Compositional Relational Abstraction for Nonlinear Hybrid Systems

XIN CHEN, University of Colorado Boulder
SERGIO MOVER, University of Colorado Boulder
SRIRAM SANKARANARAYANAN, University of Colorado Boulder

We propose techniques to construct abstractions for nonlinear dynamics in terms of relations expressed in linear arithmetic. Such relations are useful for translating the closed loop verification problem of control software with continuous-time, nonlinear plant models into discrete and linear models that can be handled by efficient software verification approaches for discrete-time systems. We construct relations using Taylor model based flowpipe construction and the systematic composition of relational abstractions for smaller components. We focus on developing efficient schemes for the special case of composing abstractions for linear and nonlinear components. We implement our ideas using a relational abstraction system, using the resulting abstraction inside the verification tool NuXMV, which implements numerous SAT/SMT solver-based verification techniques for discrete systems. Finally, we evaluate the application of relational abstractions for verifying properties of time triggered controllers, comparing with the Flow* tool. We conclude that relational abstractions are a promising approach towards nonlinear hybrid system verification, capable of proving properties that are beyond the reach of tools such as Flow*. At the same time, we highlight the need for improvements to existing linear arithmetic SAT/SMT solvers to better support reasoning with large relational abstractions.

Additional Key Words and Phrases: Hybrid systems, relational abstraction, nonlinear systems, SMT, bounded model checking

ACM Reference format:
Xin Chen, Sergio Mover, and Sriram Sankaranarayanan. 2017. Compositional Relational Abstraction for Nonlinear Hybrid Systems. *ACM Trans. Embedd. Comput. Syst.* 0, 0, Article 0 (2017), 19 pages.
DOI: 0000001.0000001

1 INTRODUCTION

A relational abstraction of a plant model uses a relation $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$ to capture the states reached at time t, starting from initial state $\vec{x}(0)$ with control input $\vec{u}$. Relational abstractions have already been studied for linear systems [35, 42, 47]. They have allowed the efficient use of SAT/SMT solver-based verification techniques such as bounded model checking (BMC) and IC3 [5, 8] in the context of linear hybrid systems. However, computing relational abstractions for nonlinear systems remains an open challenge, thus far.

The approach presented in this paper builds on the use of Taylor model-based integration of nonlinear systems [6, 7, 12, 36]. A Taylor model approximates the trajectory of a nonlinear system

This article was presented in the International Conference on Embedded Software 2017 and appears as part of the ESWEEK-TECS special issue.
 1539-9087/2017/0-ART0 $15.00
DOI: 0000001.0000001

using a polynomial plus an interval to represent the uncertainty. As such, Taylor models form a good basis for constructing relations between current and next states of a system. However, to control the size of the uncertainty, we require the overall state and input space to be subdivided into numerous cells, computing a Taylor model for each cell. This allows us to obtain a relation between the current state and a reachable state at a future time instant. However, this also involves a tradeoff between the precision of the abstraction vs the cost of computing and using the abstraction.

In this paper, we explore ideas to enable the efficient computation of relational abstractions for nonlinear systems and their use in verifying properties of closed loop control systems. The main contributions of this paper are as follows:

(a) We present an adaptive subdivision scheme to control the uncertainty intervals in our result. This allows the user to specify an "error tolerance" up front. Smaller tolerances will result in more subdivisions, naturally. At the same time, an adaptive subdivision can find larger cells where the behavior of the system is easier to approximate by a lower degree polynomial with small error, while using smaller cell sizes for regions where the system's behavior is more complex.
(b) Next, we present a compositional approach that builds a relation for a large plant model by analyzing how the plant model is put together from subsystems. Often, systems are provided to us as monolithic ODEs without specifying the components. To facilitate this, a dependency graph is computed from the plant ODE to automatically decompose a given ODE as a composition of "modules". The compositional approach is particularly effective when many of the components are linear.
(c) Finally, our approach is evaluated over a set of benchmark plant and controller models, and compared against the Flow* tool for verifying nonlinear hybrid systems through flowpipe computation [13].

The evaluation shows that the optimizations proposed in this paper can drastically reduce the computation time and the size of the resulting abstraction. Further, these relations when used inside the NuXMV tool yields fast proofs of some key properties using the IC3 algorithm and counterexamples using the BMC algorithm [10]. However, we also observe limitations arising primarily due to the tradeoff between the precision of the relational abstraction and its size. An imprecise abstraction is easier to analyze but yields more false positives, when compared to a more precise analysis.

