Task and Motion Planning Is PSPACE-Complete

William Vega-Brown, Nicholas Roy*
CSAIL, MIT
32 Vassar Street, 32-33x
Cambridge, Massachusetts 02139
{wrvb,nickroy}@csail.mit.edu

Abstract

We present a new representation for task and motion planning that uses constraints to capture both continuous and discrete phenomena in a unified framework. We show that we can decide if a feasible plan exists for a given problem instance using only polynomial space if the constraints are semialgebraic and all actions have *uniform stratified accessibility*, a technical condition closely related to both controllability and to the existence of a symbolic representation of a planning domain. We show that there cannot exist an algorithm that solves the more general problem of deciding if a plan exists for an instance with arbitrary semialgebraic constraints. Finally, we show that our formalism is universal, in the sense that every deterministic robotic planning problem can be well-approximated within our formalism. Together, these results imply task and motion planning is **PSPACE**-complete.

Introduction

Generalized deterministic robotic planning problems like mobile manipulation or legged locomotion have been studied for decades (Lozano-Pérez 1976; Wilfong 1988; Koditschek 1994). While numerous practical algorithms have been presented (recent examples include Garrett, Lozano-Pérez, and Kaelbling (2018) and Dantam et al. (2018), among many others), many fundamental questions of decidability and complexity remain open. In particular, it is not known under what conditions we can decide if a given deterministic planning problem has a solution, nor is there an effective decision procedure for the generalized problem of deterministic task and motion planning (TMP).

Part of the challenge in resolving these questions is that it is not obvious how to make these questions precise, as there is no consensus on what defines a TMP problem. In addition, many obvious approaches fail. For example, one naïve approach to TMP might be to combine a task planner with a motion planner and arrive at an effective decision procedure. Unfortunately, this requires choosing start states and goal states with which to query our motion planner, and the set of possible motion planning queries is not *a priori* bounded. This means that designing an effective decision procedure remains difficult, even if the task planner and motion planner are both complete and correct. Practical approaches work around this issue in a variety of ways, but constructing such a decomposition, or even just showing such a decomposition exists, is the core technical difficulty in analyzing TMP.

In this paper, we introduce a new intermediate representation, the *continuous constraint contract representation* (C^3). C^3 places both the discrete and continuous aspects of a problem in a single unified framework, and describes the capabilities of a system using an explicit constraint graph between the state of the world before and after an action. We show that every robotic planning problem satisfying mild technical conditions can be accurately represented by a C^3 instance, and we use tools from semialgebraic geometry and geometric control theory to show that if the constraints can be decomposed in a certain way—a technical property called *uniform stratified accessibility*—then we can decide if a semialgebraic C^3 instance has a solution using only polynomial space. Furthermore, we show that without this condition, semialgebraic C^3 is undecidable. Our algorithm bounds the complexity of TMP from above, and implies that task and motion planning is **PSPACE**-complete.

Complexity Results in Planning

To illustrate the complexity of TMP, consider the special case of *path planning*, where an articulated robot with arbitrarily many links must navigate among polygonal obstacles. Reif (1979) proved this problem was **PSPACE**-hard by explicitly constructing an oddly-shaped articulated body for which the feasible paths can simulate any given Turing machine with a polynomial space bound. Canny matched this lower bound by constructing a polynomial space algorithm for the path planning problem, thus proving path planning is **PSPACE**-complete (Canny 1988).

Canny's analysis does not extend to the case of planning under differential constraints, where not every collision-free path through configuration space is considered feasible. To the best of our knowledge there are no known upper bounds on the complexity of complete planners for differentially constrained systems, let alone general TMP problems. Still, Reif's construction provides a straightforward lower bound

*This research was supported by the US Army Research Laboratory's RCTA program. Their support is gratefully acknowledged.

on the complexity of TMP: since a general algorithm for task and motion planning must solve path planning as a special case, the **PSPACE**-completeness of path planning implies the **PSPACE**-hardness of TMP. Several recent results indicate that restricted categories of TMP problems are decidable. For example, Cheng et al. (2007) showed that differentially constrained planning is decidable if the set of permissible controls is finite and if the feasible paths can be expressed in closed form for any control. Deshpande et al. (2016) showed the decidability of plan existence for a class of controllable prehensile manipulation planning domains. Vendittelli et al. (2018) generalized this analysis to a class of driftless controllable systems. However, these results do not address task and motion planning in its full generality, do not extend to optimizing planners in a straightforward way, and do not address issues of computational complexity.

The Continuous Constraint Contract Representation

In this section, we describe the continuous constraint contract representation, our novel framework for studying TMP problems. At a high level, this representation can be understood as a generalization of planning languages like propositional STRIPS to include continuous variables, requirements, and effects. C^3 uses *constraints* to encode the structure of the planning problem. We do not discuss the details of how to model problems using C^3 in this document; instead, we focus on the theoretical ideas needed to analyze our representation.

States and Variables We assume the domain of discourse has a time-varying state $\mathbf{s}$ drawn from a compact set $\mathcal{S} \subseteq [0,1]^n$. The restriction to the unit hypercube is arbitrary—our analysis holds for any compact set. Note that while the state is a vector of real numbers, not every vector of real numbers is a valid state; we can easily include discrete information by limiting the domain of some elements of the state vector to a finite set. For example, we could model the state space of a propositional STRIPS instance by choosing $\mathcal{S}$ to be the set of binary vectors of length n: $\mathcal{S} = \{0,1\}^n \subset \mathbb{R}^n$.

