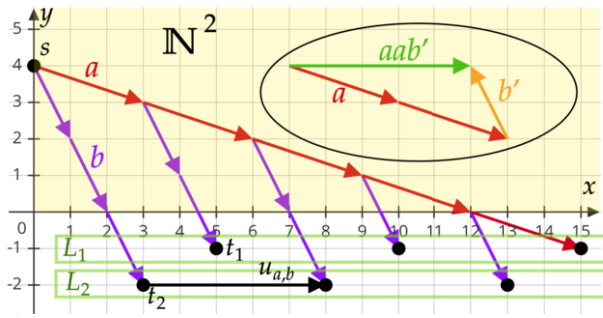


Figure 1. Example of arithmetic progressions $\mathbf{AP}(t_i, u_{a,b}, 2); i \in \{1, 2\}$ (black) that are produced by two y-crossings a, b (red and blue) from a state $s := \langle 0, 4\rangle$ within $\mathbb{N}^2$ (light yellow). One can also see L_1, L_2 (green), the y-exiting lines. In the ellipse, a repetition aab' (green) can be constructed from a y-crossing a (red) and an unparallel x-crossing b' (orange).

An arithmetic progression has a structure similar to that of a for-loop, e.g., $\mathbf{AP}(i, k, n) := (i, i + k, i + 2k, ..., i + nk)$ for $i, k \in \mathbb{Z}$ and $n \in \mathbb{N}$. We will extend this idea to 2D and allow for infinite arithmetic progressions.

Definition 8. *For a starting state $s \in \mathbb{Z}^2$, an update vector $v \in \mathbb{Z}^2$, and a number of repetitions $n \in \mathbb{N} \cup \{\infty\}$, an arithmetic progression (AP) is defined by $\mathbf{AP}(s, v, n) := \{s + iv : i \in \{0, ..., n\}\}$.*

We will extend the definition of projection to APs as well by:

$$\mathbf{Proj}(s, v, n; a) := \{\mathbf{Proj}(s + iv; a) : i \in \{0, ..., n\}\}$$

We are now able to prove our first theorem, which states that a finite number of arithmetic progressions can represent an infinite number of projections from states that follow an arithmetic progression.

Theorem 2. *For a y-crossing $a \in A$ and an AP $\mathbf{AP}(s, v, n) \subseteq \mathbb{N}^2$, there exist $u \in \mathbb{Z}^2$, $k \in \mathbb{N}$, and $t_i \in y\text{-}\mathbf{EX}; i \in \{1, ..., k\}$ s.t.:*

$$\mathbf{Proj}(s, v, n; a) = \bigcup_{i=1}^{k} \mathbf{AP}(t_i, u, n_i)$$

with $k = \frac{|v_a[y]|}{d}$, $d = \gcd(|v_a[y]|, |v[y]|)$, and $\left\lfloor \frac{n}{k} \right\rfloor \leq n_i \leq \left\lceil \frac{n}{k} \right\rceil$.

The theorem follows from known principles in number theory, such as the extended Euclidean algorithm and Bézout pairs.

Consider Fig. 1 and notice that the states reachable by a combination of a, b on any exiting line are exactly those reachable from some intermediate state of the direct path $P[s, \mathbf{Proj}(s; a)]$, i.e. $(s, s + v_a, ..., v + \#_s[a]v_a)$, which follow an arithmetic progression $\mathbf{AP}(s, v_a, \#_s[a])$.

We can generalize this method for any two actions a, b, of which one (w.l.o.g. b) is a y-crossing. We first consider the AP representing

the states visited on a direct path followed when a is used, then apply b from each state in that AP. In other words, we can use the last theorem to show that $\mathbf{Proj}(s, v_a, \#_s[a]; b) \cap L$, the reachable states on a y-exiting line L by using the actions a, b, can also be partitioned into a finite number of APs. In general, we define the projection vector of two unparallel actions $a, b \in A$ by:

$$u_{a,b} := \left\langle \frac{|v_a[x]v_b[y] - v_a[y]v_b[x]|}{\gcd(|v_a[y]|, |v_b[y]|)}, 0 \right\rangle$$

which is parallel to any y-exiting line L. By Th. 2, we can find $t \in L$ and $n \in \mathbb{N} \cup \{\infty\}$ s.t. $\mathbf{Reach}(s) \cap L = AP(t, u_{a,b}, n)$. If one of the actions, w.l.o.g. a, is a repetition, applying a arbitrarily often never leaves $\mathbb{N}^2$, we get an infinite AP. Projecting from that AP using a y-crossing b, i.e., $\mathbf{Proj}(s, v_a, \infty; b)$, we get infinite APs within each exiting line, i.e., $n = \infty$. This will be crucial to find a finite representation for an infinite number of exits, as we will show later.