1.1 Motivating Example

Figure 1 shows an example of a controller for the speed and heading of a vehicle along a fixed road. The state of the vehicle is specified by 4 variables: (x, y) : position in meters, v : velocity in m/s, θ : heading angle in radians. It has two control inputs θ_{ref} the reference heading in radians, and u, the throttle input in m/s^2. The continuous time dynamics are given by:

$$\begin{array}{llll} \dot{x} & = v\cos(\theta) & \dot{y} & = v\sin(\theta) \\ \dot{\theta} & = -3(\theta - \theta_{\text{ref}}) + d_\theta(t) & \dot{v} & = -d_v(t)v^2 + u + d_u(t) \end{array} \tag{1}$$

The time varying disturbances that affect the vehicle include $d_v(t) \in [0.009, 0.01]\text{m}^{-1}$, $d_u(t) \in [-0.45, 0.45]\text{m/s}^2$, $d_\theta(t) \in [-0.1, 0.1]\text{rad/s}$.

Our goal is to control the vehicle along the desired trajectory (see Figure 1) at a desired speed $v^* = 20$ m/s. We consider a speed control and an independent heading control. Both controllers run at a period $\delta = 0.1$ seconds (10 Hz frequency). Note that the plant state variables (x, y, v, θ) sensed by the controllers are subject to sensor noise and estimation errors η. We also assume that the computation time for control inputs is negligible compared to δ.

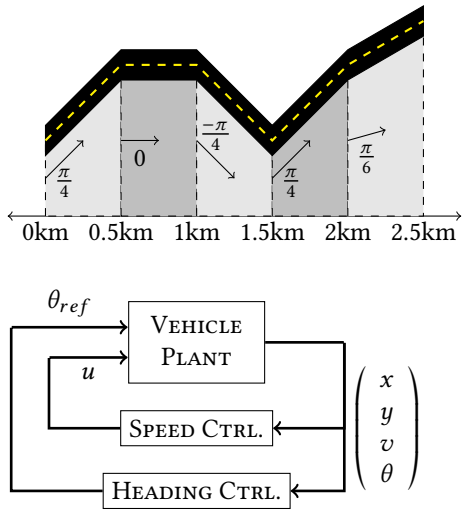

Fig. 1. Motivating Example: Automatic Navigation System for following a planned path.

Speed Control: The speed control implements a proportional control with saturation, and is computed as $u = \min(-U, \max(U, -K(v - v^*)))$, wherein $U = 5$, $K = 0.1$ and $v^* = 20$.

Heading Control: The heading control operates in 5 modes depending on the value of x, wherein mode i operates in the range $500(i-1) \leq x < 500i$ meters and attempts to set the target heading to θ_i^*, the desired heading for mode i. However, to avoid a dangerous skid, the reference heading is changed by at most $\pm\delta_{min}$ at each step. Here $\delta_{min} = 0.1$ radians.

$$\theta_{ref} = \begin{cases} \theta + \delta_{min} & \theta < \theta_i - \epsilon \\ \theta - \delta_{min} & \theta > \theta_i + \epsilon \\ \theta & \theta \in \theta_i \pm \epsilon \end{cases}$$

The value of ϵ is set to 0.1. The heading control transitions from mode i to $i+1$ whenever $x(j\delta) \geq 500i$ at the j^{th} sampling step.

Relational Modeling: A relational model of the plant is a relation of the form $R(\vec{x}(\delta), \vec{x}(0), \vec{u})$, that models all the states $\vec{x}(\delta)$ reachable over a single time period starting from an initial plant state $\vec{x}(0)$ and control input $\vec{u}$ held constant over the time period $[0, \delta]$. For instance, whenever $v(0) \in [0, 10]$, the reachable states at time $\delta = 0.1$ satisfy the following relation:

$$\begin{aligned} x(0.1) &\in x(0) - 4.983 \times 10^{-3} u + 9.95 \times 10^{-2} v(0) + [-11.116, 8.972] \\ y(0.1) &\in y(0) + 0.0716\theta_{ref} + 0.452\theta(0) + [-12.207, 12.207] \\ v(0.1) &\in 0.0709 + 0.0995u + 0.99v + [-0.0445, 0.0191] \\ \theta(0.1) &\in 0.259\theta(0) + 0.0741\theta + [-0.0866, 0.0866] \end{aligned} \tag{2}$$

Note that the relation is expressed as a formula in linear real arithmetic although the original ODE (1) is nonlinear. Likewise, by partitioning the full range $v \in [0, 50]$ and $\theta \in [-\pi, \pi]$, into smaller sub ranges, we obtain our relational model as a disjunction of relations of the form shown in (2).

A relational model abstracts the plant as a discrete-time transition system. Furthermore, as long as the relations involved are expressed in linear arithmetic, tools that use efficient linear

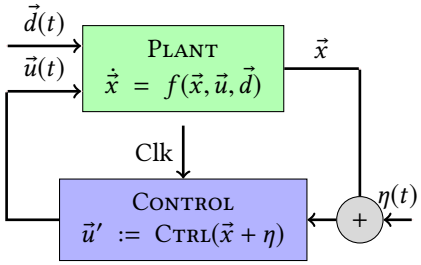

Fig. 2. Schematic block diagram of the closed loop plant and control model with disturbances.

arithmetic SMT solvers can be directly applied to tackle verification problems for nonlinear hybrid systems [16, 20, 23]. However, this process involves a tradeoff between obtaining a more precise relation that may involve a subdivision of the state space versus a more coarser relation that can lead to false counterexamples. In Section 5, we demonstrate how properties can be verified for these controllers using standard constraint-based verification techniques such as bounded model checking and IC3 [5, 8, 17].