In addition, we will often constrain some part of a state vector $\mathbf{s}$ in order to endow it with higher-level meaning. For example, in a three-dimensional planning problem, we often need to reason about the poses of arbitrary objects. A three-dimensional pose $\xi \in \mathrm{SE}(3)$ can be smoothly embedded into $\mathbb{R}^{12}$, subject to the constraint that nine of those numbers constitute a valid rotation matrix. A part of the state vector with semantic meaning is called a *variable*, which takes on a *value* that can be derived from the world state. That is, for each variable $v \in \mathcal{V}$, there is a domain $\mathcal{D}_v$ and a function $f_v : \mathbf{s} \to \mathcal{D}_v$ that extracts the relevant information from the state vector and maps it to the appropriate problem-specific domain. We define the valid state vectors using *constraints* $g_v : \mathbf{s} \to \mathbb{R}$, so that $g_v(\mathbf{s}) = 0$ if and only if the part of the world state in $\mathbf{s}$ corresponding to v is valid. This requirement is expressed symbolically as $g_v(f_v^{-1}(\mathcal{D}_v)) = \{0\}$. In this way, we can represent complicated spaces, including arbitrary manifolds and complex geometry, using real numbers.

Actions We define the capabilities of an agent using *parameterized actions*. Much as in propositional STRIPS, each action is defined in terms of requirements and effects. Unlike in propositional STRIPS, each action has a set of continuous parameters, and these requirements and effects are expressed in terms of these parameters.

Actions describe the capabilities of the modeled robotic system in terms of parameterized conditional contracts between a planner and a controller. Actions are contracts in the sense that there is a guarantee that if certain *requirements* hold, then there is an executable policy that will terminate in finite time, incur only bounded cost, and ensure that certain *effect* conditions hold. Actions are parameterized in the sense that a single action represents a continuum of possible control policies, and the specific effect conditions depend on the choice of parameters.

For example, an action `place_object_1` might represent placing a particular object at some specified coordinates. The choice of target pose would be represented by a *parameter* $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^m$ for some $m \in \mathbb{N}$. Much as we represented arbitrary variables using real numbers, here we represent arbitrary parameter spaces using constrained subsets of the reals.

We will use the notation $a \in \mathcal{A}$ to refer to an action, and $\mathbf{a} \in \bigsqcup_{a \in \mathbf{a}} \boldsymbol{\Theta}_a$ to refer to an action together with its parameters. We define the requirements and effects of an action a in terms of two functions $f_a : \mathcal{S} \times \boldsymbol{\Theta}_a \to \mathcal{S}$ and $g_a : \mathcal{S} \times \boldsymbol{\Theta}_a \to \mathbb{R}$. The function $f_a(\mathbf{s}, \boldsymbol{\theta})$ defines the *effects* of the action. If the system is in state $\mathbf{s}$ and executes action a with parameters $\boldsymbol{\theta}$, then at some point in the future the system will reach state $\mathbf{s}' = f_a(\mathbf{s}, \boldsymbol{\theta})$. The set of states reachable by an action is called the image of the action, denoted $\textsc{Post}(a) = \{\mathbf{s}' | \exists \mathbf{s}, \boldsymbol{\theta} : f(\mathbf{s}, \boldsymbol{\theta}) = \mathbf{s}'\}$. The function $g_a(\mathbf{s}, \boldsymbol{\theta})$ defines the *requirements*; an action is feasible from a state $\mathbf{s}$ if and only if there exist parameters $\boldsymbol{\theta}$ such that $g_a(\mathbf{s}, \boldsymbol{\theta}) = 0$. The set of states for which an action is feasible is called the preimage of the action, denoted $\textsc{Pre}(a) = \{\mathbf{s} | \exists \boldsymbol{\theta} : f_a(\mathbf{s}, \boldsymbol{\theta}) = 0\}$. Finally, we define the set of pairs of states that are connected by an action as the connected set $\textsc{Con}(a) = \{(\mathbf{s}, \mathbf{s}') | \exists \boldsymbol{\theta} : f_a(\mathbf{s}, \boldsymbol{\theta}) = \mathbf{s}' \wedge g_a(\mathbf{s}, \boldsymbol{\theta}) = 0\}$.

This formalization is extremely expressive. First, it is a strict generalization of languages like propositional STRIPS; positive and negative preconditions correspond to requiring a variable take on a value of 1 or 0, respectively, and we can define propositional STRIPS effects by constructing the function $f_a(\mathbf{s}, \boldsymbol{\theta})$ that leaves every variable unchanged but assigns the desired value (0 or 1) to the target variables. Second, it can define the capabilities of very complicated switching-mode control systems.

Planning Problems We define a C^3 *instance* $\mathcal{I}$ as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathbf{s}_0, \mathbf{S}_g, \mathcal{C})$, where $\mathcal{S}$ is the set of possible world configurations, $\mathcal{A}$ is a collection of actions, $\mathbf{s}_0 \in \mathbf{s}$ is an initial state, $\mathbf{S}_g \subset \mathcal{S}$ is a set of possible goal states, and $\mathcal{C} : \mathcal{S} \times \bigsqcup_{a \in \mathbf{a}} \to \mathbb{R}_{>0}$ is a *cost function* mapping each pair of state and parameterized action to the cost of executing the action from that state. A C^3 instance contains all the information needed to specify a well-defined planning problem.

A *plan* $\mathbf{p}$ is a sequence of actions $(\mathbf{a}_1, \ldots, \mathbf{a}_n)$ together

with their parameters. The *length* of a plan $|\mathbf{p}|$ is the number of actions it contains. The *trace* of a plan $\mathrm{Trace}\,(\mathbf{p})$ is the sequence of states the system reaches when executed from the initial state $\mathbf{s}$. A *feasible* plan is a plan $\mathbf{p} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ whose trace $\mathrm{Trace}\,(\mathbf{p}) = (\mathbf{s}_0, \ldots, \mathbf{s}_N)$ satisfies

- $\mathbf{s}_0$ is the initial state for the instance,
- $\forall i \in [1, N]$, $\mathbf{s}_i = f_{\mathbf{a}_{i-1}}(\mathbf{s}_{i-1})$,
- $\forall i \in [1, N]$, $g_{\mathbf{a}_i}(\mathbf{s}_i) = 0$, and
- $\mathbf{s}_N \in \mathbf{S}_g$.

