What's Decidable About Motion Planning?

Shankara Narayanan Krishna
IIT Bombay
krishnas@cse.iitb.ac.in

Aviral Kumar
IIT Bombay
aviral@cse.iitb.ac.in

Ashutosh Trivedi
University of Colorado Boulder
ashutosh.trivedi@colorado.edu

ABSTRACT

Efficient planning algorithms form central components in various stages of autonomous vehicle navigation right from route planning to lower-level motion planning. The classical *piano mover's problem*—of moving a piano from one room to another while avoiding hitting other obstacles—and its variants are paradigmatic examples of such planning problems and have been extensively studied in robotics and artificial intelligence. As a result a number of practical solutions exists ranging from "best-effort" sampling-based techniques such as rapidly exploring random trees and dense trees, to "complete" combinatorial techniques such as cylindrical algebraic decomposition algorithm and Canny's algorithm. The general motion planning problem with polyhedral obstacles was shown (Reif 1979) to be PSPACE-complete and certain form of planning under uncertainty in 3D environment is known (Canny and Reif 1987, and Canny 1988) to be NEXPTIME-hard. In this paper we take a fresh look at this classical problem, and give a precise boundary between decidable and undecidable motion planning problems for systems with simple and natural dynamics. For this purpose we consider constant-rate multi-mode systems and study reach-while-avoid problem for point-objects with obstacles given using linear inequalities. We show that this problem is in general undecidable, and we recover decidability by making certain assumptions about the obstacles. We also present a new algorithm to solve this problem and compare the performance with sampling based algorithms.

Categories and Subject Descriptors

D.4.7 [**Organization and Design**]: Real-time systems and embedded systems; B.5.2 [**Design Aids**]: Verification; I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

Keywords

Switched Systems, Motion Planning, Hybrid Automata

1. INTRODUCTION

Autonomous vehicles (AVs)—also known as self-driving, robotic, or driver-less vehicles—have come a long way since the first DARPA grand challenge in 2004. While, during the first grand challenge, out of fifteen participating teams, no autonomous vehicle could cover more than 5 percent of the way; as of April 30, 2016, Google self-driving cars (`https://www.google.com/selfdrivingcar/`) have driven more than 1.5 million miles in autonomous mode with an average of 15 thousand miles per week [13]. Arguably, efficient motion planning algorithms are one of the key drivers behind the success of autonomous vehicle navigation and understanding the boundary between hard and easy instances of the motion planning problem can potentially contribute towards the design of better algorithms. The goal of this paper is to study the theoretical complexity of motion planning and control problem to provide a precise boundary between decidability and undecidability.

Most of the autonomous vehicle planing and control frameworks [17, 21] follow the hierarchical planning architecture outlined by Firby [10] and Gat [12]. The key idea here is to separate the complications involved in low-level hardware control from high-level planning decisions to accomplish the navigation objective. A typical example of such separation-of-concerns include proving controllability property (vehicle can be steered from any start point to arbitrary neighborhood of the target point) of the motion-primitives of vehicle followed by the search (path-planning) for an obstacle-free path (called the *roadmap*) and then utilizing the controllability property to compose the low-level primitives to follow the path (path-following). However, in the absence of the controllability property, it is not always possible to follow a roadmap with given motion-primitives, even in the situations when there is indeed a separate roadmap that can be followed by the controller. For such cases, a more desirable alternative is to study motion planning that is not opaque to the motion-primitives available to the controller.

In order to study this motion planning problem we model the system with constant-rate multi-mode systems [4]—a switched system with constant-rate dynamics (vector) in every mode—and study the problem of motion planning in a convex workspace $\mathcal{W}$ in the presence of a finite set of convex (polyhedral) obstacles $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k\}$ for a given start and target state. Alur et al. [4] studied this problem in the absence of obstacles and showed that reachability can be performed in polynomial time. Our key result is that in case of arbitrary polyhedral obstacles the problem of deciding reachability is undecidable. On a positive side we show that if for all the obstacles are full-dimensional and closed, hen the problem is decidable and can be solved using a variation of cell-decomposition algorithm [24]. We present a novel bounded model-checking [9] inspired algorithm equipped with acceleration to decide the reachability. We use Z3-theorem prover as the constraint satisfaction engine for quadratic formulas in our implementation. We show the efficiency of our algorithm by comparing its performance with our implementation of popular sampling based algorithm *rapidly-exploring random tree* (RRT).

Related Work. Piano mover's problem is a classical problem in robotic motion planning that concerns with finding a sequence of translations and rotations to move a rectangular shaped object in an environment constrained by obstacles given as semi-algebraic models. While path-connectivity problems can be efficiently solved (NL-complete [22]) in discretized spaces of finite graphs, the complexity of the problem of solving motion planning problem in continuous spaces is very high [23, 8]. A key step of hierarchical motion planning and control is the search for roadmaps. A roadmap intuitively is a connected path on the obstacle-free space such that it is efficient to compute a path from any point in obstacle-free space to corresponding "closest" points in the roadmap, and it is connection-preserving i.e. there is a path between two points on the obstacle-free space, if and only if there is a path between the corresponding closest points in the roadmap. Cell decomposition, visibility graphs, and potential field methods offer efficient ways to compute roadmaps in two- and three-dimensional spaces, however the complexity of the problem of computing optimal roadmaps for higher-dimension systems remains high [24].

Reif [23] showed that mover's problem is PSPACE-hard for spaces with arbitrary dimensions for robots made of polyhedral shape and a finite set of polyhedral obstacles. Schwartz and Sharir [24] established first lower bound for generalized mover's problem by giving an exact cell decomposition algorithm which is doubly exponential in the number of dimensions. Canny gave an improved algorithm which was exponential in the dimension [8], while he showed that the problem of generating shortest-path in workspaces with polyhedral obstacles is NP-hard. Canny and Reif [7] also studied the problem in the presence of uncertainty and showed the problem to be NEXPTIME-hard. For a detailed survey of well-known motion planning algorithms we refer to excellent expositions by Latombe [18] and LaValle [19].

Motion-planning problem while respecting dynamics of the system can be modeled [11] using the framework of hybrid automata [1, 14], however the problem of deciding reachability is undecidable even for simple stopwatch automata [16]. There is a vast literature on decidable subclasses of hybrid automata [1, 6]. Most notable among these classes are initialized rectangular hybrid automata [16], two-dimensional piecewise-constant derivative systems [5], and discrete-time control for hybrid automata [15], and timed automata [2]. For a review of related work on constant-rate multi-mode systems we refer the reader to [3, 4]. Le Ny and Pappas [20] initiated work on the sequential composition of robust controller specifications. In this light, our results can be understood as an effort to analyze complexity of this problem for the system of relatively simple dynamics.

Organization. The paper is organized as follows. We begin by providing a motivating example and illustrate our key results in the next section. In Section 3 we provide formal definition of the key problem, followed by the proof of our decidability result in Section 4. We show our central undecdability result in Section 5 and illustrate our experience with an implementation of our algorithm and its comparison with RRT algorithm in Section 6 followed by conclusion.

2. MOTIVATING EXAMPLE

As a motivating example for our algorithm, consider a two-dimensional multi-mode system with three modes m_1, m_2 and m_3 shown geometrically with their rate-vectors in Figure 1(a). We consider the problem of motion planning in the arena given in Figure 1(b) with two rectangular obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$ and source and target points $\overline{x}_s$ and $\overline{x}_t$. In particular, we are interested in the question whether it is possible to move a point-size robot from point x_s to point x_t using directions dictated by the multi-mode system given in Figure 1(a) while avoiding passing through or even grazing through any obstacle.

For two dimensional systems this problem can be efficiently solved using visibility graphs [18]. For higher dimension system it appears that the problem should be decidable using an exact cell-decomposition algorithms as studied by Canny [8] for even polynomial obstacles. However, the cell-decomposition algorithms are not constrained by system dynamics, and hence it is not immediate whether our problem can be answered using cell-decomposition algorithms. Indeed, we present a surprising result in Section 5 that in general the problem of deciding reachability is not decidable even when obstacles are given as linear inequalities.

However, the example considered in Figure 1 has an interesting property that all the obstacles are full-dimensional, convex, and closed. This property, as we show later, makes the problem decidable. In fact, if we decompose the workspace into cell using any off-the-shelf cell-decomposition algorithm, we need to consider all sequences of obstacle-free cells to decide the reachability. In particular, for a given sequence of obstacle-free cells such that starting point is in the first cell, and the target point is in last cell, one can write a linear program checking whether there is a sequence of intermediate states, two in each cell, such that these points are reachable in the sequence. Our key observation is that one need not to consider cell-sequence larger than total number of cells since for reachability, if the full-dimensionality property holds, it does not help for the system to leave a cell and enter again at a later time. This is primarily due to lack of guards in multi-mode system and the convexity of cells.

This approach, while implying decidability, is not very efficient since one needs to consider exponentially many cells, and their permutations. However, this result provides an upper bound on sequence of "meta-steps" or "hops" through the cells that system needs to take in order to reach the target and hint towards an bounded model-checking [9] approach. We progressively increase bound k and ask whether there is a sequence of points $\overline{x}_0, \ldots, \overline{x}_{k+1}$ such that $\overline{x}_0 = \overline{x}_s$, $\overline{x}_{k+1} = \overline{x}_t$, and for all $0 \leq i \leq k$ we have that $\overline{x}_i$ can reach $\overline{x}_{i+1}$ using the rates provided by the multi-mode system (convex cone of rates translated to $\overline{x}_i$ contains $\overline{x}_{i+1}$) and the line segment $\lambda\overline{x}_i + (1-\lambda)\overline{x}_{i+1}$ does not intersect any obstacle. Notice that if this condition is satisfied, then the system can safely move from point $\overline{x}_i$ to $\overline{x}_{i+1}$ safely by carefully choosing a scaling down of the rates so as to stay in the safety set. This idea is nicely illustrated in Figure 1.