2 PRELIMINARIES

We present preliminary details that will define the model of computation used for closed loop systems, formally define relational abstractions, summarize the various classes of abstractions for linear systems and finally, introduce flowpipe construction technique for nonlinear systems using Taylor models.

We call the set of reals between two rationals $a \leq b$, an *interval*, and denote it by $[a, b]$. A vector of intervals has the form $[\vec{a}, \vec{b}]$, wherein $\vec{a} \leq \vec{b}$. The width of an interval $\text{width}([\vec{a}, \vec{b}])$ is defined as $\max_i(\vec{b}_i - \vec{a}_i)$.

2.1 Time Triggered Models

We will work with time triggered models, as shown in Figure 2, that compose a continuous time plant $\mathcal{P}$ with a discrete controller that samples the plant state periodically, updating a control input $\vec{u}$, that is held constant over a time period.

The plant $\mathcal{P}$ is defined by a system of Ordinary Differential Equations (ODEs) over state variables $\vec{x}$, inputs $\vec{u}$ and (possibly time-varying) disturbances $\vec{d}(t)$: $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d}), \vec{x} \in \mathcal{X},\ \vec{u} \in \mathcal{U},\ \vec{d} \in \mathcal{D}$, wherein $\mathcal{X}$ denotes the state-space, $\mathcal{U}$ denotes the control input space and $\mathcal{D}$ denotes the disturbance input space. We will assume that f is C^∞ (i.e, continuous and differentiable to any order) for simplicity. Throughout this paper, we will assume that $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{D}$ are *compact*, i.e., closed and bounded. This is often reasonable since physical quantities involved in many plant models naturally range over a possibly large but compact set.

Given an initial condition $\vec{x}(0)$, input signal $\vec{u}(t)$ and a disturbance signal $\vec{d}(t)$, the solution φ_f exists over some time horizon $\Delta > 0$, such that for any $t \in [0, \Delta]$, $\vec{x}(t) :\ \varphi_f(t, \vec{x}(0), \vec{u}(\cdot), \vec{d}(\cdot))$.

The controller operates periodically at times $t = 0, \delta, 2\delta, \ldots$, computes a function Ctrl, sensing the plant state with an associated sensing/estimation noise $\eta(t)$. At each time step, it computes a function $\vec{u} = \text{Ctrl}(\vec{x} + \eta)$, outputting the resulting control to the plant, where it is held constant for a time period.

2.2 Relational Abstraction

In general, a relational abstraction $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$ of a plant model $\mathcal{P}$ is a relation T involving the initial state $\vec{x}(0)$, (constant) control input $\vec{u}$ and the state reachable $\vec{x}(t)$ at some time $t \geq 0$.

Definition 2.1 (Relational Abstraction). A relation $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$ is an abstraction of a plant model $\mathcal{P}$ defined by an ODE $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d})$ over the time interval $t \in [0, \delta]$ iff for all (a) initial conditions $\vec{x}(0) \in \mathcal{X}$, (b) control inputs $\vec{u} \in \mathcal{U}$ (assumed to be held constant over $[0, t]$), and (c) (time-varying) disturbance signals $\vec{d}(t) \in \mathcal{D}$, the state $\vec{x}(t)$ reachable at any time $0 \leq t \leq \delta$ satisfies the relation $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$.

$$\vec{x}(t) = \varphi_f(t, \vec{x}(0), \vec{u}, \vec{d}(\cdot)) \;\rightarrow\; T(\vec{x}(t), \vec{x}(0), \vec{u}, t)\,.$$

The relations contain everything that can be reached by the underlying dynamics. On the other hand, the relations will often be over approximations of the underlying system. As a result, although $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$ may hold, $\vec{x}(t)$ is not reachable from $\vec{x}(0)$ through the choice of inputs $\vec{u}$ and time t and any valid choice for the disturbance signal $\vec{d}(\cdot)$. Since the overall abstraction is composed with a controller that is time triggered, the control input $\vec{u}$ is assumed to be held constant over the time period $[0, \delta]$ of the relational abstraction. Finally, the relational abstraction holds over all possible choices of the disturbance signals $\vec{d}(t) \in \mathcal{D}$. However, the disturbance input itself is not part of the relation.