In Fig. 1, notice that $u_{a,b} = u_{a,b'} = \langle 5, 0 \rangle$ which are parallel to the y-exiting lines. The reason for the equality $u_{a,b} = u_{a,b'} = v_{aab'}$ is that $v_{b'} = -v_b$.

By induction, we can extend this idea to any number of actions and prove that in general $\mathbf{Reach}(s) \cap y\text{-}\mathbf{EX}$ can be partitioned into a finite number of APs. With that, we can prove the following result.

Corollary 3. *Restricted simple numeric planning is decidable.*

Proof. Given an initial state s_0 and goal conditions G. Assume first that $R(G) \subseteq \mathbb{N}^2$, then, we can solve the integer linear program with positive integer variables $i_a \in \mathbb{N}$ and the constraints $0 \leq \sum_{a \in A} i_a v_a \in R(G)$. Remember that G is a finite set of linear constraints. Otherwise, if $R(G)$ intersects with some exiting line, we can use the methods considered before. All other states are not reachable as there are no actions available outside of $\mathbb{N}^2$. □

For our algorithm, later, it is important to extend the definition of projection to states reachable with any number of actions from $\mathbf{AP}(s, v, n)$ with $s, v \in \mathbb{Z}^2$ and $n \in \mathbb{N}$ by:

$$\mathbf{Proj}_y(s, v, n) := \bigcup_{i=0}^{n} \Big(\mathbf{Reach}(s + iv) \cap y\text{-}\mathbf{EX} \Big)$$

As shown before, a direct path corresponds to an AP, i.e., we can prove that $\mathbf{Proj}_y(s, v, n)$ can be partitioned into a finite number of APs each being contained within some y-exiting line. The same argument works for $\mathbf{Proj}_x(s, v, n)$. In other words, given an AP $\mathbf{AP}(s, v, n)$, we can, with one step, compute both $\mathbf{Proj}_x(s, v, n)$ and $\mathbf{Proj}_y(s, v, n)$, which determine all the reachable exits of $\mathbb{N}^2$ from any state $t \in \mathbf{AP}(s, v, n)$.

5 Encoding the State Space

In this section, we will define a partition of the state space into a finite number of regions where SNP behaves similarly to the restricted domain. For that, we will construct an encoding of the states within each region.

Definition 9. *For $s \in \mathbb{Z}^2$ and a linear constraint $c \in LC(\{x, y\})$, we define the integer distance between s and c as*

$$d[s, c] := f_c \cdot s - h_c = f_c[x]s[x] + f_c[y]s[y] - h_c \in \mathbb{Z}$$

The goal is to normalize the relation between states and linear constraints. For example, notice that $d[s, c] \geq 0$ iff $s \in R(c)$. We can also use it to pick certain lines within the region $R(c)$.

Definition 10. *For a finite set of linear constraints $C \subset LC(\{x, y\})$ and a linear constraint $c \in C$, we define $E^c_{R(C)} := \{s \in R(C) : d[s, c] = 0\}$ as an edge of the region $R(C)$.*

We say that two edges $E^{c_1}_R, E^{c_2}_R$ of some region R are parallel iff c_1, c_2 are parallel. We assume that all considered sets of constraints C are minimal, i.e., $R(C) \subsetneq R(C \setminus \{c\})$ for all constraints $c \in C$.

Notice that in Fig. 2, the region $R := R(\{c_1, c_2, c_3\})$ has two infinite edges $E^{c_1}_R, E^{c_2}_R$, and that $R' := R(\{c_1, c_2, c'_3\}) = R(\{c_1, c_2\})$ because c_2, c'_3 are parallel to each other. This can be generalized for any polygonal infinite convex region of $\mathbb{Z}^2$.

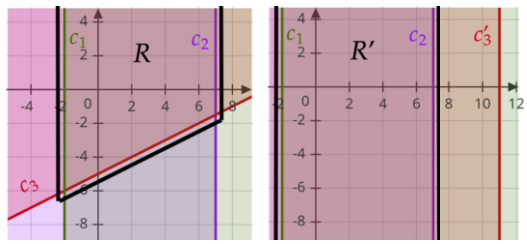


Figure 2. Shown are the linear constraints $c_1 := [x \geq -2]$ (green), $c_2 := [x \leq 7]$ (blue), $c_3 := [x - 2y \leq 10]$ (red on left), $c'_3 := [x \leq 11]$ (red on left), and the edges (bold black) of the regions $R := R(\{c_1, c_2, c_3\})$ (left) and $R' := R(\{c_1, c_2, c'_3\})$ (right).