Let us first consider $k = 0$ and notice that one can reach

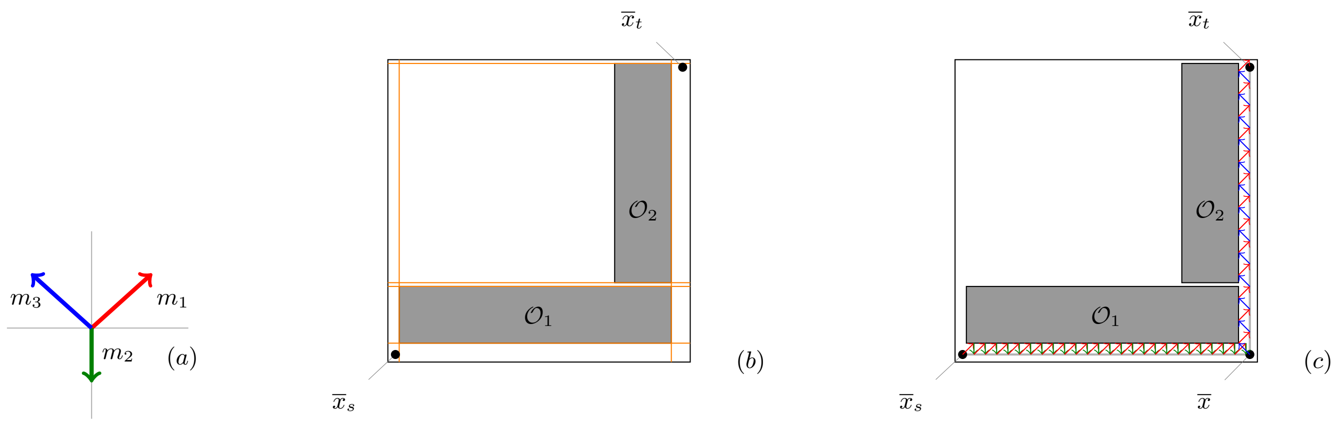


Figure 1: a) A multi-mode system, b) a rectangular workspace with an "L"-shaped arena consisting of with two obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$ with starting and target points $\overline{x}_s$ and $\overline{x}_t$ along with the cell-decomposition shown by orange lines, and c) a safe schedule from $\overline{x}_s$ to $\overline{x}_t$.

point $\overline{x}_t$ from $\overline{x}_s$ using just the mode m_1, however unfortunately the line segment connecting these points passes through both obstacles. In this case we increase the bound by 1 and consider the problem of finding a point $\overline{x}$ such that the system can reach from $\overline{x}_s$ to $\overline{x}$ and also from $\overline{x}$ to $\overline{x}_t$, and the line segment connecting $\overline{x}_s$ with $\overline{x}$, and $\overline{x}$ with $\overline{x}_t$ do not intersect with the obstacles. It is easy to see from the Figure 1 that it is indeed the case. Hence, there is a schedule that steers the system from $\overline{x}_s$ to $\overline{x}_t$ as shown in the Figure 1(c).

The property that we need to check at each step is a first-order formula of the form $\exists X \forall Y F(X,Y)$ where $F(X,Y)$ is a linear formula. By invoking Tarski-Seidenberg theorem we know that checking the satisfiability of this property is decidable. However, since the universal quantification is occurring in a very special format, we exploit this property and give a direct quantifier elimination based on Fourier-Motzkin elimination procedure to get existentially quantified quadratic constraints that can be efficiently checked using theorem provers such as Z3 (`https://github.com/Z3Prover/z3`). This gives us a complete procedure to decide reachability for multi-mode systems when the set of convex obstacles do not intersect with each other. We show that this algorithm compares well even with sampling based algorithm RRT.

3. PROBLEM FORMULATION

In this section we introduce the notations used in the paper and present known results that we use in this paper.

3.1 Preliminaries

Points and Vectors. Let $\mathbb{R}$ be the set of real numbers. We represent the states in our system as points in $\mathbb{R}^n$ that is equipped with the standard *Euclidean norm* $\|\cdot\|$. We denote points in this state space by $\overline{x}, \overline{y}$, vectors by $\vec{r}, \vec{v}$, and the i-th coordinate of point $\overline{x}$ and vector $\vec{r}$ by $\overline{x}(i)$ and $\vec{r}(i)$, respectively. We write $\vec{0}$ for a vector with all its coordinates equal to 0; its dimension is often clear from the context. The distance $\|\overline{x}, \overline{y}\|$ between points $\overline{x}$ and $\overline{y}$ is defined as $\|\overline{x} - \overline{y}\|$. For two vectors $\vec{v}_1, \vec{v}_2 \in \mathbb{R}^n$, we write $\vec{v}_1 \circ \vec{v}_2$ to denote their dot product, defined as $\sum_{i=1}^{n} \vec{v}_1(i) \cdot \vec{v}_2(i)$.
Boundedness and Interior. We denote an *open ball* of radius $d \in \mathbb{R}_{\geq 0}$ centered at $\overline{x}$ as $B_d(\overline{x}) = \{\overline{y} \in \mathbb{R}^n : \|\overline{x}, \overline{y}\| < d\}$. We denote a closed ball of radius $d \in \mathbb{R}_{\geq 0}$ centered at $\overline{x}$ as $\overline{B_d(\overline{x})}$. We say that a set $S \subseteq \mathbb{R}^n$ is *bounded* if there exists $d \in \mathbb{R}_{\geq 0}$ such that, for all $\overline{x}, \overline{y} \in S$, we have $\|\overline{x}, \overline{y}\| \leq d$. The *interior* of a set S, $\text{int}(S)$, is the set of all points $\overline{x} \in S$, for which there exists $d > 0$ s.t. $B_d(\overline{x}) \subseteq S$.
Convexity. A point $\overline{x}$ is a *convex combination* of a finite set of points $X = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k\}$ if there are $\lambda_1, \lambda_2, \ldots, \lambda_k \in [0,1]$ such that $\sum_{i=1}^{k} \lambda_i = 1$ and $\overline{x} = \sum_{i=1}^{k} \lambda_i \cdot \overline{x}_i$. The *convex hull* of X is the set of all points that are convex combinations of points in X. We say that $S \subseteq \mathbb{R}^n$ is *convex* iff, for all $\overline{x}, \overline{y} \in S$ and all $\lambda \in [0,1]$, we have $\lambda\overline{x} + (1-\lambda)\overline{y} \in S$ and moreover, S is a *convex polytope* if it is bounded and there exists $k \in \mathbb{N}$, a matrix A of size $k \times n$ and a vector $\vec{b} \in \mathbb{R}^k$ such that $\overline{x} \in S$ iff $A\overline{x} \leq \vec{b}$. We call it a convex polyhedron if the boundedness condition is relaxed. A closed *hyper-rectangle* is a special kind of convex polytope if it can be characterized as $\overline{x}(i) \in [a_i, b_i]$ for each $i \leq n$ where $a_i, b_i \in \mathbb{R}$.

A point $\overline{x}$ is a *vertex* of a convex polytope P if it is not a convex combination of two distinct (other than $\overline{x}$) points in P. For a convex polytope P we write $\mathsf{vert}(P)$ for the finite set of points that correspond to the vertices of P. Each point in P can be written as a convex combination of the points in $\mathsf{vert}(P)$. In other words, P is the *convex hull* of $\mathsf{vert}(P)$.

3.2 Multi-Mode Systems

Definition 1 ((Constant-Rate) Multi-Mode Systems). *A multi-mode system (MMS) is a tuple $\mathcal{H} = (M, n, R)$ where:*

- *M is the finite nonempty set of* modes,
- *n is the number of continuous variables, and*
- *$R : M \to \mathbb{R}^n$ gives for each mode the rate vector whose i-th entry specifies the change in value of the i-th variable per time unit.*

For computation purposes, we assume that all real numbers are rational and represented in the standard way by writing down the numerator and denominator in binary.

Example 1. *An example of a 2-dimensional multi-mode system $\mathcal{H} = (M, n, R)$ is shown in Figure 1(a) where $M =$*

$\{m_1, m_2, m_3\}$, $n = 2$, *and the rate vector is such that* $R(m_1) = (1, 1)$, $R(m_2) = (0, -1)$, *and* $R(m_3) = (-1, 1)$.

A *schedule* of an MMS specifies a timed sequence of mode switches. Formally, a *schedule* is defined as a finite or infinite sequences of *timed actions*, where a timed action $(m, t) \in M \times \mathbb{R}_{\geq 0}$ is a tuple consisting of a mode and a time delay. A finite *run* of an MMS $\mathcal{H}$ is a finite sequence of states and timed actions $r = \langle \overline{x}_0, (m_1, t_1), \overline{x}_1, \ldots, (m_k, t_k), \overline{x}_k \rangle$ such that for all $1 \leq i \leq k$ we have that $\overline{x}_i = \overline{x}_{i-1} + t_i \cdot R(m_i)$. For such a run r we say that $\overline{x}_0$ is the *starting state*, while $\overline{x}_k$ is its *terminal state*. An *infinite run* of an MMS $\mathcal{H}$ is similarly defined to be an infinite sequence $\langle \overline{x}_0, (m_1, t_1), \overline{x}_1, (m_2, t_2), \ldots \rangle$ such that for all $i \geq 0$ we have that $\overline{x}_i = \overline{x}_{i-1} + t_i \cdot R(m_i)$.

Given a finite schedule $\sigma = \langle (m_1, t_1), (m_2, t_2), \ldots, (m_k, t_k) \rangle$ and a state $\overline{x}$, we write $Run(\overline{x}, \sigma)$ for the (unique) finite run $\langle \overline{x}_0, (m_1, t_1), \overline{x}_1, (m_2, t_2), \ldots, (m_k, t_k), \overline{x}_k \rangle$ such that $\overline{x}_0 = \overline{x}$. In this case, we also say that schedule σ steers the MMS $\mathcal{H}$ from state $\overline{x}_0$ to state $\overline{x}_k$.