In this paper, we distinguish between two types of relations that will be computed: (a) *time-aware relations* (as defined in Def. 2.1) explicitly involve time t as a variable [35], and (b) *fixed-time relations* that fix time t to a specific value $t = \delta$. Naturally, a fixed-time relation can be inferred given a time-aware relation by simply substituting the fixed value δ for time. For a time triggered system, as defined in Section 2.1, we will seek a fixed-time relation, fixing $t = \delta$, the time period of the controller. However, as we will subsequently show, the derivation of *time-aware* relations will be a critical step towards computing fixed-time relations.

Form of the Relation: Modern SMT-based verification tools can efficiently handle systems whose transitions can be expressed in linear arithmetic [20, 23]. Recall that a linear arithmetic formula over a set of variables X is defined as a Boolean combination of linear inequalities over X.

In this paper, we will first examine relations polynomial in the time variable and linear in the state/control variables.

Definition 2.2 (Polynomial Time Linear State (PTLS) Predicates). A predicate $\mathcal{T}(\vec{x}(t), \vec{x}(0), \vec{u}, t)$ is said to be polynomial in time and linear in the state and controls (PTLS) iff it is of the form:

$$\vec{x}(t) \;\in\; P(t)\vec{x}(0) + Q(t)\vec{u} + p(t) + I,$$

wherein $P(t), Q(t)$ are matrices whose entries are polynomials over time t, $p(t)$ is a polynomial and I is a vector of intervals.

The (time-aware) relational abstractions will be of the form:

$$\bigwedge_{i=1}^{N} \underbrace{\varphi_i(\vec{x}(0), \vec{u})}_{\text{Linear Arithmetic}} \;\rightarrow\; \underbrace{\mathcal{T}_i(\vec{x}(t), \vec{x}(0), \vec{u}, t)}_{\text{PTLS}}\,. \tag{3}$$

Fixing $t = \delta$ in (3) yields a linear arithmetic relation.

Lemma 2.3. *Given a relation T of the form shown in (3), fixing time $t = \delta$ for a constant δ yields a relation $R : T(\vec{x}(\delta), \vec{x}(0), \vec{u}, \delta)$ expressible in linear arithmetic.*

2.3 Linear Systems and Relational Abstractions

We first consider linear plant models of the form

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} + W\vec{d}\,, \tag{4}$$

wherein A, B and W are constant matrices. The closed-form solution of the linear ODE is given by

$$\vec{x}(t) = e^{At}\vec{x}(0) + \int_0^t e^{A(t-s)}(B\vec{u}(s) + W\vec{d}(s))ds\,.$$

Zutshi et al. [46] compute fixed time relations of the form $R(\vec{x}(\delta), \vec{x}(0), \vec{u})$ expressed as linear arithmetic formula of the form: $\vec{x}(t) \in P\vec{x}(0) + Q\vec{u} + I$, wherein $P := e^{A\delta}$ and $Q := \int_0^\delta e^{A(\delta-s)}Bds$. The technique does not handle disturbance inputs, as such. Furthermore, the techniques does not handle time-aware relations.

Mover et al. [35] improved upon previous work by computing time aware abstractions $T(\vec{x}(t), \vec{x}(0), \vec{u}, t)$. However, their approach computes relations that are linear in time t, as well. As a result, these abstractions lead to reduced overall precision when we compute fixed time relations for $t = \delta$.

Therefore, as introduced by Chen [11], for given fixed control inputs $\vec{u}$ and possibly time-varying disturbance $\vec{d}(t)$, the matrix exponential e^{At} and the integral $\int_0^t e^{A(t-s)}Bds$ can be approximated by a polynomial matrices $\Phi(t)$ and $\Psi(t)$ respectively. Furthermore, as the degree of the matrices in t is increased, so does the resulting accuracy. The disturbance term $\int_0^t e^{A(t-s)}W\vec{d}(s)ds$ is handled using approaches specific to time-varying disturbances $\vec{d} \in \mathcal{D}$ as a polynomial $q(t)$. The error bounds are computed as an interval I, so that the resulting relation T is obtained as a PTLS relation:

$$T(\vec{x}(t), \vec{x}(0), \vec{u}, t) : \ \vec{x}' \in \Phi(t)\vec{x}(0) + \Psi(t)\vec{u} + q(t) + I\,.$$

Example 2.4. We consider the dynamics of θ in the motivating example: $\dot{\theta} = -3(\theta - \theta_{\text{ref}}) + d_\theta(t)$, such that $\theta_{\text{ref}} \in [-\pi/3, \pi/3]$, and $d_\theta(t) \in [-0.1, 0.1]$. For $\theta(0) \in [-\pi/3, \pi/3]$, the relation for $t \in [0, 0.02]$ can be computed as below.

$$\begin{aligned}\theta(t) \ \in\ & \theta(0) + 3t\theta_{\text{ref}} - 2.999t\theta(0) - 4.499t^2 * \theta_{\text{ref}} + 4.5t^2\theta(0) \\ & + 4.5t^3\theta_{\text{ref}} - 4.499 * t^3 * \theta(0) + [-0.024, 0.024]\end{aligned}$$

2.4 Taylor Model Integration

We briefly review the method of Taylor model integration that forms a fundamental primitive for flowpipe construction techniques for nonlinear systems [6, 7, 12, 13, 36].