An *optimal* plan is a feasible plan whose cost $\mathcal{C}(\mathbf{p}) = \sum_{i=1}^{n} \mathcal{C}(\mathbf{s}_{i-1}, \mathbf{a}_i)$ is no greater than the cost of any other feasible plan. An ϵ-near optimal solution is a feasible plan whose cost is no more than $1 + \epsilon$ times the cost of an optimal plan.

We can then formally define two important decision problems related to planning. First, the problem of *plan existence* is to decide, given a C^3 instance $\mathcal{I}$, whether a feasible plan exists. Second, the decision form of the *optimization problem* asks whether, given a C^3 instance $\mathcal{I}$, a plan of cost less than c exists. Note that if we can solve the optimization problem, then we can compute the cost of the optimal plan arbitrarily accurately by repeatedly solving the optimization problem for a sequence of cost thresholds chosen by (for example) binary search.

Semialgebraic C^3 Domains

In order to analyze the problems of plan existence and optimal planning, we must first decide how the constraints defining each action and axiom are specified. In line with prior research on the complexity of robot motion planning (Reif 1979; Schwartz and Sharir 1986; Canny 1988), we focus on systems defined by *semialgebraic* sets. A set S is *semialgebraic* if it can be described by a Tarski sentence, a quantified Boolean combination of polynomial equalities and inequalities. For example, the sentence $\exists x : ax^2+bx+c = 0 \wedge x > 0$ defines the set of possible coefficients of a quadratic equation that has a positive real solution. A sentence with no free variables is either true or false. $\forall x : x^2 \geq 0$ and $\forall y \exists x : x^2 < y$ are both Tarski sentences; the former happens to be true, while the latter is false.

It is a remarkable and important observation that any set defined by a Tarski sentence that includes the quantifiers $\forall$ or $\exists$ can also be defined *without* quantifiers: real algebraic geometry supports *quantifier elimination*. This result is the Tarski-Seidenberg theorem (Tarski 1948; Seidenberg 1954), which can be used to show that the closure, interior, and complement of a semialgebraic set are all semialgebraic sets, as is the distance between semialgebraic sets. In addition, there are effective algorithms for deciding a variety of questions about semialgebraic sets, such as whether a set is empty (that is, whether a system of polynomial equalities and inequalities has a solution) or whether two points lie on the same connected component.[1] These facts underly modern results on the decidability of motion planning.

[1]For a more thorough introduction to semialgebraic geometry, we recommend the short book by Coste (2000).

When we refer to semialgebraic C^3 domains, we mean any C^3 instance in which the constraints are all given by Tarski sentences defined by polynomials with rational coefficients. We can use semialgebraic constraints to describe both continuous and discrete phenomena. Most important manifolds for robotics, such as the special Euclidean groups $\mathrm{SO}(2)$ and $\mathrm{SE}(3)$, are semialgebraic sets, and object geometry (including meshes) can be modeled as semialgebraic sets. A binary variable can be represented by a real number subject to the constraint $x_i(x_i - 1) = 0$, ensuring the permissible values are 0 and 1. Using these constrained variables, we can represent complicated ideas. Because semialgebraic sets can be defined using the Boolean operators $\{\wedge, \vee, \neg\}$ as well as the quantifiers $\{\exists, \forall\}$, any sentence from propositional logic can be represented as a semialgebraic set. If x, y, z are binary variables, the propositional sentence $x \vee \neg y \implies z$ is equivalent to the polynomial equality $(x^2+(1-y)^2)^2(1-z)^2 + x^2(1-x)^2 + y^2(1-y)^2 + z^2(1-z)^2 = 0$. The statement "the robot does not collide with an obstacle when it follows a given spline" can be captured by the Tarski sentence $\forall t \in [0, 1], \forall x \in \mathbb{R}^3, (x \notin X_{\mathrm{robot}} \vee \xi(t)x \notin X_{\mathrm{obs}})$ where $\xi(t)$ is the pose of the robot at time t, and X_{robot} and X_{obs} are semialgebraic sets representing the robot and obstacle geometry.

Stratified C^3 Domains

Unfortunately, as we will show later in this paper, plan existence for semialgebraic C^3 is undecidable. In this section, we give sufficient conditions under which the problem of deciding if an instance has a solution can be solved using only polynomial space. In particular, we show that if the instance admits a *stratification* for which the actions are *uniformly accessible*—technical terms that we define in this section—then our algorithm uses only polynomial space.

We begin by defining stratification and uniform accessibility, then introduce the notion of uniform stratified accessibility, a particular kind of compatibility between a stratification and a C^3 instance. We show that if a C^3 instance has this property of uniform stratified accessibility, then we can bound the length of any feasible plan. Finally, we use this bound to construct an algorithm to decide if a stratifiable semialgebraic C^3 instance has a feasible plan, and show that our algorithm requires only polynomial space.

Stratification

A *stratification* is a way of partitioning a complex configuration space into pieces called *strata* that look like (i.e., are homeomorphic to) subsets of Euclidean space. In particular, in a stratification, each piece has constant dimension, and the boundary of each piece is covered by other pieces of strictly smaller dimension. Consider a square; the interior of the square looks like a two dimensional space, while each of its four edges look like a one dimensional space. The boundaries of the edges are the corners of the square, which look like zero-dimensional spaces.

Every space we consider in this paper, and essentially every space we care about for robotics, admits a stratification.

This is true even if the topology of the space is quite complicated, as in robotic arms or other linkages, or if the configuration space has obstacles of arbitrary smoothness. Stratifications are relevant to TMP because they provide some measure of regularity while still allowing us to describe actions that reach constrained subspaces, such as grasping actions in a manipulation problem. If the actions allow us to control the configuration of the robot on each stratum, then we can bound the number of actions required to reach any point on a stratum. If, in addition, the set $\textsc{Pre}(\mathbf{a})$ can be defined in terms of the strata of a suitable stratification, then we can bound the number of strata we must reach. Together, these two conditions will allow us to prove that if a plan exists at all for a given instance, there is a plan whose length is at most exponential in the length of the input. That some problems require exponentially long plans should not be surprising—after all, propositional STRIPS instances can require exponentially long plans, and C^3 is a superset of propositional STRIPS. It may, however, be surprising that plans need only ever be exponentially long, despite the fact that there are uncountably many states in the domain.