Lemma 4. *For a finite set of linear constraints $C \subset LC(V)$ with $|V| = 2$, $R(C)$ has at most two infinite edges.*

5.1 Precondition-Free Regions

We will describe a partition of the state space into "precondition-free" regions, and show afterwards that each one could be analyzed similarly to the restricted domain.

Definition 11. *For a function $\gamma : C_A \to \{0, 1\}$, we define:*

$$C_\gamma := \{c_i : \gamma(c_i) = 1\} \cup \{\overline{c_i} : \gamma(c_i) = 0\}$$

*A precondition-free region (*PFR*) is defined by $R_\gamma := R(C_\gamma)$.*

Consider the following example with two actions $A = \{a, b\}$, which is shown in Fig. 3 and will serve as a running example for the algorithms of this paper.

- $\text{pre}_a = \{[x \leq 0]\}, \text{eff}_a(x) = 0, \text{eff}_a(y) = +1.$
- $\text{pre}_b = \{[y \geq 1]\}, \text{eff}_b(x) = +1, \text{eff}_b(y) = -1.$

Starting at the intitial state $s_0 := \langle 0, 0 \rangle$, all valid plans have the form $a^i b^j$ where $i \geq j \in \mathbb{N}$. Additionally, since $C_A = \{[x \leq 0], [y \geq 1]\}$ the state space $\mathbb{Z}^2$ is partitioned into 4 PFRs $R_\gamma; \gamma : C_A \to \{0, 1\}$, all infinite with two infinite edges. We denote the actions applicable in R_γ with $A_\gamma; \gamma : C_A \to \{0, 1\}$:

- $R_{11} := R(\{[x \leq 0], [y \geq 1]\})$ with $A_{11} = \{a, b\}$.
- $R_{10} := R(\{[x \leq 0], [y \leq 0]\})$ with $A_{10} = \{a\}$.
- $R_{01} := R(\{[x \geq 1], [y \geq 1]\})$ with $A_{01} = \{b\}$.
- $R_{00} := R(\{[x \geq 1], [y \leq 0]\})$ with $A_{00} = \emptyset$.

Definition 12. *For two states $s, s' \in R_\gamma$, we say that s' is regionally reachable from s iff $s' \in \mathbf{Reach}_\gamma(s) := \mathbf{Reach}_{R_\gamma}(s)$. We call $P[s, s']$ a regional path, and $A(P[s, s'])$ a regional plan in that case. We call the problem of finding out whether $s' \in \mathbf{Reach}_\gamma(s)$ "regional reachability".*

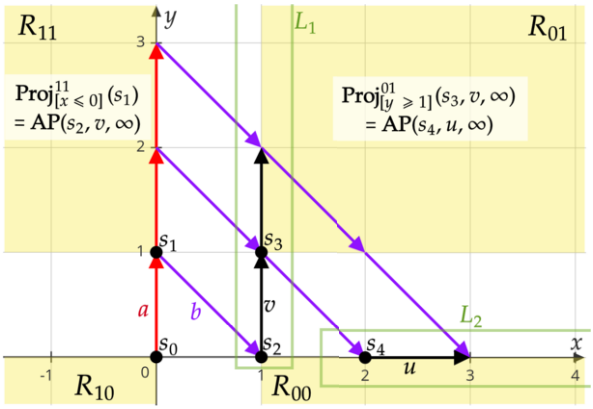


Figure 3. An example of the partitioning of the state space into 4 PFRs (light yellow). The exiting lines, L_1 of R_{11} and L_2 of R_{01}, are marked (green). The plans $a^i b^j; i \geq j \in \{0, .., 3\}$ starting at s_0 are drawn (red and blue). Finally, two examples of projection, from R_{11} to R_{01}, and from R_{01} to R_{00}, are shown (black).

The regions are called precondition free because for all actions $a \in A$, and all $\gamma : C_A \to \{0, 1\}$, if $s \vDash a$ for some $s \in R_\gamma$, then, $s \vDash a$ for all $s \in R_\gamma$. We denote the actions applicable at any state in R_γ with $A_\gamma := \{a \in A : \text{pre}_a \subseteq C_\gamma\}$. Remember that all PFRs must be convex polygons and thus have at most two infinite edges by Lemma 4. In order to make later proofs easier, we will assume that any infinite PFR (which we call IPFR) has exactly two infinite edges. If an IPFR has one infinite edge only, we can partition it into two IPFRs by adding a dummy constraint to produce an edge unparallel to that infinite edge. With that, both of the resulting IPFRs have exactly two infinite edges.

Having done that, notice that the number of IPFRs is $2|C_A|$. We order the IPFRs $R_0, ..., R_{2|C_A|-1}$ by starting at any one and going clockwise (the mathematical negative rotation in 2D, check Fig. 4). With that, the two neighbors of R_i are $R_{(i+1) \mod 2|C_A|}$ and $R_{(i-1) \mod 2|C_A|}$ for all $i \in \{0, ..., 2|C_A| - 1\}$. The respective actions would be $A_i = A_\gamma$ for $R_i = R_\gamma$.