Definition 2.5 (Taylor model). A *Taylor Model (TM)* is defined by a pair $(p(\vec{x}), I)$ such that $p(\vec{x})$ is a polynomial over $\vec{x} \in D$ for an interval D, and I is an interval.

Given a smooth function $f(\vec{x})$ over D, we may compute a TM $(p(x), I)$ so that $\forall x \in D : \ (f(x) \in p(x) + I)$. TMs can be used to overapproximate the solutions of nonlinear ODEs. Given a nonlinear ODE $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d})$ along with an initial condition $\vec{x}(0) \in X_0$, the flowmap φ_f can be overapproximated by a TM (p_f, I_f):

$$\varphi_f(t, \vec{x}(0), \vec{u}, \vec{d}) \in p_f(t, \vec{x}(0), \vec{u}) + I_f \tag{5}$$

for all $\vec{x}(0) \in X$, $t \in [0, \delta]$ and disturbance $\vec{d}$. The method to compute such a TM is called *Taylor model integration* [7].

Extensions to handle time varying disturbances inside a bounded range were introduced by Chen [11].

Example 2.6. We consider the dynamics of the velocity v in the motivating example. It is defined by the ODE $\dot{v} = -d_v(t)v^2 + u + d_u(t)$ wherein $d_v \in [0.009, 0.01]$ and $d_u \in [-0.45, 0.45]$ are time-varying disturbances, $u \in [-6, 6]$ is a time-invariant input. By the TM integration method, the flowmap in the time interval $[0, 0.02]$ w.r.t. the initial condition $v(0) \in [10, 15]$ can be overapproximated by the order 4 TM (p_f, I_f) wherein

$$\begin{aligned} p_f &= v + 0.45t + tu + 0.176t^2 - 0.0095tv_0^2 - 0.042t^2v_0 + 0.063t^3 \\ &\quad - 0.0095t^2v_0u + 0.0033t^2v_0^2 + 0.019t^3u - 0.0067t^3v_0 + 0.0025t^4 \\ I_f &= [-0.0025, 0.00238] \end{aligned}$$

such that v_0 is the symbolic representation of $v(0)$.

3 ADAPTIVE RELATIONAL ABSTRACTION

The purpose of relational abstraction is to compute a relation T for an ODE $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d})$ such that

$$(\vec{x}' = \varphi_f(t, \vec{x}, \vec{u}, \vec{d})) \rightarrow T(\vec{x}', \vec{x}, \vec{u}, t) \tag{6}$$

holds for all $t \geq 0$ and $\vec{x}, \vec{x}'$ in the state space. To obtain a fixed-time relation R, we may simply replace $t = \delta$ in the relation T. The problem of relationizing linear ODEs has been well studied [35, 47]. In this paper, we study a general approach for nonlinear ODEs.

For $\vec{x} \in X$, $\vec{u} \in U$ and $t \in \Delta_t$ for an interval Δ_t, we may compute a Taylor model (p_f, I_f) for φ_f wrt the initial condition $\vec{x}(0) \in X$ in the time interval Δ_t.

Definition 3.1 (Linear Truncation). The linear trunction of a TM $(p(\vec{x}_0, t), I)$ of a TM over $\vec{x}_0 \in X$ and $t \in \Delta_t$ is a TM $(p_L(\vec{x}_0, t), I_L)$ that is linear in $\vec{x}_0$ and potentially polynomial over t. The resulting interval I_L is a conservative approximation:

$$I_L \supseteq I + \mathsf{range}_{\vec{x}_0 \in X, t \in \Delta_t}(p - p_L)\,.$$

To compute a relation in the PTLS form (see Definition 2.2), a simple strategy can be applied:

(1) Partition the possibly large set $X \times U$ into smaller sets

$$(X_1 \times U_1) \cup \cdots \cup (X_K \times U_K).$$

(2) Compute a Taylor model $(p_j(t, \vec{x}_0, \vec{u}), I_j)$ for each $(\vec{x}(0), \vec{u}) \in X_j \times U_j$.
(3) Perform a linear truncation to obtain (p_{Lj}, I_{Lj}).
(4) The overall relation $T(\vec{x}', \vec{x}(0), \vec{u}, t)$ is obtained as

$$\bigwedge_1^K (\vec{x}(0) \in X_j \ \wedge \ \vec{u} \in U_j) \ \rightarrow \ (\vec{x}' \in p_{jL}(\vec{x}(0), \vec{u}, t) + I_{jL})$$

Unfortunately, it has been pointed out that computing TMs is challenging when $X \times U$ is large (see [11, 32]). At the same time, a uniform subdivision of $X \times U$ is expensive. Therefore, as a first step, we will consider an adaptive algorithm that will subdivide the region $X \times U$ so that the width of each interval $\mathsf{width}(I_{Lj}) \leq w_{\max}$, for a user defined threshold. The technique is shown in Algorithm 1.