Formally, a *stratification* of the configuration space $\mathcal{S}$ of a C^3 instance $\mathcal{I}$ is a finite collection Σ of subsets of $\mathbf{s}$ called *strata* satisfying three technical conditions: each $\sigma \in \Sigma$ has constant dimension; the strata are pairwise disjoint and cover the space, so that every $\mathbf{s} \in \mathcal{S}$ is an element of exactly one stratum $\sigma \in \Sigma$; and the *frontier* $\mathrm{cl}(\sigma) \setminus \sigma$ of each stratum $\sigma \in \Sigma$ is the union of some members of Σ with dimensions strictly smaller than $\dim \sigma$. A *semialgebraic* stratification is a stratification in which each stratum is defined by a semialgebraic formula.

Uniform Accessibility

Recall that for each action, there is an associated function $f_a(\mathbf{s}, \boldsymbol{\theta})$ mapping the initial state $\mathbf{s}$ and the selected parameters $\boldsymbol{\theta}$ to the resulting state $\mathbf{s}'$. This is a *discrete-time dynamical system*.[2] Such systems can exhibit a variety of exotic behavior in general, but if they are *accessible* they are generally quite tame (Jakubczyk and Sontag 1990). Loosely speaking, a system is accessible if the parameters $\boldsymbol{\theta}$ actually behave like a control signal: if some parameters $\boldsymbol{\theta}$ take the system to a state $\mathbf{s}$, we can reach any nearby $\mathbf{s}$ with small changes to $\boldsymbol{\theta}$.

Formally, an action $\mathbf{a}$ is *accessible* if for any $\mathbf{s} \in \textsc{Pre}(\mathbf{a})$, the set $\{\mathbf{s}' | \exists \boldsymbol{\theta} : \mathbf{s}' = f_{\mathbf{a}}(\mathbf{s}, \boldsymbol{\theta})\}$ has a non-empty interior, and for any $\mathbf{s}' \in \textsc{Post}(\mathbf{a})$, the set $\{\mathbf{s} | \exists \boldsymbol{\theta} : \mathbf{s}' = f_{\mathbf{a}}(\mathbf{s}, \boldsymbol{\theta})\}$ has a non-empty interior. That is, for any point $\mathbf{s}$ there is a point z and a radius $\delta > 0$ such that every point in the ball $B(z, \delta)$ is reachable from $\mathbf{s}$ in one step, and conversely there is a z' such that $\mathbf{s}$ is reachable from every point in the ball $B(z, \delta)$ in one step.

While accessibility is enough to categorize the reachable space under arbitrarily long plans, it is insufficient to bound plan length. This is because the amount of free space that is reachable from any point can become arbitrarily small. Consider $f_{\mathbf{a}}(\mathbf{s}, \boldsymbol{\theta}) = \mathbf{s} \cdot (\boldsymbol{\theta} + \frac{1}{2})$, where $\mathbf{s} \in [0, 1]$ and $\boldsymbol{\theta} \in [0, 1]$. This system is accessible, but as it moves closer and closer to zero, the speed with which it moves decreases. This means it takes n steps to reach the state $\frac{1}{n}$, and that there is no bound on the number of steps needed to reach any reachable state. We introduce the notion of *uniform accessibility*[3] to avoid this issue.

We say a system is uniformly accessible if there exists a radius $\delta > 0$—independent of the state $\mathbf{s}$—such that for any point $\mathbf{s}$ every point in some ball $B(z', \delta)$ is reachable from $\mathbf{s}$ in one step, and $\mathbf{s}$ is reachable from every point in some ball $B(z, \delta)$ in one step. As we will show, if a system is uniformly accessible and its state space is bounded, then every reachable state is reachable by a plan of length $\mathcal{O}(2^{\dim \mathcal{S}})$.

Uniform Stratified Accessibility

For robotic planning, accessibility is often too stringent of a condition to require. In any problem involving contact-based manipulation, the accessible space will be restricted to the subspace where non-grasped objects do not move. To accommodate this, we introduce a notion of uniform *stratified* accessibility, asking that each state in a ball of radius δ be reachable from some state near the initial state. That is, we require that there is some δ such that if $(\mathbf{s}, \mathbf{s}') \in \textsc{Con}(\mathbf{a}) \subset \sigma \times \sigma'$, then for all z, there exists z' such that the interior of $\{\mathbf{s}'' \in \sigma' | \exists \mathbf{s} \in B(z, \delta) \cap \sigma : (\mathbf{s}, \mathbf{s}'') \in \textsc{Con}(\mathbf{a})\}$ contains a ball $B(z', \delta) \cap \sigma'$. Note that if the state space is discrete, as it would be in a pure task planning problem, each strata will contain only a single point. Consequently, discrete systems always have uniform stratified accessibility for small enough δ.

This condition is complex, but meshes nicely with our intuitive understanding of controllability. If we can break the configuration space into a finite number of pieces such that on each piece we can move around more-or-less freely, than the instance is stratifiable. This is closely related to the notion of *stratified controllability* developed by Goodwine and Burdick (2001). The constant δ is related to the size of the geometry; a problem with very fine geometry might require a very small δ, in order to ensure that an action can actually reach every point in a ball of radius δ. Observe that if the stratification is semialgebraic, the statement that an instance has uniform stratified accessibility can be expressed as a Tarski sentence—which means there is an effective algorithm using only polynomial space to both decide if an instance is stratifiable, and to choose the largest δ for which it is stratifiable.