Next, we will generalize all concepts defined in the restricted domain to any PFR. Finite PFRs include a finite number of states and can be explored with a finite number of steps. Therefore, we need to focus on IPFRs. We define $c_i^j; i \in \{1, 2\}$ s.t. $E_{R_j}^{c_i^j}$ is one of the infinite edges of R_j. To ensure that the enumeration of these two edges respects the order defined before, we assume that $\overline{c_2^j} = c_1^{(j+1) \mod 2|C_A|}$ for all $j \in \{0, ..., 2|C_A| - 1\}$ (Check Fig. 4 again). With that, we define an encoding function

$$e_j : R_j \to \mathbb{N}^2 : s \mapsto \langle d[s, c_1^j], d[s, c_2^j] \rangle$$

In the restricted domain, the infinite edges of $\mathbb{N}^2$ are the x and y axes, and $d[s, [x \geq 0]] = s[x]$, $d[s, [y \geq 0]] = s[y]$.

For an action $a \in A$, we can define the integer distance effect of a on a constraint c by $d[a, c] := d[s+a, c] - d[s, c] = f_c \cdot v_a$. Similar to the restricted domain, we say that $a \in A_j$ is a repetition/co-repetition iff $d[a, c_i^j] \geq 0 / d[a, c_i^j] \leq 0$ for all $i \in \{1, 2\}$, respectively. Additionally, we say that a is a positive crossing iff $d[a, c_1^j] \geq 0$ and $d[a, c_2^j] < 0$, denoting that applying a sufficiently often from R_j leads to $R_{j+1 \mod 2|C_A|}$. Similarly, a is a negative crossing iff $d[a, c_1^j] < 0$ and $d[a, c_2^j] \geq 0$. Positive and negative refer, therefore, to a direction of rotation.

We call an IPFR R_j conic iff it has unparallel infinite edges, i.e., c_1^j is unparallel to c_2^j, and non-conic otherwise (Check the non-conic IPFRs 2 and 6, and the conic ones $0, 1, 3, 4, 5, 7$ in Fig. 4). Notice

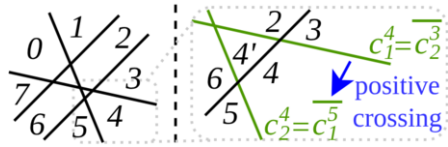


Figure 4. On left, an example of the partition of the state space by 4 linear constraints, and the enumeration of the resulting IPFRs is shown. On the right, we focus on the IPFR 4 and denote $4'$ the finite region where the exceptions to the restricted domain occur. We also show the infinite edges of region 4 (green) and the direction of a positive crossing (blue) applicable in region 4.

that our encoding fails for non-conic IPFRs R_j since $f_{c_i^j}; i \in \{1, 2\}$ would be insufficient to encode a (subset of a) 2D-vector space R_j, specifically, $f_{c_i^j}; i \in \{1, 2\}$ are not linearly independent. However, since each (infinite) exiting line can be represented by an infinite sequence of states $(s_l)_{l \in \mathbb{N}}$, we can define $\text{level}_L(s) := l$ iff $s = s_l$ in that sequence. If R_j is non-conic, it can be partitioned into a finite number of lines, i.e., $k \in \mathbb{N}$ exists s.t. $R_j := L_0 \cup ... \cup L_k$ where $L_i := \{s \in R_j : d[s, c_1^j] = i\}$ for all $i \in \{0, ..., k\}$. Therefore, we can define:

$$e_j : R_j \to \mathbb{N}^2 : s \mapsto \langle i := d[s, c_1^j], \text{level}_{L_i}(s) \rangle$$

5.2 *Exiting Lines*

To finalize a proof of decidability for regional reachability, we still need to analyze the exiting lines of the constructed IPFRs, which are similar to those in the restricted domain. Let $j \in \{0, ..., 2|C_A| - 1\}$ denoting that R_j is some IPFR. The c_i^j-exits of R_j are defined by:

$$c_i^j\text{-}\mathbf{EX} := \bigcup_{s \in R_j} \left(\mathbf{Reach}_{R_j}(s) \setminus R(c_i^j) \right)$$

for $i \in \{1, 2\}$. Let $\mu := -\min_{a \in A_j} d[a, c_1^j]$ denote the minimal integer distance effect on c_1^j. We partition the c_1^j-exits into a finite number of lines $L_1, ..., L_\mu$, where $L_i := \{s \in c_1^j\text{-}\mathbf{EX} : d[s, c_1^j] = -i\}$. Here again, $L_1, ..., L_\mu$ are called the c_i^j-exiting lines of R_j. We will use the term "exiting lines" in general meaning all exiting lines of all IPFRs.