We characterize a safety set using a tuple $(\mathcal{W}, \mathcal{O})$ where $\mathcal{W} \subseteq \mathbb{R}^n$ is a convex *workspace* and $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k\}$ is a finite set of convex obstacles. We assume in the rest of the paper that $\mathcal{W}$ and $\mathcal{O}_i$ for all $1 \leq i \leq k$ are convex polytopes. We say that a set of convex obstacles $\mathcal{O}$ is *full-dimensional* if every obstacle $\mathcal{O}_i$ for all $1 \leq i \leq k$ is full-dimensional. We define the safety set characterized by $(\mathcal{W}, \mathcal{O})$ as

$$S_{\mathcal{W} \setminus \mathcal{O}} = \{x \in \mathbb{R}^n \ : \ x \in \mathcal{W} \text{ and } x \notin \mathcal{O}_i \text{ for all } 1 \leq i \leq k\}.$$

we say that a finite run $\langle \overline{x}_0, (m_1, t_1), \overline{x}_1, (m_2, t_2), \ldots \rangle$ is S-safe if for all $i \geq 0$ we have that $\overline{x}_i \in S$ and $\overline{x}_i + \tau_{i+1} \cdot R(m_{i+1}) \in S$ for all $\tau_{i+1} \in [t_i, t_{i+1}]$, assuming $t_0 = 0$. Notice that if S is a convex set then for all $i \geq 0$, $\overline{x}_i \in S$ implies that for all $i \geq 0$ and for all $\tau_{i+1} \in [t_i, t_{i+1}]$ we have that $\overline{x}_i + \tau_{i+1} \cdot R(m_{i+1}) \in S$. Sometimes we simply call a schedule or a run safe when the safety set and the starting state is clear from the context. We say that a state $\overline{x}'$ is S-safe reachable from a state $\overline{x}$ if there exists a finite schedule σ that is S-safe at $\overline{x}$ and steers the system from state $\overline{x}$ to $\overline{x}'$. We are interested in solving the following reachability problem for multi-mode systems.

Definition 2 (Reachability). *Given a constant-rate multi-mode system $\mathcal{H} = (M, n, R)$, convex workspace $\mathcal{W}$, obstacles set $\mathcal{O}$, start state $\overline{x}_s$ and target state $\overline{x}_t$, the reachability problem* $\textsc{Reach}(\mathcal{H}, S_{\mathcal{W} \setminus \mathcal{O}}, \overline{x}_s, \overline{x}_t)$ *is to decide whether there exists a $S_{\mathcal{W} \setminus \mathcal{O}}$-safe finite schedule that steers the system from state $\overline{x}_s$ to $\overline{x}_t$.*

Alur et al. [4] gave a polynomial-time algorithm to decide if a state $\overline{x}_t$ is S-safe reachable from a state $\overline{x}_0$ for an MMS $\mathcal{H}$ for a convex safety set S. In particular, for starting and target states in the interior of the safety set, they characterized a necessary and sufficient condition.

Theorem 1 ([4]). *Let $\mathcal{H} = (M, n, R)$ is a multi-mode system and let $S \subset \mathbb{R}^n$ be a open, bounded, and convex safety set. Then, there is an S-safe schedule from starting state $\overline{x}_s \in S$ and target state $\overline{x}_t \in S$, if and only if there is $\vec{t} \in \mathbb{R}_{\geq 0}^{|M|}$ satisfying:*

$$\overline{x}_s(j) + \sum_{i=1}^{|M|} R(i)(j) \cdot \vec{t}(i) = \overline{x}_t(j) \text{ for } 1 \leq j \leq n. \quad (1)$$

A key property of this result is that $\overline{x}_t$ is reachable form $\overline{x}_s$ without considering the safety set, then it is also reachable inside arbitrary convex set as long as both $\overline{x}_s$ and $\overline{x}_t$ are strictly in the interior of the safety set. The exact schedule to reach from $\overline{x}_s$ to $\overline{x}_t$ is computed by scaling the schedule in absence of the safety set as explained in the Algorithm 1.

Algorithm 1: $\textsc{Reach\_Convex}(\mathcal{H}, \overline{x}_s, \overline{x}_t, S)$

```
Input: MMS $\mathcal{H} = (M, n, R)$, two points $\overline{x}_s, \overline{x}_t$, convex, open, and bounded safety set $S$
Output: NO if no $S$-safe schedule from $\overline{x}_s$ to $\overline{x}_t$ exists and otherwise such a schedule.
1 $t_1 = \min_{m \in M} \max \{\tau \ : \ \overline{x}_s + \tau \cdot R(m) \in S\}$;
2 $t_2 = \min_{m \in M} \max \{\tau \ : \ \overline{x}_t + \tau \cdot R(m) \in S\}$;
3 $t_{\text{safe}} = \min \{t_1, t_2\}$;
4 Check whether the following linear program is feasible:
      $\overline{x}_s + \sum_{m \in M} R(m) t^{(m)} = \overline{x}_t$ and      (2)
      $t^{(m)} \geq 0$ for all $m \in M$.
5 if no satisfying assignment exists then return NO;
6 else
7     Find an assignment $\{t^{(m)}\}_{m \in M}$ of polynomial size.
8     Set $l = \lceil (\sum_{m \in M} t^{(m)}) / t_{\text{safe}} \rceil$.
9     return the following schedule $\langle (m_i, t_i) \rangle$ where
          $m_k = (k \bmod |M|) + 1$ and $t_k = t^{(m_k)} / l$
      for $k = 1, 2, \ldots, l|M|$.
```

In this paper we study extension of this theorem for the reachability problem with potentially non-convex safety set. In particular, the key contribution of the paper is a precise characterization of the decidability of the reachability problem for multi-mode systems.

Theorem 2. *Given a constant-rate multi-mode system $\mathcal{H}$, convex workspace $\mathcal{W}$, obstacles set $\mathcal{O}$, start state $\overline{x}_s$ and target state $\overline{x}_t$, the reachability problem* $\textsc{Reach}(\mathcal{H}, S_{\mathcal{W} \setminus \mathcal{O}}, \overline{x}_s, \overline{x}_t)$ *is in general undecidable. However, if the obstacle set $\mathcal{O}$ is full-dimensional and closed, the reachability is decidable.*

4. DECIDABILITY

We prove the following theorem in this section.

Theorem 3. *For a multi-mode system $\mathcal{H} = (M, n, R)$, a convex workspace $\mathcal{W}$, an obstacles set $\mathcal{O}$, a start state $\overline{x}_s$ and a target state $\overline{x}_t$, the problem* $\textsc{Reach}(\mathcal{H}, S_{\mathcal{W} \setminus \mathcal{O}}, \overline{x}_s, \overline{x}_t)$ *is decidable if set of obstacles are full-dimensional and closed.*

For the rest of this section let us fix a multi-mode system $\mathcal{H} = (M, n, R)$, a convex workspace $\mathcal{W}$, an obstacles set $\mathcal{O}$, a start state $\overline{x}_s$ and a target state $\overline{x}_t$.

Before we prove this theorem, we need to introduce the idea of convex cell decomposition [18]

Definition 3 (Cell Decomposition). *Given a convex workspace $\mathcal{W} \in \mathbb{R}^n$ and full-dimensional and closed obstacle set $\mathcal{O}$, a* convex cell-decomposition *of $S_{\mathcal{W} \setminus \mathcal{O}}$ is a finite collection $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ of convex polyhedra called cells such that the interiors of any two cells do not intersect and the union of the cells is equal to $S_{\mathcal{W} \setminus \mathcal{O}}$. Two cells in $c, c' \in \mathcal{C}$ are adjacent if and only if $c \cap c'$ if a face of dimension $n-1$.*

Given a convex cell-decomposition $\mathcal{C}$ of $(\mathcal{W}, \mathcal{O})$, a *cellular path* is a sequence of cells $\langle c_1, c_2, \ldots, c_N \rangle$ of $\mathcal{C}$ such that for all $1 \leq i \leq N$ we have that c_i and c_{i+1} are adjacent. We say that a cellular path is acyclic $\langle c_1, \ldots, c_N \rangle$ if no cell repeats in the sequence, i.e. for all $1 \leq i, j \leq N$ we have that $i \neq j$ implies that $c_i \neq c_j$.

Given a cellular path $\pi = \langle c_1, \ldots, c_N \rangle$, a multi-mode system $\mathcal{H} = (M, n, R)$, starting and target points $\overline{x}_s, \overline{x}_t \in S_{\mathcal{W} \setminus \mathcal{O}}$, we say that π is a witness to the reachability if the following linear program is feasible:

$$\exists_{1 \leq i \leq N} \overline{x}_i \quad \exists_{1 \leq i \leq N, m \in M} t_i^{(m)} \cdot (\overline{x}_s = \overline{x}_1 \wedge \overline{x}_t = \overline{x}_N) \wedge \qquad (3)$$
$$\bigwedge_{i=2}^{N} \left(\overline{x}_i = \overline{x}_{i-1} + \sum_{m \in M} R(m) t_i^{(m)} \right) \wedge \bigwedge_{1 \leq i \leq N, m \in M} t_i^{(m)} \geq 0.$$

LEMMA 4. *If set of obstacles are full-dimensional and closed, then for every cellular path π witness to a reachability problem, there exists an acyclic cellular path π' that is also a witness to the same reachability problem.*

PROOF. Assume that we have a non-acyclic path $\pi = \langle c_1, c_2, \ldots, c_N \rangle$ i.e. there are $1 \leq i < j \leq N$ such that we have that $c_i = c_j = c$, i.e. the cell repeats in the sequence. Moreover, assume that π is a witness to reachability for a given multi-mode system reachability problem. We show that there is an acyclic path that is also witness to the reachability.

By definition of being witness to reachability, it follows that there exists a sequence of states $\langle \overline{x}_1, \ldots, \overline{x}_N \rangle$ satisfying the conditions 3. In particular, it implies that there are states $\overline{x}_i, \overline{x}_j \in c$ such that there exists $t^{(m)} \in \mathbb{R}_{\geq 0}$ for all $m \in M$ such that

$$\overline{x}_i + \sum_{m \in M} t^{(m)} \cdot R(m) = \overline{x}_j.$$

In particular for every $m \in M$ we have that $t^{(m)} = \sum_{i \leq k < j} t_i^{(m)}$ where $t_i^{(m)}$ is a feasible solution of 3. Hence, from theorem 1 it follows that there is a path from the stat $\overline{x}_i$ to $\overline{x}_j$ that does not leave the convex cell $c = c_i = c_j$. In follows that the cellular path π' given as

$$\{c_1, c_2, \ldots, c_i, c_{j+1}, \ldots, c_N\}$$

is also a witness to reachability. We continue the process of removing segments of repeated cells until we remove all the repeated cells from the path, while preserving the witness property, to get an acyclic cellular path that is also a witness. □

PROOF OF THEOREM 3. For full-dimensional and closed obstacle sets, if there is a path from source to target point, there is one that goes through a sequence of open obstacle-free cells. Moreover, from Lemma 4 it follows that in order to decide reachability problem for multi-mode systems with full dimensional and closed obstacles, it suffices to restrict focus to reachability in acyclic cellular paths.