Deciding Stratifiable Semialgebraic C^3

We are now ready to formulate the main result of this paper. We say that a C^3 instance $\mathcal{I}$ is *stratifiable* if there is a stratification Σ of its configuration space and a constant $\delta > 0$ such that each action $\mathbf{a} \in \mathcal{A}$ has the uniform stratified accessibility property with radius δ, and such that the preimage of each action, the goal set, and the singleton set containing

[2] Note that systems of this type are called discrete-time because they were first studied as an extension of continuous-time dynamical systems. We do not actually discretize time in this paper, as the actions in our plans are not indexed by time.

[3] The terminology is a direct analogy to standard notions like uniform convergence and uniform continuity.

just the initial state all must be elements of the stratification. Most standard instances of TMP are stratifiable, including all instances with semialgebraic geometry where each action either leaves an object fixed or allows free motion through free space. In this section, we show that if a semialgebraic C^3 instance is stratifiable, then we can decide if a plan exists using polynomial space. Note the algorithm we present does not need to know the stratification itself, just its size; the decision procedure we give accepts as input an instance $\mathcal{I}$, a natural number $|\Sigma|$, and a constant $\delta > 0$.

First, we show that if an instance is stratifiable and has a feasible plan, then there is an upper bound on the length of the shortest plan.

Theorem 1. *Let Σ be a stratification compatible with a C^3 instance $\mathcal{I}$ that has uniform stratified accessibility with constant δ. Then either the instance $\mathcal{I}$ has no solution, or it has a solution of length at most $N^* = \left\lceil |\Sigma||\mathcal{A}|\frac{1}{\zeta_d}\left(\frac{3}{\delta}\right)^d \right\rceil$, where d is the dimension of $\mathbf{s}$ and ζ_d is the volume of the unit ball in $\mathbb{R}^d$.*

Proof. Note first that for any $\delta > 0$ there is a set of $N_\delta = \left\lceil \frac{1}{\zeta_d}\left(\frac{3}{\delta}\right)^d \right\rceil$ balls that cover the unit cube. Thus if there there is a plan $p = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$ of length $N > N^*$ with trace $(\mathbf{s}_0, \cdots, \mathbf{s}_N)$, there must exist $i < j$ such that $\mathbf{a}_i = \mathbf{a}_j$, $\|\mathbf{s}_{i-1} - \mathbf{s}_{j-1}\| < 2\delta$, $\mathbf{s}_i \in \sigma', \mathbf{s}_j \in \sigma'$. Since $\mathbf{s}_i, \mathbf{s}_j \in B(z', \delta) \cap \sigma'$, it follows from the definition of stratified accessibility that there are parameters for $\mathbf{a}_{i+1}$ that reach some point in $B(\mathbf{s}_{j+1}, \delta) \cap \sigma_{j+1}$. This argument can be repeated until we reach the end of the plan. The result is a feasible plan $(\mathbf{a}_1, \ldots, \mathbf{a}_i, \mathbf{a}_{j+1}, \mathbf{a}_{j+2}, \ldots, \mathbf{a}_N)$ of length $N - j + i$, which is strictly less than N. Because we can do this whenever $N > N^*$, we can iteratively shorten any plan until we arrive at a plan of length N^*. Thus provided a plan exists, we can construct a plan of length at most N^*. □

Our decision algorithm closely resembles Savitch's algorithm (Savitch 1970), which can be used to decide propositional STRIPS plan existence. Recall that for discrete planning, we can check for the existence of a plan of length less than N linking two states by recursively checking if there exists a plan of length $N/2$ to some state, and a plan of length $N/2$ from that state to the goal. This process involves a maximum recursion depth of $\log_2(N)$ (using constant space at each level). Since the longest possible plan in a propositional STRIPS instance has length exponential in the number of predicates, this means we can decide plan existence using only polynomial space.

In the context of semialgebraic C^3, there are uncountably many possible intermediate states $\mathbf{s}'$, so we cannot simply enumerate them. However, given a semialgebraic set S_0, we can decompose the set of states reachable from some state in S into a finite number of cells, each of which is a semialgebraic set. Furthermore, we can enumerate these cells using only polynomial space (Basu, Pollack, and Roy 2016). This forms the core of our algorithm; starting with the bound N^* on the length of the shortest plan that we calculated in Theorem 1, we recursively enumerate the semialgebraic cells reachable using a plan of length at most $N^*/2$. This algorithm is defined in detail in Algorithm 1.

Algorithm 1 An algorithm to enumerate reachable subsets in semialgebraic C^3.

```
1:  function REACH(n ∈ ℕ, semialgebraic sets S_0, S_1)
2:      if n = 1 then
3:          for each action a ∈ 𝒜 do
4:              S_pre ← PRE(a)
5:              S_post ← APPLY(a, S_Pre)
6:              for set S in DECOMPOSE(S_post) do
7:                  yield INTERSECTION(S, S_1)
8:      else
9:          for each action a ∈ 𝒜 do
10:             S_pre ← PRE(a)
11:             for set S in REACH(⌊n/2⌋, S_0, S_pre) do
12:                 S_post ← APPLY(a, S)
13:                 for set S' in REACH(⌈n/2⌉, S_post, S_1) do
14:                     yield S'
```

Note that the **yield** keyword inherits its meaning from the Python programming language: it returns the indicated value, then suspends execution until the next time the caller requests a value. Algorithm 1 invokes several subroutines not defined in this paper. The subroutine $\textsc{Decompose}(x)$ generates a decomposition of the semialgebraic set x into basic semialgebraic cells. It can be implemented using the algorithms in chapter 14 of Basu et al. (2016). The subroutine $\textsc{Pre}(\mathbf{a})$ returns the semialgebraic set of all configurations from which action $\mathbf{a}$ is valid. The subroutine $\textsc{Apply}(\mathbf{a}, S)$ returns the semialgebraic set of states reachable from some state in S by action $\mathbf{a}$. Finally, the subroutine $\textsc{Intersection}(S_1, S_2)$ returns the intersection of two semialgebraic sets as a semialgebraic set.