The analysis of the restricted domain works for all states in an IPFR with a finite number of exceptions, as we will prove in the next corollary. With exceptions, we mean invalid action applications that are assumed to be valid by the methods discussed in the previous section but are not actually valid. Check region $4'$ in Fig. 4.

Corollary 5. *Regional reachability is decidable.*

Proof. For goal states within the same conic IPFR, notice that the finite region, where exceptions to the restricted domain take place can be excluded by a finite number of linear constraints. For goal states on the exits of the region, we can use the methods defined before to represent the reachable states with a finite number of APs. Since only a finite number of minimal paths pass through any finite region, only a finite number of the states in these APs will be invalid. We can remove all these and ensure that we keep the finite representation. In other words, the methods used in Corollary 3 can be applied here with finitely many exceptions. On the other hand, regional reachability for non-conic IPFR can be modeled as a SNP problem with one numeric and a finite number of propositional variables because one of the numeric variables is bounded. □

6 Arithmetic Progression Mining

If all paths are finite, reachability is decidable. Therefore, we must find methods to prune the search if an infinite path is discovered. We call an infinite path $(s_i)_{i\in\mathbb{N}}$ minimal iff for all $i < j \in \mathbb{N}$, $s_i \neq s_j$. With this definition, notice that any minimal infinite path, must eventually leave all finite PFRs and: (1) remain in exactly one IPFR or (2) visit the exiting lines infinitely often. Since regional reachability is established as decidable, if we are searching for a goal state in some IPFR, we can prune all infinite paths of type (1) because: (1.1) If the goal state is in the same IPFR where the path remains, regional reachability should suffice to find out whether it is reachable or not. (1.2) Otherwise, the path remains in a region different than that of the goal state. We can therefore focus the search on paths of type (2). In other words, focusing on paths that keep visiting the exiting lines is sufficient. We can do the same for any goal state s_g by finding all states on exiting lines from which s_g is reachable. Even for goal conditions $G \subset LC(V)$, where $R(G)$ is infinite, we can find an equivalent SNP with a finite number of goal states. We can, therefore, design a search algorithm that abstracts from the PFRs by applying projections to exit the current PFR with one iteration. Given an initial state $s \in R_\gamma$ for some $\gamma : C_A \to \{0, 1\}$, remember that for all $t \in \mathbf{Reach}_\gamma(s) \setminus R(c)$, t is on some c-exiting line of R_γ. Similar to the restricted domain, we define:

$$\mathbf{Proj}_c^\gamma(s, v, n) := \bigcup_{i=0}^{n} \left(\mathbf{Reach}_\gamma(s + iv) \cap c\text{-}\mathbf{EX}\right)$$

For any $c \in C_\gamma$, the reachable c-exits from some AP $\mathbf{AP}(s, v, n)$, denoted $\mathbf{Proj}_c^\gamma(s, v, n)$, can be partitioned into a finite number of APs. For any $R \subseteq \mathbb{Z}^2$ we define $\lfloor R \rfloor$ as the minimal partition of R into arithmetic progressions, each being contained in some PFR. This is possible because all states in an AP are contained in a line and all PFRs are convex.

Algorithm 1: Arithmetic Progression Mining (APM)

Input: Initial state $s_0 \in \mathbb{Z}^2$
1 Found $\leftarrow \{\mathbf{AP}(s_0, 0, 0)\}$; Covered $\leftarrow \emptyset$;
2 **while** *there exists* $\mathbf{AP}(s, v, n) \in$ Found $\setminus$ Covered **do**
3 Covered $\leftarrow$ Covered $\cup \{\mathbf{AP}(s, v, n)\}$;
4 Let $\gamma : C_A \to \{0, 1\}$ s.t. $s \in R_\gamma$;
5 **for** $c \in C_\gamma$ **do**
6 Found $\leftarrow$ Found $\cup \lfloor \mathbf{Proj}_c^\gamma(s, v, n) \rfloor$;
7 reduce(Found)