Since for a given convex cell-decomposition, there are finitely many cells, one can enumerate all acyclic cellular paths and check the condition 3 for such a sequence. If this condition doe not hold for any acyclic cell-sequence, then there is no path from source point to the target point. However, if this property holds for an acyclic cell sequence $\langle c_1, c_2, \ldots, c_N \rangle$, we get a sequence of points $\langle \overline{x}_1, \ldots, \overline{x}_N \rangle$ such that $\overline{x}_1 = \overline{x}_s$, $\overline{x}_N = \overline{x}_t$ and all the points are strictly in the interior of the the safety set. For every $1 \leq i < N$ the points $\overline{x}_i$ and $\overline{x}_{i+1}$ are in the interior of the convex set given by $c_i \cup c_{i+1}$ and hence a schedule to reach from $\overline{x}_i$ to $\overline{x}_{i+1}$ can be computed using Algorithm 1. This gives us an algorithm to decide the reachability for full-dimensional and closed obstacles. □

Algorithm 2: BOUNDEDMOTIONPLAN$(\mathcal{H}, \mathcal{W}, \mathcal{O}, \overline{x}_s, \overline{x}_t)$

```
Input: MMS $\mathcal{H} = (M, n, R)$, two points $\overline{x}_s, \overline{x}_t$,
       workspace $\mathcal{W}$, obstacle set $\mathcal{O}$
Output: NO if no $S$-safe schedule from $\overline{x}_s$ to $\overline{x}_t$ exists
        and otherwise such a schedule.
1 $k \leftarrow 0$;
2 Let $B$ be an upper bound on number of cells in a
    cell-decomposition.
3 while $k \leq B$ do
4     Check if the following formula is satisfiable:
          $\exists_{1 \leq i \leq N} \overline{x}_i \quad \exists_{1 \leq i \leq N, m \in M} t_i^{(m)}$ such that
              $(\overline{x}_s = \overline{x}_1 \wedge \overline{x}_t = \overline{x}_N)$
          $\bigwedge_{i=2}^{N}$   $\left( \overline{x}_i = \overline{x}_{i-1} + \sum_{m \in M} R(m) t_i^{(m)} \right)$
          $\bigwedge_{1 \leq i \leq N, m \in M}$   $t_i^{(m)} \geq 0$
          $\bigwedge_{i=2}^{N}$   ObstacleFree$(\overline{x}_{i-1}, \overline{x}_i)$
      if not satisfiable then $k \leftarrow k + 1$ ;
5     else
6         Let $\sigma$ be an empty sequence;
7         for $i = 1$ to $k - 1$ do
8             $\sigma = \sigma :: $ Reach_Convex$(\mathcal{H}, \overline{x}_i, \overline{x}_{i+1}, S)$
9         return $\sigma$;
```

The algorithm implicit in the proof of Theorem 3 requires one to compute the cell-decomposition in advance, and enumerate sequence of cells in order to decide reachability. We next present a bounded-model checking [9] inspired algorithm that implicitly enumerates increasing length sequences of cells till the upper bound on number of cells is reached, or a safe schedule from the source point to the target point is discovered. The key idea is to guess a sequence of points $\overline{x}_1, \ldots, \overline{x}_N$ starting from the source point and ending in the target point such that for every $1 \leq i < N$ the point $\overline{x}_{i+1}$ is reachable from $\overline{x}_i$ using rates provided by the multi-mode system. Moreover, we need to check that the line segment connecting $\overline{x}_i$ and $\overline{x}_{i+1}$ does not intersect with obstacles, i.e:

$$\forall_{0 \leq \lambda \leq 1} \lambda \overline{x}_i + (1 - \lambda) \overline{x}_{i+1} \notin \cup_{j=1}^{k} \mathcal{O}_j.$$

We write ObstacleFree$(\overline{x}_i, \overline{x}_{i+1})$ for this condition. Algorithm 2 sketches a bounded-step algorithm to decide reachability for multi-mode systems which always terminates for multi-mode systems with with full-dimension and closed obstacles thanks to theorem 3.

Notice that at line 4 of the algorithm, we need to check the feasibility of the constraints system which is of the form

$\exists X \forall Y F(X, Y)$ where universal quantifications are implicit in the test for ObstacleFree. If the solver we use to solve the constraints has full support to solve the $\forall$ quantification, we can use that to solve the above constraint. In our experiments, we used Z3 solver (`https://github.com/Z3Prover/z3`) in python to implement the Algorithm 2 and found that the solver was unable to solve in some cases. Fortunately, the universal quantification in our constraints is of very special form and can be easily removed using Fourier-Motzkin Elimination procedure which results in quadratic constraints that are efficiently solvable by Z3 solver. In Section 6 we present the experimental results on some benchmarks to demonstrate scalability of our algorithm.

5. UNDECIDABILITY

Now we focus our attention to the general motion planning problem for multi-mode systems with arbitrary obstacle sets characterized using linear inequalities. We prove the following theorem.

Theorem 5. *Given a constant-rate multi-mode system $\mathcal{H}$, convex workspace $\mathcal{W}$, obstacles set $\mathcal{O}$, start state $\overline{x}_s$ and target state $\overline{x}_t$, the reachability problem* $\text{Reach}(\mathcal{H}, S_{\mathcal{W},\mathcal{O}}, \overline{s}_s, \overline{s}_t)$ *is in general undecidable.*

Proof. We show undecidability of the motion planning problem by giving a reduction from the undecidable halting problem for two-counter machines.

A *two-counter machine* (Minsky machine) $\mathcal{A}$ is a tuple (L, C) where: $L = \{\ell_0, \ell_1, \dots, \ell_n\}$ is the set of instructions and $C = \{c_1, c_2\}$ is the set of two *counters*. There is a distinguished terminal instruction ℓ_n called HALT and the instructions L are one of the following types:

increment $\ell_i : c := c + 1$; goto ℓ_k,

decrement $\ell_i : c := c - 1$; goto ℓ_k,

zero-test ℓ_i : if $(c{>}0)$ then goto ℓ_k else goto ℓ_m,

Halt ℓ_n : HALT.

where $c \in C$, $\ell_i, \ell_k, \ell_m \in L$. In the following, we replace ℓ_n by ℓ_{halt}. Let I, D, O represent the set of increment, decrement and zero-check instructions s.t. $L = I \cup D \cup O$.

A configuration of a two-counter machine is a tuple (ℓ, c, d) where $\ell \in L$ is an instruction, and c, d are natural numbers that specify the value of counters c_1 and c_2, respectively. The initial configuration is $(\ell_0, 0, 0)$. A run of a two-counter machine is a (finite or infinite) sequence of configurations $\langle k_0, k_1, \dots \rangle$ where k_0 is the initial configuration, and the relation between subsequent configurations is governed by transitions between respective instructions. The run is a finite sequence if and only if the last configuration is the terminal instruction ℓ_{halt}. Note that a two-counter machine has exactly one run starting from the initial configuration. We assume without loss of generality that ℓ_0 is an increment instruction. Clearly, it cannot be a decrement instruction. If ℓ_0 were a zero check instruction, we add two dummy instructions ℓ_0', ℓ_0'' such that ℓ_0' increments a counter, and ℓ_0'' decrements it, and passes control to ℓ_0. The dummy instructions ℓ_0', ℓ_0'' will never again be encountered in the two counter machine after control passes to ℓ_0.

The *halting problem* for a two-counter machine asks whether its unique run ends at the terminal instruction ℓ_{halt}. It is well known that the halting problem for two-counter machines is undecidable.

Given a two counter machine $\mathcal{A}$ having instructions $L = \ell_1, \dots, \ell_{n-1}, \ell_{halt}$, we construct a MMS having a number of modes and variables polynomial in n. The idea is to simulate the two-counter machine in the MMS by going through a sequence of modes such that a target point is reachable iff the counter machine halts. We will give a brief sketch of our construction.

- **Modes.** For every increment/decrement instruction ℓ_i we have two modes $\mathcal{M}_i$ and $\mathcal{M}_{ik}$, where k is the index of the unique instruction ℓ_k to which the control shifts in $\mathcal{A}$ from ℓ_i. Corresponding to a zero check instruction ℓ_i, we have four modes $\mathcal{M}_i^1, \mathcal{M}_i^2, \mathcal{M}_{ik}$ and $\mathcal{M}_{im}$, where k, m are respectively the indices of the unique instructions ℓ_k, ℓ_m to which the control shifts from ℓ_i depending on whether the counter value is >0 or $=0$. There are three modes $\mathcal{M}_{halt}, \mathcal{M}_{halt}^{c_1}$ and $\mathcal{M}_{halt}^{c_2}$ corresponding to the halt instruction. Let $\mathcal{I}$ be a unique mode where the computation is forced to begin, to ensure that we do not run into an obstacle.

- **Variables.** There are two variables $C = \{c_1, c_2\}$ corresponding to the two counters. There is a unique variable $S = \{s_0\}$, whose rate is 0 at all modes other than the mode $\mathcal{I}$. For $\ell_i \in I \cup D$, there are variables n_{ij}, x_{ij}, where j is the index of the unique instruction ℓ_j to which control shifts from ℓ_i.

  - Let $N = \{n_{ij}, n_{i\ halt} \mid 0 \le i, j \le n \text{ and } \ell_i \notin O\}$,
  - $X = \{x_{ij} \mid 0 \le i, j \le n\}$, and
  - $Z = \{z_{i\#} \mid \ell_i \in O\}$.

  The set of variables used is $\mathcal{V} = X \cup N \cup Z \cup C \cup S \cup \{n_{halt}\}$. Let X_{halt} be the subset of X consisting of variables of the form $x_{i\ halt}$. The total number of variables and modes is $\mathcal{O}(n^2)$, where n is the number of instructions in the two counter machine.