We claim that Algorithm 1 will generate all semialgebraic sets reachable in a plan of length n. The first branch of the function (lines 3-7) covers the base case, generating a decomposition of the space reachable from S_0 in a single step. The second branch (lines 9-14) is the recursive case; it generates a decomposition of the subset of S_1 reachable from S_0 in at most $n + 1$ steps. The function recurses to enumerate the reachable subsets of the set from which each action can be executed, then recurses again to enumerate the subsets of S_1 that can be reached after executing that action.

Algorithm 1 has a recursion depth of $\mathcal{O}(\log n)$, and uses polynomial space at any point in the computation. That is, if the C^3 instance can be encoded in a string of length m on some Turing machine, Algorithm 1 requires space polynomial in m and time exponential in m.

The correctness of Algorithm 1 and the upper bound given in Theorem 1 suggest a straightforward decision procedure: invoke Algorithm 1 to check for plans of length $\left\lceil |\Sigma||\mathcal{A}|\frac{1}{\zeta_d}\left(\frac{3}{\delta}\right)^d \right\rceil$ linking $\mathbf{s}_0$ to some state in S_g. This procedure is given in Algorithm 2.

Algorithm 2 returns True if and only if there is a state in S_g reachable from $\mathbf{s}_0$ via a plan of of length less than n. Since a plan exists if and only if a plan of length less

Algorithm 2 An algorithm for determining plan existence.

function PLANEXISTS(semialgebraic sets $\mathbf{s}_0, S_g, \delta \in \mathbb{R}_{>0}$)
  $n \leftarrow \left\lceil |\Sigma||\mathcal{A}| \frac{1}{\zeta_d} \left(\frac{3}{\delta}\right)^d \right\rceil$
  if REACH$(n, \{\mathbf{s}_0\}, S_g)$ yields anything **then**
    return True
  else
    return False

than n exists, Algorithm 2 is correct. Since our bound on plan length is exponential in the size of the input and Algorithm 2 requires space logarithmic in the length bound and polynomial in the size of the input, it follows that Algorithm 2 uses only polynomial space. Specifically, if the description involves polynomials of degree at most r in a space of dimension d, Algorithm 2 requires $\mathcal{O}(\left(\log_2 |\Sigma| + \log_2 |\mathcal{A}| + d \log_2 \frac{1}{\delta}\right) (d \log_2(r))^{\mathcal{O}(1)})$ space. This allows us to prove the **PSPACE**-completeness of stratified semialgebraic C^3.

Theorem 2. *Stratified semialgebraic C^3 is **PSPACE**-complete.*

Proof. Algorithm 2 decides plan existence using only polynomial space; this establishes that plan existence is in **PSPACE**. As argued in a previous section, any finite propositional STRIPS instance with binary variables can be expressed as an instance of stratified semialgebraic C^3 by replacing binary variables with real variables subject to the constraint $x_i(1 - x_i) = 0$. Since this construction is only polynomially larger than the original propositional propositional STRIPS instance and since planning using propositional STRIPS is **PSPACE**-complete (Bylander 1994), plan existence is **PSPACE**-hard. Because plan existence for stratified semialgebraic C^3 is **PSPACE**-hard and in **PSPACE**, it is **PSPACE**-complete. □

Finally, observe that it is straightforward to extend Algorithm 2 into an algorithm for the optimization problem. Given a C^3 instance $\mathcal{I}$ and any cost threshold $\bar{c}$, we can construct an augmented C^3 instance $\mathcal{I}_{\bar{c}}$ from $\mathcal{I}$ by adding a variable c that does not appear in any preconditions and whose value is the total cost incurred by all prior actions. If the goal of $\mathcal{I}_{\bar{c}}$ includes the semialgebraic constraint $c < \bar{c}$, we can use Algorithm 2 to decide if a plan exists that incurs cost at most $\bar{c}$. This construction requires only one additional variable and thus only polynomially more space than deciding if a plan exists at all; accordingly, near-optimal planning for stratified semialgebraic C^3 is also **PSPACE**-complete.

Undecidability of Non-stratifiable C^3

We have shown that plan existence is decidable for stratified semialgebraic C^3 instances. In this section we study the converse, and show that there is no algorithm to decide the more general problem of plan existence for semialgebraic C^3 instances that may or may not be stratifiable. In fact, we can show that for any Turing machine T and string ω, there is a semialgebraic C^3 instance $\mathcal{I}$ that has a solution if and only if T accepts ω. To prove this, we first establish a connection between semialgebraic C^3 and Diophantine equations, i.e., equations of the form $p(\mathbf{x}) = 0$ with $\mathbf{x} \in \mathbb{R}^k$ and p a polynomial with integer coefficients.

Theorem 3. *Let $p(\mathbf{x}) = 0$ be a Diophantine equation in k variables of degree d, i.e., a polynomial in $x_1, \ldots, x_k$ with integer coefficients. Then there is a semialgebraic C^3 instance $\mathcal{I}_p$ with k variables and degree at most kd that has a solution if and only if there exist natural numbers $x_1, \ldots, x_k$ such that $p(x_1, \ldots, x_n) = 0$.*

Corollary 1. *There is no algorithm to determine whether or not a semialgebraic C^3 instance has a solution.*

Proof. The proof is constructive. We will create a C^3 instance for which the reachable states are precisely the inverse natural numbers, i.e., of the form $(\frac{1}{n_1}, \ldots, \frac{1}{n_k})$. We then construct a polynomial of degree at most kd that has a solution in the inverse natural numbers if and only if $p(x) = 0$ has a solution in the natural numbers.

The instance has k variables $\mathcal{V} = \{v_1, \ldots, v_k\}$, each with a domain $v_i \in (0, 1]$, and k actions $\mathcal{A} = \{a_1, \ldots, a_k\}$. Each action has no parameters, and has postcondition $v_k' = \frac{v_k}{v_k+1}$ with all other variables unchanged. This is semialgebraic—it is the unique solution to the quadratic polynomial $v_k v_k' + v_k' - v_k = 0$, which has integer coefficients. Observe that if we apply $\mathbf{a}_k$ from a state where $v_k = \frac{1}{n}$ for some natural number n, then $v_k' = \frac{1}{n+1}$. Consequently, if our instance has initial state $\mathbf{s} = (1, \ldots, 1)$, then a state is reachable if and only if it is of the form $(\frac{1}{n_1}, \ldots, \frac{1}{n_k})$ for some natural numbers $n_1, \ldots, n_k \in \mathbb{N}^k$.