We will demonstrate APM with the example in Fig. 3. We start at the state $s_0 = \langle 0, 0\rangle$: (1) We project from $s_0 \in R_{10}$ onto R_{11} using $A_{10} = \{a\}$ and get one reachable state $s_1 = \langle 0, 1\rangle$. (2) Afterwards, we project from s_1 using $A_{11} = \{a, b\}$ and get $\mathbf{AP}(s_2, v, \infty)$ which is located in the $[x \leq 0]$-exiting line L_1 of R_{11}, where $v := \langle 0, 1\rangle$. This indeed corresponds to reachable states, specifically, since a is a repetition in R_{11}, we get $s_2 + nv = s_0 + v_{w_n}$ with $w_n := a^{n+1}b$ for all $n \in \mathbb{N}$. We partition $\mathbf{AP}(s_2, v, \infty) = \{s_2\} \cup \mathbf{AP}(s_3, v, \infty)$ by its intersection with the PFRs R_{01} and R_{00}. (3) Finally, we project $\mathbf{AP}(s_3, v, \infty)$ onto the $[y \geq 1]$-exiting line L_2 of R_{01}, and get $\mathbf{AP}(s_4, u, \infty)$, where $u = \langle 1, 0\rangle$. This again corresponds to reachable states, since for all $n \in \mathbb{N}$, $s_4 + nu = s_0 + v_{w_n}$ with $w_n := a^{n+2}b^{n+2}$. After that, APM terminates since no more projections that lead to new states are available. Given a goal state s_g in some exiting line, we can determine if s_g is reachable by checking whether s_g is in any of the reachable APs. This is a decidable problem.

Definition 13. $\mathbf{AP}(s_1, v_1, n_1)$ *collides with* $\mathbf{AP}(s_2, v_2, n_2)$ *iff* $\mathbf{AP}(s_1, v_1, n_1) \cap \mathbf{AP}(s_2, v_2, n_2) \neq \emptyset$.

6.1 Reductions

To prove termination of APM, let us consider the function reduce in command line 7 of Alg. 1. The goal is to decrease the size of Found. We define the following reductions:

- **Inclusion Reduction**: Let $v \in \mathbb{Z}^2 \setminus \{0\}$, and let $\mathbf{AP}(s_1, v, n_1)$ collide with $\mathbf{AP}(s_2, kv, n_2)$ for $k \in \mathbb{N}, s_1, s_2 \in \mathbb{Z}^2, n_1, n_2 \in \mathbb{N}$. If $(s_2 - s_1) \cdot v > 0$ and $(s_1 + n_1 v - (s_2 + n_2 kv)) > 0$, then $\mathbf{AP}(s_2, kv, n_2) \subseteq \mathbf{AP}(s_1, v, n_1)$. We can safely remove $\mathbf{AP}(s_2, kv, n_2)$ from Found in that case.
- **Collision Reduction**: Let $v \in \mathbb{Z}^2 \setminus \{0\}$, and let $\mathbf{AP}(s_1, v, n_1)$ collide with $\mathbf{AP}(s_2, v, n_2)$. Assume w.l.o.g. that $(s_2 - s_1) \cdot v \geq 0$. Then, there must exist $i \in \{0, ..., n_1\}$ s.t. $s_1 + iv = s_2$ denoting the first collision. We can safely replace both APs in Found with $\mathbf{AP}(s_1, v, n_2 + i) = \mathbf{AP}(s_1, v, n_1) \cup \mathbf{AP}(s_2, v, n_2)$. If we can prove that an infinite number of different arithmetic progressions $\mathbf{AP}(s_i, v, n_i); i \in \mathbb{N}$ can be found s.t. $\mathbf{AP}(s_i, v, n_i)$ collides with $\mathbf{AP}(s_{i+1}, v, n_{i+1})$ for all $i \in \mathbb{N}$, then, we can replace all with $\mathbf{AP}(s_0, v, \infty)$. This is important because, when using the same pair of actions within some region a, b, the projection vector $u_{a,b}$ is fixed, i.e., all APs found by projection using these actions will have a fixed update vector $v = u_{a,b}$.

6.2 Completion

Remember that all states visited by APM should be on exiting lines, except for the first one, and that any goal conditions can be translated into a finite number of goal states s_g that are on exiting lines. Let $\lceil \text{Found} \rceil := \{t \in \mathbf{AP}(s, v, n) : \mathbf{AP}(s, v, n) \in \text{Found}\}$. If the goal state $s_g \in \lceil \text{Found} \rceil$, we can terminate APM. Otherwise, specifically when s_g is not reachable, we need ways to prove that the search has been completed.

Definition 14. *Given an initial state $s_0 \in \mathbb{Z}^2$, and an exiting line L of some* IPFR *R_j; $j \in \{0, ..., 2|C_A|-1\}$, we say that L is completed from the side of R_j iff all reachable states in L that are reachable from R_j were found, i.e., iff for all $s \in \mathbf{Reach}(s_0) \cap L$, if $s - v_a \in \mathbf{Reach}(s_0) \cap R_j$ for some $a \in A_j$, then, $s \in \lceil \text{Found} \rceil$.*

An exiting line L is completed once all the exiting lines, from which L is reachable, are completed. The reason is that all reachable paths towards L will be discovered. APM terminates if all exiting lines are completed. If all infinite exiting lines are completed, the finite exiting lines will be eventually completed. Therefore, we will focus on the infinite exiting lines of IPFRs.