- **Rates at Modes.**

  1. Variable rates at mode $\mathcal{I}$ are such that $R(\mathcal{I})(s_0) = -1, R(\mathcal{I})(n_{0j}) = 1$, while $R(\mathcal{I})(v) = 0$ for all variables v other than s_0 and n_{0j}. Here j is the index of the unique instruction ℓ_j to which the control shifts from the initial instruction ℓ_0 (recall that ℓ_0 is an increment instruction, and hence control shifts deterministically to some ℓ_j).
  2. Assume $\ell_i \in I \cup D$ is an increment/decrement instruction for counter $c_1(c_2)$ and let ℓ_j be the resultant instruction. The rates of variables at mode $\mathcal{M}_i$ are $R(\mathcal{M}_i)(n_{ij}) = -1, R(\mathcal{M}_i)(x_{ij}) = 1$, and $R(\mathcal{M}_i)(v) = 0$ for $v \neq c_1$ (c_2), while $R(\mathcal{M}_i)(c_1) = 1$ $(R(\mathcal{M}_i)(c_2) = 1)$ if $\ell_i \in I$ and $R(\mathcal{M}_i)(c_1) = -1$ $(R(\mathcal{M}_i)(c_2) = -1)$ if $\ell_i \in D$.
  3. For $\ell_i \in O$ (zero-check) the rates of variables in the modes $\mathcal{M}_i^1, \mathcal{M}_i^2, \mathcal{M}_{ik}, \mathcal{M}_{im}$ are
     (a) $R(\mathcal{M}_i^1)(z_{i\#}) = -1, R(\mathcal{M}_i^1)(x_{ik}) = 1$, and we have $R(\mathcal{M}_i^1)(v) = 0$ for all other variables v.

(b) $R(\mathcal{M}_i^2)(z_{i\#}) = -1, R(\mathcal{M}_i^2)(x_{im}) = 1$, and we have $R(\mathcal{M}_i^2)(v) = 0$ for all other variables v.

4. We have the modes $\mathcal{M}_{ij}$ for $i \in I \cup D \cup O$. The rates of variables at mode $\mathcal{M}_{ij}$, $j \neq halt$ are
   (a) $R(\mathcal{M}_{ij})(x_{ij}) = -1$
   (b) If ℓ_j is not a zero check instruction, then there is a unique instruction ℓ_k to which the control will shift from ℓ_j. Then $R(\mathcal{M}_{ij})(n_{jk}) = 1$, while $R(\mathcal{M}_{ij})(v) = 0$ for all variables $v \notin \{n_{jk}, x_{ij}\}$.
   (c) If ℓ_j is a zero check instruction, then we have $R(\mathcal{M}_{ij})(z_{j\#}) = 1$, while $R(\mathcal{M}_{ij})(v) = 0$ for all variables $v \notin \{z_{j\#}, x_{ij}\}$.
5. The rates of variables at mode $\mathcal{M}_{i\ halt}$ are as follows: $R(\mathcal{M}_{ihalt})(x_{i\ halt}) = -1$, $R(\mathcal{M}_{i\ halt})(n_{halt}) = 1$, while all other variables have rate 0. The rates at modes $\mathcal{M}_{halt}, \mathcal{M}_{halt}^{c_1}, \mathcal{M}_{halt}^{c_2}$ are given by :
   - $R(\mathcal{M}_{halt})(n_{halt}) = -1$ and $R(\mathcal{M}_{halt})(v) = 0$ for all other variables.
   - $R(\mathcal{M}_{halt}^{c_1})(c_1) = -1$, $R(\mathcal{M}_{halt}^{c_1})(n_{halt}) = 1$ while $R(\mathcal{M}_{halt}^{c_1})(v) = 0$ for all other variables.
   - $R(\mathcal{M}_{halt}^{c_2})(c_2) = -1$, $R(\mathcal{M}_{halt}^{c_2})(n_{halt}) = 1$ while $R(\mathcal{M}_{halt}^{c_2})(v) = 0$ for all other variables.

A simulation of the two counter machine going through instructions $\ell_0, \ell_1, \ell_2, \ldots, \ell_y, \ell_{halt}$ is achieved by going through modes $\mathcal{I}, \mathcal{M}_0, \mathcal{M}_{01}, \mathcal{M}_1$ or $\mathcal{M}_1^1$ or $\mathcal{M}_1^2 \ldots, \mathcal{M}_y, \mathcal{M}_{y\ halt}$ in order, spending exactly one unit of time in each mode. Starting from a point with $s_0 = 1$ and $v = 0$ for all variables v other than s_0, we want to reach a point where $n_{halt} = 1$ and $v = 0$ for all variables v other than n_{halt}. The idea is to start in mode $\mathcal{I}$, and spending one unit of time in $\mathcal{I}$ obtaining $s_0 = 0, n_{01} = 1$. Growing n_{01} represents that the current instruction is ℓ_0, and the next one is ℓ_1. Next, we shift to mode $\mathcal{M}_0$, spend one unit of time there to obtain $x_{01} = 1, n_{01} = 0$. This is followed by mode $\mathcal{M}_{01}$, where x_{01} becomes 0, and one of the variables $z_{1\#}, n_{12}$ attain 1, depending on whether ℓ_1 is a zero check instruction or not.

In general, while at a mode $\mathcal{M}_{ij}$, the next instruction ℓ_k after ℓ_j is chosen by "growing" the variable n_{jk} if ℓ_j is not a zero-check instruction, or by "growing" the variable $z_{j\#}$ if ℓ_j is a zero-check instruction. In parallel, x_{ij} grows down to 0, so that $x_{ij} + n_{jk} = 1$ or $x_{ij} + z_{j\#} = 1$.

– In the former case, the control shifts from $\mathcal{M}_{ij}$ to mode $\mathcal{M}_j$ where variable x_{jk} is grows at rate 1 while n_{jk} grows at rate -1, so that $x_{jk} + n_{jk} = 1$. Control shifts from $\mathcal{M}_j$ to $\mathcal{M}_{jk}$, where the next instruction ℓ_g after ℓ_k is chosen by growing variable n_{kg} if ℓ_k is not zero-check instruction, or the variable $z_{k\#}$ is grown if ℓ_k is a zero-check instruction.

– In the latter case, one of the modes $\mathcal{M}_j^1, \mathcal{M}_j^2$ is chosen from $\mathcal{M}_j$ where $z_{j\#}$ grows at rate -1. Assume ℓ_j is the instruction "If the counter value is > 0, then goto ℓ_m, else goto ℓ_h". If $\mathcal{M}_j^1$ is chosen, then the variable x_{jm} grows at rate 1 while if $\mathcal{M}_j^2$ is chosen, then the variable x_{jh} grows at rate 1. In this case, we have $z_{j\#} + x_{jm} = 1$ or $z_{j\#} + x_{jh} = 1$. From $\mathcal{M}_j^1$, control shifts to $\mathcal{M}_{jm}$, while from $\mathcal{M}_j^2$, control shifts to $\mathcal{M}_{jh}$.

Continuing in the above fashion, we eventually reach mode $\mathcal{M}_{y\ halt}$ where $x_{y\ halt}$ grows down to 0, while the variable n_{halt} grows to 1, so that $x_{y\ halt} + n_{halt} = 1$. It remains to use the modes $\mathcal{M}_{halt}, \mathcal{M}_{halt}^c$ as many times to obtain $c_1 = 0, c_2 = 0$ and $n_{halt} = 1$.

Workspace and the Set of Obstacles. Instead of describing the obstacles directly, we describe its complement, and call it a *non-hitting zone*. The non-hitting zone consists of points in the Euclidean space satisfying $\bigwedge_{t=a}^{g} \varphi_t$ where

(φ_a) *Init*: $0 \le y \le 1$ for $y \in N \cup S \cup X \cup Z$, $n_{halt}, c_1, c_2 \ge 0$

(φ_b) *Mutex*(X) :

$$\bigwedge_{i,j} (x_{ij} > 0 \Rightarrow \bigwedge_{\substack{(g \neq i) \vee \\ (f \neq j)}} x_{gf} = 0)$$

(φ_c) *Mutex*(N, Z) :

$$\bigwedge_{i,j} (n_{ij} > 0 \Rightarrow (\bigwedge_{\substack{(g \neq i) \vee \\ (f \neq j)}} n_{gf} = 0 \wedge \bigwedge_{g} z_{g\#} = 0))$$

(φ_d) *Mutex*(Z, N):

$$\bigwedge_{i} (z_{i\#} > 0 \Rightarrow (\bigwedge_{k \neq i} z_{k\#} = 0 \wedge \bigwedge_{g,f} n_{gf} = 0))$$

(φ_e) *Mutex*(S, X):

$$s_0 > 0 \Rightarrow \bigwedge_{i,j} x_{ij} = 0$$

(φ_f) *Mutex*$(n_{halt}, X \cup N \cup Z \cup S)$:

$$n_{halt} > 0 \Rightarrow \bigwedge_{y \in \mathcal{V} \setminus (X_{halt} \cup C)} (y = 0)$$

(φ_g) *Sum*(X_{halt}, n_{halt}):

$$\bigwedge_{i} (x_{i\ halt} > 0 \Rightarrow (x_{i\ halt} + n_{halt} = 1))$$

An obstacle $\mathcal{O}$ is thus one which satisfies $\bigvee_{t=a}^{g} \neg \varphi_t$. As an example, $\mathcal{O}_{i\ halt} = x_{i\ halt} > 0 \wedge (x_{i\ halt} + n_{halt} \neq 1)$ is an obstacle obtained by negating φ_g. Note that the obstacles are not closed.

Key Lemmas for Reachability. A point in the Euclidean space is represented as a tuple of values to all variables. We start with s_0 as the first variable in the tuple, c_1, c_2 as the next two variables, and n_{halt} as the last variable in the tuple. On the MMS constructed as above, we show that starting from the initial mode $\mathcal{I}$ from a point $(1, 0, 0, \ldots, 0)$, it is possible to safely reach the point $(0, 0, 0, \ldots, 0, 1)$ iff the two counter machine halts.

The following lemmas establish that (1) The computation begins in mode $\mathcal{I}$, and exactly one unit of time is spent in $\mathcal{I}$ (Lemma 6), (2) the order of choosing modes is decided by the sequence of instructions chosen in a correct simulation of the two counter machine (Lemmas 7, 8, 9), and (3) starting from $s_0 = 1$ and $v = 0$ for all variables $v \neq s_0$, one can reach the point with $n_{halt} = 1$ and $v = 0$ for all variables $v \neq n_{halt}$ avoiding obstacles iff (1), (2) are true. This is shown by Lemma 10. The proofs of all lemmas can be found in the Appendix A.