In each iteration, the algorithm tries to compute a relation for the first region $X \times U$ in the queue. If the TM integration method along with the linear truncation produces a TM whose width is small enough, then we add the current TM to the abstraction. Otherwise, the region Y is uniformly subdivided along each dimension, and the results are added to the queue.

Assume that $T(\vec{x}', \vec{x}, \vec{u}, t)$ is the result generated by Algorithm 1 for some $t \in [0, \delta]$.

Theorem 3.2. *T is a valid relational abstraction, satisfying the condition in (6).*

Algorithm 1 Relational abstraction for a nonlinear ODE

Input: An ODE $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d})$, a region X, a time interval Δ_t, a width cutoff $w_{\max}$.
Output: A relational abstraction Abs over X, U.

```
Abs ← ⊤;                                              # formula for abstraction
Queue ← ∅;                                            # queue for unprocessed regions
Enqueue the entire domain (X, U) to Queue;
while Queue is not empty do
    Dequeue (X_j, U_j) from Queue;
    Compute TM (p_f, I_f) with t ∈ Δ_t for x(0)∈X_j, u ∈ U_j;
    Linear truncate (p_f, I_f) to (p_L, I_L);
    if width(I_L) ≤ w_max then
        Abs ← Abs ∧ ((x ∈ X_j ∧ u ∈ U_j) → (x' ∈ p_L(t, x, u) + I_L));
    else
        Uniformly subdivide X_j × U_j into Y_1, ..., Y_r;
        Enqueue each Y_1, ..., Y_r into Queue;
    end if
end while
return Abs;
```

Although the adaptive subdivision algorithm improves upon the basic uniform subdivision scheme, it continues to be expensive since it involves subdividing over the entire state space. We alleviate this further in the next section by considering a compositional approach for subsets of the state variables.

4 COMPOSITIONAL RELATIONAL ABSTRACTION

In this section, we consider *compositional* relational abstractions that are obtained as a conjunction of relations over subsets of variables. In a compositional relation, the state variables $\vec{x}$ and control inputs $\vec{u}$ are partitioned into subsets $\vec{x}_1, \vec{u}_1, \vec{x}_2, \ldots, \vec{x}_k, \vec{u}_k$ such that all state variables appear in at least one subset $\vec{x}_j$, and control variables in some $\vec{u}_j$. A relation $T(\vec{x}', \vec{x}, \vec{u}, t)$ is of the form

$$T_1(\vec{x}_1', \vec{x}_1, \vec{u}_1, t) \ \wedge \ T_2(\vec{x}_2', \vec{x}_2, \vec{u}_2, t) \ \wedge \ \cdots \ \wedge \ T_k(\vec{x}_k', \vec{x}_k, \vec{u}_k, t)\,,$$

wherein each relation T_i is said to be a *factor*. Compositional relations are naturally obtained by considering the plant model as a composition of multiple smaller components over the subset of the entire state space. Even if the plant model is obtained as a "monolithic" ODE, it is possible to define components by considering a *dependency graph* between the state and control variables. The idea of studying state variable dependencies has been applied to flowpipe construction [15], stability analysis [43] and computing differential invariants [38]. Here, we use it to compositionally compute relational abstractions.

Definition 4.1. Given an ODE $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{d})$, a dependency graph has vertices corresponding to each state variable x_i and control variable u_j. An edge is added from x_j (or u_j) to x_i if the RHS for $\frac{dx_i}{dt}$ involves x_j (or u_j). Self-edges are elided from this graph.

A decomposition of the strongly connected components over the states of the dependency graph allows us to *decompose* a plant that is not given as a composition of components.

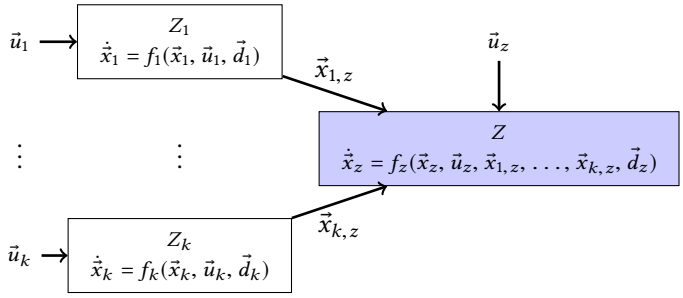

Fig. 3. Schematic diagram for basic composition step.

Example 4.2. Consider the vehicle plant model in Section 1.1. The dependency graph is shown below to the left. Each node in this graph is its own SCC. As a result, the components each contain a single variable, and are shown to the right.