Remember that $\text{level}_L(s)$ is an encoding of all states $s \in L$. Therefore, for all $\mathbf{AP}(s, v, n)$ in Found, some exiting line L exists s.t. $\mathbf{AP}(s, v, n) \subseteq L$. We can define a corresponding one-dimensional AP over the levels of L by $\mathbf{AP}(l_s, \delta_v, n)$ for some $l_s, \delta_v \in \mathbb{N}$. This will make later arguments easier to understand.

Theorem 6. *Let R_j be some* IPFR *where a repetition is applicable and let L be some infinite exiting line of R_j. Any exiting line of R_j can be completed from the side of R_j with a finite number of iterations of* APM.

Proof. If $\mathbf{Reach}(s_0) \cap R_j$ is empty, we are done. Otherwise, some state $s \in \mathbf{Reach}(s_0) \cap R_\gamma$ will be found. Since $a \in A^*_\gamma$ is a repetition, we have $\#_s[a] = \infty$. Therefore, $\mathbf{AP}(t, v, \infty)$ will be in Found in the next iteration of the while-loop, for some $t, v \in \mathbb{Z}^2$. Let $\mathbf{AP}(l_t, \delta_v, \infty)$ be the corresponding level AP with $l_t = \text{level}_L(t)$ and $\delta_t \in \mathbb{N}$. We partition the levels of L, into the classes modulo δ_v, $M_i; i \in \{0, ..., \delta_v - 1\}$. Each such class is an AP itself, i.e.:

$$\mathbb{N} = \bigcup_{i=0}^{\delta_v - 1} M_i = \bigcup_{i=0}^{\delta_v - 1} \mathbf{AP}(i, \delta_v, \infty)$$

With each new visit to some state $s' \in R_j$, R_j can be exited again in the next iteration. If L is reachable from s', then, there exists $i \in \{0, ..., \delta_v - 1\}$ and $t' \in M_i$ s.t. t' is visited in the next step. Here again, $\mathbf{AP}(t', v, \infty)$ will be in Found because $\#_{s'}[a] = \infty$. Additionally, $n \in \mathbb{N}$ exists s.t. $l'_t = n\delta_v + i$ for t' to be in M_i. After that, only the sub-AP $\mathbf{AP}(i, \delta_v, n-1)$ of M_i is left, i.e., M_i will be visited at most n extra times after t'. The other cases will be covered by the inclusion reduction and would not increase the size of Found. In other words, we partitioned L into a finite number of subsets, each of which will be visited a finite number of times only. □

The last theorem has an important impact on the ease of solving the problem because it clears out many cases. Notice that, by Lemma 1, if two unparallel crossings with different directions are applicable in some IPFR R_j, a repetition or co-repetition can be constructed. The case for repetition has been covered in the last theorem. For the case of co-repetition, we can ensure that, when starting from the goal state s_g and considering co-reachability, the co-repetition has the same function as repetition has with reachability. Therefore, by running APM for co-reachability, if an IPFR contains a co-repetition, its exiting lines will eventually be completed from its side. If all crossings in all IPFRs have one direction only, w.l.o.g. positive, each IPFR R_j has infinite exiting lines on one side only, i.e., c_1^j-**EX** is finite for all $j \in \{0, ..., 2|C_A| - 1\}$. In this case, all paths within the IPFRs have a clockwise direction (Check Fig. 6 for an example).

6.3 *Infinite Path Pruning*

When considering both reachability and co-reachability for APM, we reach, after a finite number of iterations, some state s_0 in some exiting line L_0 that is reachable from the initial state, and a goal state s_g in some other exiting line L_g that is co-reachable from the goal state. By studying the crossing directions between L_0 and L_g, including any intermediate exiting lines, we can know whether at least some path between L_0 and L_g exists, and construct one if that is the case. If no paths between L_0 and L_g exist, clearly, s_g is not reachable from s_0 and we can terminate the search. Otherwise, we can ensure that both s_0 and s_g are on the same exiting line L and terminate whenever L is completed from all sides, of which there are two in $2D$. Having done that, we can prove the decidability of SNP with exactly two numeric variables.