Next, construct a polynomial $q(\mathbf{s})$ that has a solution if and only if $p(\mathbf{x})$ has a solution. Let d_i be the maximal exponent of x_i in p, and let $q(v_1, \ldots, v_k) = p(1/v_1, \ldots, 1/v_k) \prod_i v_i^{d_i}$. Then since $v_i > 0 \, \forall i$, $\prod_i v_i^{d_i} > 0$. Consequently, we have $0 = p(\frac{1}{v_1}, \ldots, \frac{1}{v_k}) \iff q(v_1, \ldots, v_k) = 0$. Furthermore, if $q(\mathbf{s}) = 0$ for $\frac{1}{v_i} \in \mathbb{N} \forall i$, then $p(x_1, \ldots, x_k) = 0$ with $x_i \in \mathbb{N} \forall i$.

Let $\mathbf{S}_g = \{\mathbf{s} : q(\mathbf{s}) = 0\}$, and $s_0 = (1, \ldots, 1)$. Then the C^3 instance $\mathcal{I}_p = (\mathcal{V}, \mathcal{A}, , \mathbf{s}_0, \mathbf{S}_g)$ has a solution if and only if $\mathbf{S}_g$ contains a point in the inverse natural numbers, which is true if and only if $p(\mathbf{x})$ has a solution in the natural numbers. □

Theorem 3 implies that an algorithm to decide if a C^3 instance is solvable would imply an algorithm to decide if a Diophantine equation is solvable. Hilbert's tenth problem (Hilbert 1902) asked for such an algorithm, and Matiyasevič, Robinson, Davis, and Putnam (Matiyasevič 1970; Davis 1973) proved no such algorithm can exist. Thus, there is no algorithm to decide if an arbitrary semialgebraic C^3 instance has a solution. □

Stratified Semialgebraic C^3 is universal

While we have shown that stratified semialgebraic C^3 is **PSPACE**-complete, it is not obvious that our results extend

to TMP more generally. While semialgebraic sets can capture arbitrary shapes, in general the solution to arbitrary differential equations are not semialgebraic. However, if a system is defined by differential constraints that are piecewise-analytic, then the reachable sets can be *approximated* by semialgebraic sets arbitrarily well. This implies that if we relax the objective of planning to include some small tolerance for inaccuracy—which we always do, when we implement algorithms using floating point arithmetic—then semialgebraic C^3 is in a sense as good as any other representation. That is, semialgebraic C^3 is *universal*: it can model any robotic system arbitrarily well. In this section, we formalize and prove this claim.

We first observe that a dynamical system with piecewise-analytic differential constraints can be described by a stratified C^3 instance with analytic constraints.

Theorem 4. *Let the dynamics $\dot{x} = f(x, u)$ be piecewise-analytic in the sense of Sussmann (1979). If the system is stratified controllable in the sense of Goodwine and Burdick (2001), then there is a stratified C^3 instance whose actions represent piecewise-analytic vector fields, in which the constraints can be expressed as equalities and inequalities involving only analytic functions.*

A proof of this theorem is beyond the scope of this paper; however, in our previous work (2016), we proved essentially the same result as part of our proof of asymptotic optimality. Essentially, the proof involves constructing a foliation on each stratum of a motion planning problem; the leaves of these foliations are the states reachable without leaving a particular stratum.[4]

Unfortunately, there is no hope of an algorithm to decide plan existence for semianalytic C^3. This is a consequence of Richardson's theorem (1969), which shows that there is no algorithm to decide if an arbitrary semianalytic set is empty, even for highly restricted subsets of analytic functions. For example, Laczkovich (2003) showed that if $\mathcal{F}$ is the set of functions formed by substituting x, $\sin x^n$, and $\sin(x \sin x^n)$ into an arbitrary polynomial with integer coefficients, then there is no algorithm to decide if there exists x such that $f(x) > 0$, nor is there an algorithm to decide if there exists x such that $f(x) = 0$. Consequently, there exist instances of C^3 with semianalytic constraints for which we cannot even decide if a plan of length one exists.

However, analytic functions on a C^3 domain can be well-approximated by algebraic functions. In particular, Baouendi and Goulaouic (1971; 1974) showed that if f is an analytic function defined on a compact set $K \subset \mathbb{R}^n$, then there exist constants $\alpha, \beta > 0$ such that for any $n \in \mathbb{N}$, there is a polynomial p_n of degree n with

$$\sup_{x \in K} |f(x) - p_n(x)| \leq 2^{\alpha - \beta n}.$$

While constructing such a polynomial may be difficult, this is sufficient to show that we can solve a relaxed version of the plan existence problem for compact semianalytic domains.

[4] In the notation of control theory, the strata are called *modes* and the leaves of the foliations are called *orbits*.

Theorem 5. *For any semianalytic instance $\mathcal{I}$ and positive ϵ, there is a corresponding semialgebraic instance $\hat{\mathcal{I}}$ with maximal degree $\mathcal{O}(\log \epsilon)$ such that any plan $\hat{\mathbf{p}}$ feasible for $\hat{\mathcal{I}}$ is nearly feasible for $\mathcal{I}$. Let $d(\mathbf{p}, \mathbf{p}') = \sup_s \inf_t \|\mathrm{Trace}(\mathbf{p})(s) - \mathrm{Trace}(\hat{\mathbf{p}})(t)\|$ be the distance between two plans. If $P_{\mathrm{feas}}[\mathcal{I}]$ is the set of feasible plans for $\mathcal{I}$, then*

$$\hat{\mathbf{p}} \in P_{\mathrm{feas}}[\hat{\mathcal{I}}] \iff \exists \mathbf{p} \in P_{\mathrm{feas}}[\mathcal{I}] : d(\mathbf{p}, \hat{\mathbf{p}}) < \epsilon.$$

Together, the results of the previous sections imply that we can decide plan existence in any planning domain with piecewise-analytic differential constraints to any degree of precision using only polynomial space. While the set of all TMP problems is not well-defined, no formulation of which we are aware is not a special case of the more general set of problems with piecewise-analytic constraints. In this sense then, we can claim that TMP, or more generally, all deterministic robotic planning, are **PSPACE**-complete.