The composed relation has the following structure:

$$T_1(v', v(0), u, t) \ \wedge \ T_2(\theta', \theta(0), \theta_{ref}, t) \wedge$$
$$T_3(x', x(0), \theta(0), v(0), u, \theta_{ref}, t) \wedge$$
$$T_4(y', y(0), \theta(0), v(0), u, \theta_{ref}, t)$$

The advantages of a compositional approach are three-fold: (a) it avoids an unnecessary decomposition of the entire state/control space while creating the relations. Therefore, we expect a computationally faster approach with a more succinct resulting relational abstraction. (b) In many plant models, some of the components will be linear, and thus can be handled more efficiently with a simpler and more scalable approach sketched in Section 2.3; and (c) the interconnections between components is often *sparse*, i.e, even if a component involves numerous state variables, often a very small subset of these are treated as "outputs" seen by the other components. We will exploit each of these common features.

4.1 Composing Relations

In this section, we consider a single composition step involving the composition of components $Z_1, \ldots, Z_k$ that have been previously relationalized to obtain a relation for a component Z, as shown in Figure 3. Let $\vec{x}_1, \ldots, \vec{x}_k$ be the state variables and $\vec{u}_1, \ldots, \vec{u}_k$ be the control inputs for components $Z_1, \ldots, Z_k$. Furthermore, let $\vec{x}_z$, $\vec{u}_z$ be the state variables for the component Z which depends on $Z_1, \ldots, Z_k$. We will assume that the variables $\vec{x}_{j,z}$ are the *interface variables* from $\vec{x}_j$ used in component Z.

Considering each of the components $Z_1, \ldots, Z_k$ in turn, we construct relations $T_1(\vec{x}_1', \vec{x}_1, \vec{u}_1, t)$, $\ldots$, $T_k(\vec{x}_k', \vec{x}_k, \vec{u}_k, t)$ in the PTLS form (see Definition 2.2). Specifically, let T_j be of the form

$$\bigwedge_i \ \varphi_{i,j}(\vec{x}_j(0), \vec{u}_j) \ \rightarrow \ \vec{x}_j(t) \in M_{i,j}(t)\vec{x}_j(0) + N_{i,j}\vec{u}_j + p_{i,j}(t) + I_{i,j} \ .$$

The predicates $\varphi_{i,j}$ represent the subdivisions over the space $\vec{x}_j, \vec{u}_j$. For each combination of the predicates $\varphi_{i_1,1}, \ldots, \varphi_{i_k,k}$ of the components $Z_1, \ldots, Z_k$, we collect the resulting TMs on the

interface variables $\vec{x}_{1,z}, \ldots, \vec{x}_{k,z}$ and substitute these into the ODE for the component Z. The result is a new ODE of the form

$$\dot{\vec{x}}_z = f_z(\vec{x}_z, \vec{u}_z, \vec{x}_1(0), \ldots, \vec{x}_k(0), \vec{u}_1, \ldots, \vec{u}_k, t)$$

Since $\vec{x}_1(0), \ldots, \vec{x}_k(0), \vec{u}_1, \ldots, \vec{u}_k$ are constants over the time interval $t \in [0, \delta]$, we proceed to perform a TM integration of this system following the adaptive scheme defined in Algorithm 1.

As such, if the relations $T_1, \ldots, T_k$ have $n_1, \ldots, n_k$ decompositions each, the target relation for Z will require $n_1 \times \cdots \times n_k$ calls to Algorithm 1. Each call further subdivides $\vec{x}_z$. Furthermore, if Z itself is composed with another block $\hat{Z}$, up stream, the number of subdivisions can be forbiddingly large.

However, despite this explosion, we have already saved on unnecessary splitting of the state variables for the components $Z_1, \ldots, Z_k$ in this process. We now consider two optimizations for mitigating the need to consider a combinatorial blowup for the component Z: (a) selective adaptive decomposition of the interface variables; and (b) composition into a linear component.

4.2 Interface Variable Adaptive Decomposition

The first optimization to improve the basic composition focuses on the interface variables $\vec{x}_{1,z}, \ldots, \vec{x}_{k,z}$ between components in Figure 3. Thus far, we independently abstract each of the components $Z_1, \ldots, Z_k$ to obtain relations $T_1, \ldots, T_k$ that serve two purposes: (a) they abstract the corresponding components, and (b) the relations for the interface variables are then used to obtain the composed relation for Z.

However, note that Algorithm 1 is adaptive with respect to a maximum width $w_{\max}$ over all the variables of the component under abstraction. Nevertheless, for the purposes of composition, the variables that matter are the interface variables $\vec{x}_{1,z}, \ldots, \vec{x}_{k,z}$.

To mitigate this, we create new relations for the purposes of composition: $\tilde{T}_1, \ldots, \tilde{T}_k$ by modifying Algorithm 1 in addition to the original relations $T_1, \ldots, T_k$.