Theorem 7. *Any* SNP $(A, \{x, y\}, s_0, G)$ *is decidable.*

Proof. (Sketch) Remember that each of s_0 and s_g correspond to some levels $l_0, l_g \in \mathbb{N}$ of L, resp. Therefore, we can focus on paths from and to L and study the increase or decrease in levels they produce. We call such a path odd iff it visits L from the same side, and even otherwise (check Fig. 5). The existence of an odd path implies that it passes through some IPFR where a repetition or a co-repetition can be constructed, meaning that the side of L, from which the odd path visits L, will be eventually completed. Finally, all even paths can be classified into a finite number of classes s.t.:

- If all paths from and to L increase/decrease the level, then, after reaching any state with a level higher/lower than l_g, resp., the search can be pruned (Check Fig. 6 for an example).
- If an even path that increases the level, and another that decreases it, exist, then, L will be eventually completed, specifically since an infinite number of pair-wise colliding APs will be found within L (check Fig. 5, right). □

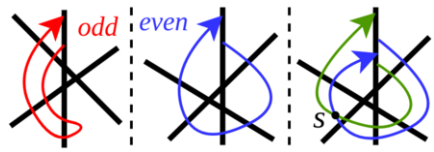


Figure 5. We show three partitions of the state space by 3 linear constraints (black). (1) On the left is a picture of an odd path. (2) In the middle is a picture of an even path. (3) On the right, an example of the collision in s between two even paths, one that increases the level (green) and another that decreases it (blue).

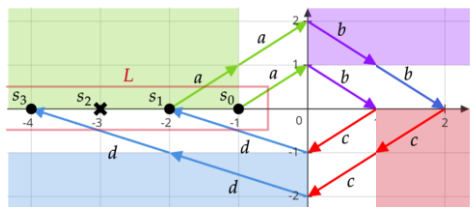


Figure 6. An example where all paths from and to some exiting line L (red) increase the level. Each action of $A = \{a, b, c, d\}$(colored) is applicable at the region colored with the same color as its effect vector. Notice that, starting at s_0, once s_3 is reached, the search for a goal state like s_2 can be pruned.

7 Conclusion

Remember that the arguments given in this paper work only for the 2-variable case. Specifically, although precondition-free regions can be defined equivalently for any number of variables, their exits cannot be partitioned into a finite number of lines. However, regional reachability, i.e., reachability within any specific PFR is generally decidable.

In [4], it is shown that two numeric and one propositional variable can model any Collatz function, e.g., given $n \in \mathbb{N}$, if n is divisible by 2, then, $n \mapsto n/2$, otherwise, $n \mapsto 3n + 1$. We can model linear functions such as $n \mapsto n/2$ and $n \mapsto 3n + 1$ with two numeric variables. However, with exactly two numeric variables, we do not have enough expressiveness to precondition any action based on the resulting multiplication value, because both variables are required for the expressiveness of linear functions to arise. The additional propositional variable in that paper determines whether to apply $n \mapsto n/2$ or $n \mapsto 3n + 1$. Having both within one 2D state space implies the existence of even paths that increase the level and others that decrease it, ultimately leading to AP collisions. The propositional variable is therefore used to separate the discovered APs into two different planes and avoid collision.

Finally, the APM algorithm can reduce the search space extensively, specifically when repetitions can be discovered. It is therefore a powerful, correct but not necessarily complete, search tool that can be used for any number of numeric variables.

8 Acknowledgment

The authors gratefully acknowledge the funding by the BMBF in Germany for the project AIStudyBuddy (No. 16DHBKI016), and the EU ICT-48 2020 project TAILOR (No. 952215).

References

[1] F. Bacchus and Q. Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, 71(1):43–100, 1994.
[2] G. S. Boolos and R. C. Jeffrey. Computability and logic, 1989.
[3] C. Domshlak and R. I. Brafman. Structure and complexity in planning with unary operators. In *AIPS*, pages 34–43, 2002.
[4] D. Gnad, M. Helmert, P. Jonsson, and A. Shleyfman. Planning over integers: Compilations and undecidability. ICAPS, 2023.
[5] H. Helal and G. Lakemeyer. An analysis of the decidability and complexity of numeric additive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 267–275, 2024.
[6] M. Helmert. Decidability and undecidability results for planning with numerical state variables. In *AIPS*, pages 44–53, 2002.
[7] M. Helmert. A planning heuristic based on causal graph analysis. In *ICAPS*, volume 16, pages 161–170, 2004.
[8] J. Hoffmann. The metric-ff planning system: Translating"ignoring delete lists"to numeric state variables. *Journal of artificial intelligence research*, 20:291–341, 2003.
[9] C. A. Knoblock. Automatically generating abstractions for planning. *Artificial intelligence*, 68(2):243–302, 1994.
[10] E. Scala, P. Haslum, S. Thiébaux, et al. Heuristics for numeric planning via subgoaling. 2016.
[11] A. Shleyfman, D. Gnad, and P. Jonsson. New complexity results for structurally restricted numeric planning. In *ICAPS 2022 Workshop on Heuristics and Search for Domain-independent Planning*, 2022.