LEMMA 6. *Exactly one unit of time is spent at $\mathcal{I}$, and the computation starts in $\mathcal{I}$.*

LEMMA 7. *The modes chosen in order after $\mathcal{I}$ are $\mathcal{M}_0$ and $\mathcal{M}_{0j}$, where ℓ_j is the unique instruction that follows ℓ_0 in the two counter machine. The times spent at $\mathcal{M}_0$ and $\mathcal{M}_{0j}$ are exactly 1.*

Now if j was not a zero check instruction, and has ℓ_k as the successor of ℓ_j, then as seen above in Lemma 7 in the case of $\mathcal{M}_0$ and $\mathcal{M}_{0j}$, we can show that the control has to shift to $\mathcal{M}_j$ from $\mathcal{M}_{0j}$. If j is a zero check instruction, then we claim that the control has to switch from $\mathcal{M}_{0j}$ to one of $\mathcal{M}^1_j$ or $\mathcal{M}^2_j$. Lemma 8 generalises this claim.

LEMMA 8. *If the current control resides in a mode $\mathcal{M}_{gf}$, without hitting any obstacle, then the control shifts to $\mathcal{M}_f$ if f is not a zero check instruction. If f is a zero check instruction, then control shifts to $\mathcal{M}^1_f$ or $\mathcal{M}^2_f$. The time spent at $\mathcal{M}_{gf}$ is 1, and the times spent respectively at $\mathcal{M}_f$ ($\mathcal{M}^1_f$ or $\mathcal{M}^2_f$) are also 1.*

Thus, it can be seen that some obstacle is hit if a time other than one is spent at any mode, or if a mode violating the order of instructions in the two counter machine is chosen. Lemmas 6, 7 and 8 prove this. Assume now that the mode switching happens respecting the instruction flow in the two counter machine, and one unit of time is spent at each mode. It can be seen that two counter machine halts iff some mode $\mathcal{M}_{i\ halt}$ is reached. After spending one unit of time at $\mathcal{M}_{i\ halt}$, we obtain $n_{halt} = 1$, and all x variables 0. Note that by condition φ_f, no x variables other than $x_{i\ halt}$ can be non-zero when $n_{halt} > 0$.

LEMMA 9. *Switching out of $\mathcal{M}_{i\ halt}$ to another mode will violate safety unless one unit of time is spent at $\mathcal{M}_{i\ halt}$.*

LEMMA 10. *(0,0,0,...,1) is reachable from (1,0,0, ...,0) iff the two counter machine halts.*

The proof is now complete. □

6. EXPERIMENTAL RESULTS

In this section, we present experimental results with an implementation of the algorithm 2. In order to show competitiveness of our algorithm, we compare its performance against RRT algorithm [19] on a set of micro-benchmarks. Before we introduce the experimental setup, we explain the framework of RRT algorithms.

6.1 RRT Algorithm

Rapidly-exploring Random Tree (RRT) [19] is a space-filling data structure which is used to search a region by incrementally building a tree. It is constructed by selecting random points in the state space and can provide better coverage of reachable states of a system than mere simulations. RRT is represented as a DAG, $G = (V, E)$, where each edge $e_{ij} \in E$ is labelled by the control input (in our terminology, control inputs are called rate vectors of the modes $R(m_i)\ \forall i : 1 \leq i \leq |M|$) used to reach j from i. There are many versions of RRTs available; we use the standard version [19] as sketched in Algorithm 3.

The GENERATE-RANDOM procedure samples a point in the continuous state space randomly. The procedure NEAREST-NEIGHBOUR finds the nearest node to the random point in the RRT based on some distance metric ρ. MIN-CONTROL-SIGNAL procedure finds the control input which when applied to the nearest node x_{near} for a given time step Δt minimises the distance of the random point from the resultant point. EXTEND procedure finds the resultant node which is a candidate to be added into the RRT and the after chacking some safety conditions in the VALID procedure, the node is added into the tree.

Algorithm 3: RRT

```
Input: Starting vertex $x_0$, target point $\mathcal{T}$, error bound
       $\varepsilon$, distance metric $\rho$
Output: RRT G = (V, E)
1  $V \leftarrow x_0$, $E \leftarrow \phi$ ;
2  while $min_{x \in V} \rho(x, \mathcal{T}) \geq \varepsilon$ do
3      $x_{rand}$ = GENERATE-RANDOM();
4      $x_{near}$ = NEAREST-NEIGHBOUR($x_{rand}, V, \rho$);
5      $u_{min}$ = MIN-CONTROL-SIGNAL($x_{rand}, x_{near}$);
6      $x_{new}$ = EXTEND($x_{near}, u_{min}$);
7      if VALID($x_{new}, u_{min}$) then
8          $V \leftarrow V \cup x_{new}$;
9          $E \leftarrow E \cup (x_{new}, x_{near}, u_{min})$;
10     else
11         continue;
```

6.2 Experimental Setup

The implementation and experiments with Algorithm 2 and 3 were performed on a computer running ubuntu-14.10 OS, with an Intel Core i7-4510 2.00 GHz quadcore CPU, with 8GB RAM. We performed comparisons of both algorithms by executing them for the following set of microbenchmarks. Although, all of the obstacles in our microbenchmarks are hyper-rectangular, our algorithms can also handle general polyhedral obstacles.

- **L-shaped obstacle.** This class of microbenchmarks contains examples with hyper-rectangular workspace and certain "L" shaped obstacles as shown in the figure 1. The initial vertex is the lower left vertex of the square ($\overline{x}_s$) and the target is the right upper vertex of the square ($\overline{x}_t$). Our algorithm can give the solution to this problem with bound $k = 2$ returning the sequence $\langle \overline{x}_1, \overline{x}, \overline{x}_t \rangle$ as shown in the figure, while RRT algorithm in this case samples most of the points which lie on the other side of the obstacles and if the control modes are not in the direction of the line segments $\overline{x}_1\overline{x}$ and $\overline{x}\overline{x}_t$, then it grows in arbitrary directions and hits the obstacles a large number of times, leading to rather large number of iterations slowing the growth of the tree. In the cases where the rate of the modes are along the directions of the above-mentioned line segments RRT computes the solution relatively faster.

  We experimented with L-shaped examples for dimensions ranging from 2 to 9. In most of the cases, we found that the RRT algorithm fails to terminate in dimensions larger than 4. Even with 2 or 3 dimensions, it takes substantially long time to terminate than our algorithm. The details and observations for this test case are summarized in Table 1.

- **Maze-Shaped Obstacles.** The motivation to study these microbenchmarks comes from motion planning problems in regular environments. The arena has rectangular obstacles coming from the top and the bottom

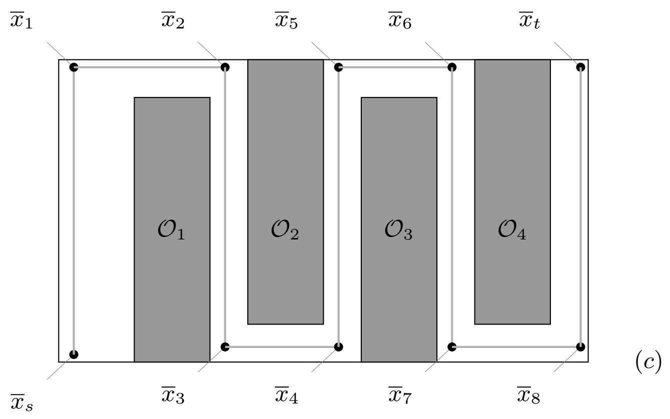


Figure 2: Maze-shapred arena with four obstacles

(as shown in Figure 2 for two dimensions) alternately. The starting point is the lower left vertex $\overline{x}_s$ and the target point is $\overline{x}_t$. A sample free-path through the arena is also shown in the figure. RRT algorithm performs well for lower dimensions but fails to terminate for higher dimensions. The results for this class of obstacles are summarised in Table 2. Experiments were performed for upto 6 dimensions and 4 obstacles.

– **Randomly generated rectangular obstacles:** We designed an microbenchmark generator tool which could take as input the size of the arena and the number of obstacles needed and returns two randomly placed non-overlapping rectangular obstacles inside the arena. We generated cases with up to 4 rectangular obstacles on a hypercubic arena with side lengths of 10 and 20, and tested the performance of the two approaches on them. The RRT algorithm again performed well for smaller dimensions ($\sim 2, 3$) but the performance of RRT was severely affected by the size of the arena. For an arena size of 20, RRT did not terminate in most of the cases. These trends are summarised in Table 3.

6.3 Discussion

In our experiments we found that RRT does not do scale up in high dimensions and could not solve reachability to ε-bounds in these cases. On the other hand, our algorithm can give reachability bounds for upto 9 dimensions on L-shaped obstacles, within the timeout limit of 40 minutes. Moreover, the performance of RRT is severely affected if the arena size is large. On the other hand, our algorithm is independent of the arena size making it useful for motion planning problems.

Another interesting pattern we observed was that if the arena is filled up with more obstacles covering more space, then the probability of a randomly sampled point lying in the obstacle region is high and this makes RRT ineffective in this situation by wasting a lot of iterations.

Our algorithm performs better in situations when it terminates early (target reachable from source in relatively fewer "hops") and the performance of our algorithm degrades as number of hops or dimensions increase due to the Fourier-Motzkin procedure implemented in our algorithm introduced exponentially many constraints as function of the dimension exhibiting the curse of dimensionality.