Comparison to Standard Formalisms

Our framework is non-standard, and one might object that it is too inclusive: many problems we can represent using stratified semialgebraic C^3 do not resemble practical robotic planning problems. However, as a consequence of the universality of semialgebraic C^3, our complexity results apply not just to problems given in our representation, but to TMP more generally. In particular, any representation for robotic systems that includes task planning or motion planning as a special case is in some sense equivalent to semialgebraic C^3, and is therefore **PSPACE**-complete. This is highly relevant to recent efforts in the robotic planning community to establish a common representation for TMP problems.

For example, Lagriffoul et al. (2018) proposed a standardized representation for benchmarking TMP. The language they describe is contained within stratified semialgebraic C^3, and thus all of our decidability and complexity results extend readily to those benchmark problems. Other frameworks, like the extended action specification of Garrett et al. (2016) or the logic-geometric programming described by Toussaint and Lopes (2015; 2017), can also be interpreted as special cases of the C^3 representation.

Implications of PSPACE-completeness

We have shown that task and motion planning is **PSPACE**-complete. The algorithm we provide is not practical; it discards most of the structure in the planning problem, and makes no use of heuristics or other standard techniques. Still, its existence has several important ramifications. First, the universality of semialgebraic C^3 implies *sufficient* conditions for a specification language. Any language that can specify arbitrary semialgebraic C^3 is "as good as" any other language. These conditions are highly relevant to the ongoing effort to standardize a specification language for TMP. Second, task and motion planning is asymptotically no harder than task planning or motion planning alone. This is the best result we could have hoped for; provided we restrict our attention to controllable systems, there is no subtle

interaction that makes hybrid planning even harder than the already-hard constituent planning problems. Finally, Corollary 1 highlights the importance of discrete structure; without a stratification, the problem quickly stops even resembling motion planning. This points to the fundamental importance of symbolic reasoning as a tool for understanding continuous decision-making.

References

Basu, S.; Pollack, R.; and Roy, M.-F. 2016. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Canny, J. 1988. *The complexity of robot motion planning*. MIT Press.

Cheng, P.; Pappas, G.; and Kumar, V. 2007. Decidability of motion planning with differential constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Coste, M. 2000. *An introduction to semialgebraic geometry*. Istituti Editoriali e Poligrafici Internazionali.

Dantam, N.; Kingston, Z.; Chaudhuri, S.; and Kavraki, L. 2018. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research* 37(10):1134–1151.

Davis, M. 1973. Hilbert's tenth problem is unsolvable. *The American Mathematical Monthly* 80(3):233–269.

Deshpande, A.; Kaelbling, L. P.; and Lozano-Pérez, T. 2016. Decidability of semi-holonomic prehensile task and motion planning. In *Workshop on the Algorithmic Foundations of Robotics*.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2016. FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2018. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research* 37(13-14):1796–1825.

Goodwine, B., and Burdick, J. 2001. Controllability of kinematic control systems on stratified configuration spaces. *Transactions on Automatic Control* 46(3):358–368.

Hilbert, D. 1902. Mathematical problems. *Bulletin of the American Mathematical Society* 8(10):437–479.

Jakubczyk, B., and Sontag, E. 1990. Controllability of nonlinear discrete-time systems: A Lie-algebraic approach. *SIAM Journal on Control and Optimization* 28(1):1–33.

Koditschek, D. 1994. An approach to autonomous robot assembly. *Robotica* 12(02):137–155.

Laczkovich, M. 2003. The removal of π from some undecidable problems involving elementary functions. *Proceedings of the American Mathematical Society* 131(7):2235–2240.

Lagriffoul, F.; Dantam, N.; Garrett, C.; Akbari, A.; Srivastava, S.; and Kavraki, L. 2018. Platform-independent benchmarks for task and motion planning. *Robotics and Automation Letters* 3(4):3765–3772.

Lozano-Pérez, T. 1976. The design of a mechanical assembly system. Master's thesis, Massachusetts Institute of Technology.

Matiyasevič, Y. V. 1970. The Diophantineness of enumerable sets. *Doklady Akademii Nauk* 191(2):279–282.

Reif, J. 1979. Complexity of the mover's problem and generalizations. In *IEEE Symposium on Foundations of Computer Science*, 421–427.

Richardson, D. 1969. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic* 33(4):514–520.

Savitch, W. 1970. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences* 4(2):177–192.

Schwartz, J., and Sharir, M. 1986. Motion planning and related geometric algorithms in robotics. In *Proceedings of the International Congress of Mathematicians*, 1594–1611.

Seidenberg, A. 1954. A new decision method for elementary algebra. *Annals of Mathematics* 365–374.

Sussmann, H. 1979. Subanalytic sets and feedback control. *Journal of Differential Equations* 31(1):31–52.

Tarski, A. 1948. A decision method for elementary algebra and geometry. Technical Report R-109, RAND Corp.

Toussaint, M., and Lopes, M. 2017. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Toussaint, M. 2015. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Vega-Brown, W., and Roy, N. 2016. Asymptotically optimal planning under piecewise-analytic constraints. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Vendittelli, M.; Laumond, J.-P.; and Mishra, B. 2018. Decidability in robot manipulation planning. *arXiv preprint arXiv:1811.03581*.

Wilfong, G. 1988. Motion planning in the presence of movable obstacles. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, 279–288. ACM.