Dim.	RRT	BoundedMotionPlan
2	380	0.09
3	582.3	1.2
4	1007.2	9.6
5	2143.37	39.3
6	TO	86.2
7	TO	382
8	TO	563.2
9	TO	1723.7

Table 1: Table showing trends for the case of L-shaped obstacles with varying number of dimensions where TO represents timeout (> 40 minutes)

Dim.	$\mathbf{N_{obst}}$	RRT	BoundedMotionPlan
2	2	220	0.2
2	3	253.32	0.26
2	4	302.3	0.647
3	2	627.3	1.53
3	3	628.1	2.17
3	4	643	3.02

Table 2: Table showing trends for the case of Maze obstacles with varying number of dimensions

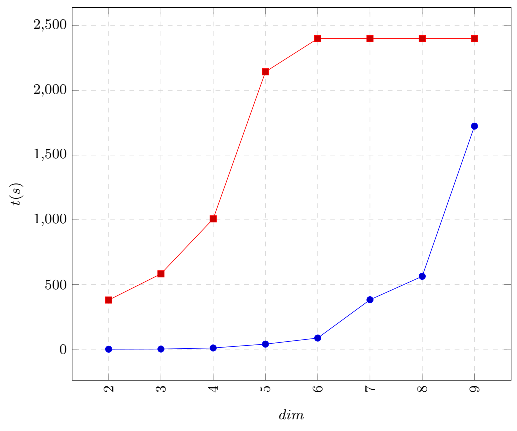
Dim.	RRT	BoundedMotionPlan
2	110	2.13
3	248.67	8.65
4	1021.3	32.6
5	TO	35.4
6	TO	252.5
7	TO	896

Table 3: Table showing trends for the case of random obstacles with varying number of dimensions (TO indicates timeout (> 40 minutes))

7. CONCLUSION

In this paper we studied the motion planning problem for constant-rate multi-mode system with non-convex safety sets given as a convex set of obstacles. We showed that while the general problem is already undecidable in this simple setting of linearly defined obstacles, decidability can be recovered by making appropriate (full-dimensionality and closed) assumption on the obstacles. Moreover, our algorithm performs satisfactorily when compared to well known algorithms for motion planning, and can easily be adapted to provide semi-algorithms for motion-planning problems for objects with polyhedral shapes.

Our algorithm can also be used as a complete procedure for path planning in order to solve motion planning for systems with general hybrid dynamics as along as we can show that the system is controllable. General controllability theorems (such as Kalman controllability theorem) can be used to verify controllability of general hybrid systems.



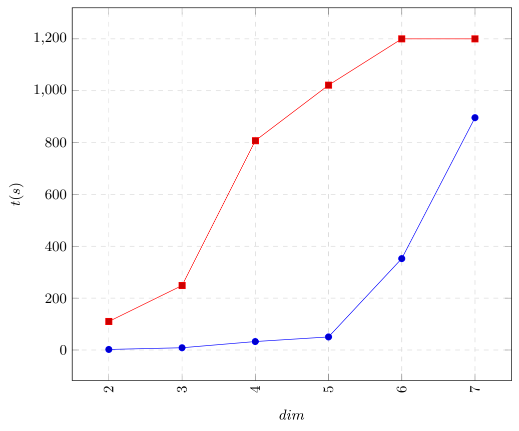


Figure 3: On the left we plot performance for the L-shaped Obstacles and on the right we plot performance for the rectangular-shaped randomly placed Obstacles. Here execution times for RRTs are shown in red and for our algorithm is shown in blue. The flatness on the top of the RRT plots indicate timeouts.

8. REFERENCES

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-S. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

[2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] R. Alur, V. Forejt, S. Moarref, and A. Trivedi. Safe schedulability of bounded-rate multi-mode systems. In *HSCC*, pages 243–252, 2013.

[4] R. Alur, A. Trivedi, and D. Wojtczak. Optimal scheduling for constant-rate multi-mode systems. In *HSCC*, pages 75–84, 2012.

[5] E. Asarin, M. Oded, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *TCS*, 138:35–66, 1995.

[6] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *Automatic Control*, 43(1):31–45, 1998.

[7] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 49–60, Oct 1987.

[8] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988.

[9] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19(1):7–34, 2001.

[10] Robert James Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, New Haven, CT, USA, 1989. AAI9010653.

[11] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Robust hybrid control for autonomous vehicle motion planning. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 1, pages 821–826. IEEE, 2000.

[12] Erann Gat. Three-layer architectures. In David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 195–210. MIT Press, Cambridge, MA, USA, 1998.

[13] Google self-driving car project, monthly report—april 2016. `https://static.googleusercontent.com/media/www.google.com/lt//selfdrivingcar/files/reports/report-0416.pdf`. Retrived on May 31, 2016.

[14] T. A. Henzinger. The theory of hybrid automata. In *LICS' 96*, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.

[15] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *TCS*, 221(1-2):369–392, 1999.

[16] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Comp. and Sys. Sciences*, 57:94–124, 1998.

[17] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, Nov 2015.

[18] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[19] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[20] J.L. Le Ny and G.J. Pappas. Sequential composition of robust controller specifications. In *Robotics and Automation*, pages 5190–5195, 2012.

[21] Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam. Apex: A tool for autonomous vehicle plan verification and execution. In *In Society of Automotive Engineers (SAE) World Congress and Exhibition*, 2016.

[22] Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[23] J. Reif. Complexity of the mover's problem and generalizations. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 00(undefined):421–427, 1979.

[24] Jacob T Schwartz and Micha Sharir. On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4(3):298–351, September 1983.

APPENDIX

A. OMITTED PROOFS

A.1 Proof of Lemma 6

Proof. We claim that a time of 1 unit has to be spent at $\mathcal{I}$, and that the computation starts in mode $\mathcal{I}$. We start with the point where $s_0 = 1$ and all other variables are 0. Assume we start in a mode other than $\mathcal{I}$ and spend t units of time there. If we start in some mode $\mathcal{M}_i$ and spend time t, then we will obtain $n_{ij} = -t$ for some j, hitting the obstacle $\neg Init$ (violating φ_a). Likewise, if we start in some mode $\mathcal{M}_i^1$ or $\mathcal{M}_i^2$, then again we will obtain $z_{i\#} < 0$, again hitting the obstacle $\neg Init$. Likewise, starting in mode $\mathcal{M}_{i\ halt}$ or $\mathcal{M}^c_{halt}$ or $\mathcal{M}_{halt}$ will give $x_{i\ halt} < 0$ or $c < 0$ or $n_{halt} < 0$ all of which will hit the obstacle $\neg Init$. Thus, we have to start in $\mathcal{I}$.

Now we show that exactly one unit of time must be spent in $\mathcal{I}$.

1. If more than one unit of time is spent at $\mathcal{I}$, then $s_0 < 0$ will violate φ_a.
2. Assume that a time $t < 1$ is spent at $\mathcal{I}$, and the control shifts to any other mode. Then we have $s_0 = 1 - t$ and $n_{0j} = t$ on entering that mode. Note that $n_{0j} + s_0 = 1$. If any time is spent at that mode, then we will either obtain $s_0 > 0$ and some $x_{ij} \neq 0$ (hits $\neg Mutex(S, X)$ and violates φ_e) or some $n_{jk}, z_{k\#} < 0$ (hits $\neg Init$ and violates φ_a).

Hence, exactly one time unit is spent at $\mathcal{I}$, and the control shifts. □

A.2 Proof of Lemma 7

Proof. After spending one unit of time in $\mathcal{I}$, we have $s_0 = 0$ and $n_{0j} = 1$. We claim that the control will switch to mode $\mathcal{M}_0$ from $\mathcal{I}$.

1. Recall that $R(\mathcal{M}_0)(n_{0j}) = -1$, and $R(\mathcal{M}_0)(x_{0j}) = 1$, where j is the unique index of the instruction ℓ_j to which control shifts in the two counter machine from ℓ_0. If control switches to $\mathcal{M}_0$, and $0 \leq t \leq 1$ time is spent, then we have $n_{0j} = 1 - t$, $x_{0j} = t$. The resultant points are all safe; note that $n_{0j} + x_{0j} = 1$.
2. If control switches from $\mathcal{I}$ to some $\mathcal{M}_k$, $k \neq 0$, or some $\mathcal{M}_k^1$ (or $\mathcal{M}_k^2$) or some $\mathcal{M}_{gf}$, then we have
   - $R(\mathcal{M}_k)(n_{kg}) = -1$, and $R(\mathcal{M}_k)(x_{kg}) = 1$ for some g, or
   - $R(\mathcal{M}_k^1)(z_{k\#}) = -1$, and $R(\mathcal{M}_k)(x_{kg}) = 1$ for some g.
   - $R(\mathcal{M}_{gf})(x_{gf}) = -1$.

   If $t > 0$ time is spent at $\mathcal{M}_k$, or $\mathcal{M}_k^1$ ($\mathcal{M}_k^2$) or $\mathcal{M}_{gf}$, then we obtain $n_{kg} < 0$ or $z_{k\#} < 0$ or $x_{gf} = -t < 0$, violating the non-hitting zone.

Next we claim that exactly one unit of time is spent at $\mathcal{M}_0$ before control switches to any other mode. By Lemma 6, we have $s_0 = 0, n_{0j} = 1$ on entering $\mathcal{M}_0$.

1. It is easy to see that if time $t > 1$ is spent at $\mathcal{M}_0$, the obstacle $\neg Init$ is hit, since $x_{0j} > 1$.
2. Assume now that t time is spent at $\mathcal{M}_0$, and control switches to some mode. Then we have $n_{0j} = 1-t, x_{0j} = t$ on exiting $\mathcal{M}_0$.
   - If mode $\mathcal{M}_{0j}$ is chosen after $\mathcal{M}_0$, and a time $t' > 0$ is spent at $\mathcal{M}_{0j}$, then we obtain $x_{0j} = t - t'$, $n_{0j} = 1 - t$. Also, we have $n_{jk} = t' > 0$ or $z_{j\#} = t' > 0$, depending on whether ℓ_j is not a zero check instruction having ℓ_k as its successor, or ℓ_j is a zero check instruction. In the former case, we obtain $n_{0j} > 0$ and $n_{jk} > 0$, violating φ_c, while in the latter case, we obtain $n_{0j} > 0$ and $z_{j\#} > 0$, again violating φ_c. However, if $t = 1$ time is spent at $\mathcal{M}_0$, then there is no violation since we have $n_{0j} = 0, x_{0j} = 1 - t'$ and exactly one of $n_{jk} = t'$ or $z_{j\#} = t'$.
   - Assume that a time $t = 1$ is spent at $\mathcal{M}_0$, but a mode other than $\mathcal{M}_{0j}$ is chosen from $\mathcal{M}_0$ and a time $t' > 0$ is spent there.
     - If $\mathcal{M}_k$ is chosen for some k, then we obtain $x_{kg} = t'$. This violates φ_b since $x_{kg} = t'$ and $x_{0j} = 1$.
     - If $\mathcal{M}_{kg}$ is chosen for some $k, g \neq 0, j$, then we obtain $n_{gh} = t'$ or $z_{g\#} = t'$ along with $x_{kg} = -t'$. This violates φ_a since $x_{kg} = -t' < 0$.

Thus, we have seen that starting from $\mathcal{I}$, the control shifts to $\mathcal{M}_0$ and then to $\mathcal{M}_{0j}$ in order, spending exactly one unit of time at $\mathcal{M}_0$. We now argue that the time t' spent at $\mathcal{M}_{0j}$ has to be 1. Assume $t' \neq 1$. Spending t' units of time at $\mathcal{M}_{0j}$ results in $x_{0j} = 1 - t'$ and one of $n_{jk} = t'$ or $z_{j\#} = t'$.

1. If $t' > 1$, then we obtain $x_{0j} = 1 - t' < 0$, violating φ_a.
2. Assume $t' < 1$, and control switches from $\mathcal{M}_{0j}$ to some $\mathcal{M}_f$. Let t'' time be spent at $\mathcal{M}_f$. If $f \neq 0$, then we obtain $x_{fg} = t'' > 0$ for some g, and $x_{0j} = 1 - t' > 0$ violating φ_b. If $f = 0$, then we obtain $x_{0j} = 1 - t' + t''$, but $n_{0j} = -t'' < 0$, violating φ_a.
3. Assume $t' < 1$, and control switches from $\mathcal{M}_{0j}$ to some $\mathcal{M}_f^1$ or $\mathcal{M}_f^2$. Let t'' time be spent at $\mathcal{M}_f^1$ ($\mathcal{M}_f^2$). Then we obtain $z_{f\#} = -t'' < 0$ violating φ_a.
4. Assume $t' < 1$, and control switches from $\mathcal{M}_{0j}$ to some $\mathcal{M}_{cd}$. Then we have one of $z_{d\#} = t'' > 0$ or $n_{dh} = t'' > 0$ along with one of $n_{jk} = t' > 0$ or $z_{j\#} = t' > 0$, both which violate one of φ_c, φ_d.

If $t' = 1$, then we have $x_{0j} = 0$ and one of $n_{jk} = 1$ or $z_{j\#} = 1$. □

A.3 Proof of Lemma 8

Proof. Assume that control resides in $\mathcal{M}_{gf}$, and assume that no obstacles have been hot so far. We know that $R(\mathcal{M}_{gf})(x_{gf}) = -1$. On entry to $\mathcal{M}_{gf}$, we have $x_{gf} = 1$, all variables other than (possibly c_1, c_2) are 0.

1. ℓ_f is not a zero check instruction. Then $R(\mathcal{M}_{gf})(n_{fq}) = 1$ for some unique index q corresponding to the successor ℓ_q of ℓ_f. Spending $t = 1$ here results in $x_{gf} = 0$ and $n_{fq} = 1$, with no violation to the non-hitting zone. As seen in Lemma 7 for the case of $\mathcal{M}_{0j}$, a time t' spent at $\mathcal{M}_{gf}$ is safe iff $t' = 1$ ($t' > 1$ violates safety immediately, while a switch in control from $\mathcal{M}_{gf}$ with $t' < 1$ to any mode disallows spending time at the new mode).
2. ℓ_f is a zero check instruction. Then $R(\mathcal{M}_{gf})(z_{f\#}) = 1$. Spending t unit of time at $\mathcal{M}_{gf}$ results in $x_{gf} = 1 - t$ and $z_{f\#} = t$.
   (a) If $t > 1$, then we obtain $x_{gf} < 0$ violating φ_a.

(b) Assume $t < 1$, and control switches out of $\mathcal{M}_{gf}$.
   - If the next mode chosen is some $\mathcal{M}_k$, and $t' > 0$ units of time spent, then we obtain $n_{k*} = -t' < 0$ violating φ_a.
   - If the next mode chosen is some $\mathcal{M}_k^1$ or $\mathcal{M}_k^2$ with $k \neq f$, and t' units of time spent there, then we obtain $z_{k\#} = -t' < 0$ violating φ_a.
   - If the next mode chosen is some $\mathcal{M}_{cd}$, then we obtain $x_{cd} = -t' < 0$ violating φ_a.
   - If the next mode chosen is $\mathcal{M}_f^1$ or $\mathcal{M}_f^2$, and $t' > 0$ units of time spent, then we obtain $z_{f\#} = t - t'$. However, we also obtain $x_{f*} = t' > 0$ and $x_{gf} = 1 - t > 0$ violating φ_b.

(c) Assume $t = 1$ unit of time is spent at $\mathcal{M}_{gf}$ and control switches out of $\mathcal{M}_{gf}$. We then have $x_{gf} = 0$ and $z_{f\#} = 1$.
   - If the next mode chosen is some $\mathcal{M}_k$, and $t' > 0$ units of time spent, then we obtain $n_{k*} = -t' < 0$ violating φ_a.
   - If the next mode chosen is $\mathcal{M}_j^1$ or $\mathcal{M}_j^2$, $j \neq f$, and $t' > 0$ units of time spent, then we obtain $z_{j\#} = -t' < 0$ violating φ_a.
   - If the next mode chosen is some $\mathcal{M}_{cd}$ and $t' > 0$ units of time spent, then we obtain $x_{cd} = -t' < 0$, violating φ_a.
   - If the next mode chosen is $\mathcal{M}_f^1$ or $\mathcal{M}_f^2$, and $t' > 0$ units of time spent, we obtain $z_{f\#} = 1 - t'$, along with $x_{f*} = t' > 0$. If ℓ_f is the instruction "if $c_1 > 0$, goto f_1, else goto f_2", then * is either f_1 or f_2. There is no violation to safety as long as $c_1 > 0$ and $\mathcal{M}_f^1$ is chosen, or $c_1 = 0$ and $\mathcal{M}_f^2$ is chosen when $t' \leq 1$. If $t' = 1$, then we obtain $z_{f\#} = 0$ and $x_{f*} = 1$. After spending $t' = 1$ unit of time at $\mathcal{M}_f^1$ or $\mathcal{M}_f^2$, assume the control switches to $\mathcal{M}_{f*}$, and a time t'' is spent there. As seen in Lemma 7, it can be shown that there is no violation to safety as long as $t'' \leq 1$. In particular, it can be shown that if $t'' < 1$, and control switches out of $\mathcal{M}_{f*}$, safety is violated.

The proof is now complete. □

A.4 Proof of Lemma 9

PROOF. On entering $\mathcal{M}_{i\ halt}$, we have $x_{i\ halt} = 1$ and $n_{halt} = 0$. Assume $t < 1$ time is spent at $\mathcal{M}_{i\ halt}$ and a mode change happens. Then we have $x_{i\ halt} = 1 - t$ and $n_{halt} = t$.

- If we move to any $\mathcal{M}_k, \mathcal{M}_k^1, \mathcal{M}_k^2$ or $\mathcal{M}_{cd}$, and elapse a time $t' > 0$, we will have a safety violation due to some variable becoming negative (n_{k*} in the case of $\mathcal{M}_k$, x_{cd} in the case of $\mathcal{M}_{cd}$ and $z_{k\#}$ in the case of $\mathcal{M}_k^1, \mathcal{M}_k^2$).
- Assume that we move to $\mathcal{M}_{halt}$ and spend $t' > 0$ time there. Then we obtain $n_{halt} = t - t'$. This violates φ_g since we have $x_{i\ halt} + n_{halt} \neq 1$. Moving to $\mathcal{M}_{halt}^{c1}, \mathcal{M}_{halt}^{c2}$ also violates safety for the same reason.

However, if $t = 1$, then we have $n_{halt} = 1$ and all other x, n variables are 0. Moving to any mode other $\mathcal{M}_{halt}$ or $\mathcal{M}_{halt}^{c1}, \mathcal{M}_{halt}^{c2}$ will violate φ_f. □

A.5 Proof of Lemma 10

PROOF. Assume that the two counter machine halts. Then starting from the initial instruction, we reach the instruction ℓ_{halt}. From the above lemmas, and the construction of the MMS, we know that starting from the initial mode $\mathcal{I}$, we will reach a unique mode $\mathcal{M}_{i\ halt}$, spending one unit of time at all the intermediate modes. From Lemma 9, we also know that a time of one unit is spent at $\mathcal{M}_{i\ halt}$, and that the only safe modes to goto from here are $\mathcal{M}_{halt}$ or $\mathcal{M}_{halt}^{c1}, \mathcal{M}_{halt}^{c2}$. The use of modes $\mathcal{M}_{halt}^{c1}, \mathcal{M}_{halt}^{c2}$ is just to obtain $c_1 = c_2 = 0$. Notice that n_{halt} stays non-zero and grows in these modes. Once we achieve $c_1 = c_2 = 0$, then we can visit $\mathcal{M}_{halt}$ and obtain $n_{halt} = 1$. Notice that when this happens, we will have all variables other than n_{halt} as 0. By our safety conditions, no x, n or z variable can stay non-zero when $n_{halt} > 0$. Even if we obtain $n_{halt} = 0$ by staying at $\mathcal{M}_{halt}$, we cannot visit any other mode, since atleast one variable will become negative and violate safety. Our starting configuration with $s_0 = 1$ ensured that we could start from $\mathcal{I}$ and continue in a safe manner.

The converse, that is, any safe computation starting from $(1, 0, 0, \ldots, 0)$ and reaching $(0, 0, 0, \ldots, 1)$ is possible only when the two counter machine halts by the construction of the MMS. □

Bringing it all together for Undecidability.

Starting from the point $(1, 0, 0, \ldots, 0)$ which lies in the hyperplane H_0 given by $s_0 + n_{0j} = 1$, where ℓ_j is the unique instruction following ℓ_0, we stay in H_0 as long as control stays in the initial mode $\mathcal{I}$. Control then switches to mode $\mathcal{M}_0$, to the hyperplane H_1 given by $n_{0j} + x_{0j} = 1$. Note that $H_0 \cap H_1$ is non-empty and intersect at the point where $n_{0j} = 1$, and all other variables are 0. Spending a unit of time at $\mathcal{M}_0$, control switches to mode $\mathcal{M}_{0j}$, and to the hyperplane H_2 given by $x_{0j} + n_{jk} = 1$ depending on whether ℓ_j is not a zerocheck instruction or not. Again, note that $H_1 \cap H_2$ is non-empty and intersect at the point where $c_1 = 1, x_{0j} = 1$ and all other variables are zero. This continues, and we obtain a seamless transition from hyperplane H_i to H_{i+1} as dictated by the simulation of the two counter machine. The sequence of safe hyperplanes lead to the hyperplane H_{last} given by $n_{halt} = 1$ and all other variables 0 iff the two counter machine halts.