Each relation $\tilde{T}_i$ is created by a modification of the adaptive subdivision in Algorithm 1 such that the maximum width $w_{\max}$ is checked just for the intervals over the interface variables $\vec{x}_{i,z}$. In this modified setup, the relations $\tilde{T}_1, \ldots, \tilde{T}_k$ are created for composition, wherein for each $i \in [1, k]$, $\tilde{T}_i(\vec{x}_{i,z}(t), \vec{x}_i(0), \vec{u}_i, t)$ is a relation that focuses on a subset of variables rather than the entire state space.

4.3 Composition Into Linear Components

Another important case is when Z is linear over the variables $\vec{x}_z$. In our experience, linear components are very common even if the system as a whole is nonlinear. In particular, given relations $\tilde{T}_1, \ldots, \tilde{T}_k$ for $Z_1, \ldots, Z_k$, we can relationalize Z *without any further calls* to Algorithm 1, or in other words, without necessarily requiring us to split the state space of $\vec{x}_z$.

Example 4.3. Continuing with the components in Example 4.2, we note that the component over θ is linear. Surprisingly, due to the decomposition, components for x, y are in fact linear over x, y although they are nonlinear over v, θ.

$$\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta)$$

In this simple case note that once we obtain relations for v, θ, the relation for x is readily computed as

$$x(t) = x(0) + \int_0^t v(s)\cos(\theta(s))ds\,.$$

Similar consideration applies to $y(t)$. The rest of this section formalizes this insight for the general case of linear components Z.

Let us write the dynamics for $\vec{x}_z$ as

$$\frac{d}{dt}\vec{x}_z = A\vec{x}_z + g(\vec{u}_z, \vec{x}_{1,k}, \ldots, \vec{x}_{k,z})\,.$$

The overall relation can be written as

$$\vec{x}_z(t) = e^{tA}\vec{x}_z(0) + \int_0^t e^{(t-s)A} g(\vec{u}_z(s), \vec{x}_{1,k}(s), \ldots, \vec{x}_{k,z}(s))ds\,. \tag{7}$$

We proceed by first approximating e^{tA} by a TM $\Phi(t)$ and $e^{(t-s)A}$ by $\Psi(t)$, wherein Φ, Ψ represent polynomial matrices and the corresponding error intervals over $t \in [0, \delta]$.

$$\vec{x}_z(t) \in \Phi(t)\vec{x}_z(0) + \int_0^t \Psi(s) g(\vec{u}_z(s), \vec{x}_{1,k}(s), \ldots, \vec{x}_{k,z}(s))ds\,. \tag{8}$$

Next, we consider the expressions for $\vec{x}_{1,k}(s), \ldots, \vec{x}_{k,z}(s)$

$$\vec{x}_{j,k}(s) := M_{j,k}(s)\vec{x}_j(0) + N_{j,k}\vec{u}_j + p_{j,k} + I_{j,k} \tag{9}$$

Plugging the expressions from (9) into (8), we obtain

$$\vec{x}_z(t) \in \Phi(t)\vec{x}_z(0) + \int_0^t \Psi(s)\, \hat{g}\left(\begin{array}{c} \vec{u}_z(s), \vec{x}_{1,k}(0), \ldots, \vec{x}_{k,z}(0), s, \\ \vec{u}_1, \ldots, \vec{u}_k \end{array} \right) ds\,.$$

We now compute a TM for $\vec{x}_z(t)$ by performing the integration above, which may in turn require approximating $\hat{g}$ by a polynomial using Taylor expansion.

Thus, if relations for $Z_1, \ldots, Z_k$ are already available and the component Z is linear, its relation is constructed without needing any further calls to Algorithm 1, i.e, without decomposing $\vec{x}_z$.

5 EXPERIMENTS

Implementation. We implemented a prototype tool for computing relational abstractions for hybrid systems. Given a continuous dynamics defined by a composed nonlinear ODE, the tool computes TM flowpipes of a given order using Flow* [13], and then merges and translates the flowpipes to obtain a linear relational abstraction. Although the abstraction defines a discrete system, it is still an *overapproximation* of the hybrid system on the behaviors at each abstraction time step. In order to input the abstraction to a model checker, we express it in the input language of *nuXmv* [10]. We try two model checking algorithms implemented in *nuXmv*: Bounded Model Checking (BMC) and a version of IC3 for infinite-state systems [17] to prove a property on an abstraction.

Performance evaluation. We evaluate the performance based on the time cost for proving the safety property of a hybrid system. More precisely, given a hybrid system along with a safe set, we derive a relational abstraction for it based on a time step size. Then we try to prove that the reachable set at each discrete time step is contained in the safe set. Hence, the total cost includes the time consumed to compute an abstraction and the time consumed to prove the safety or find a counterexample.

Comparison. We mainly compare our relational abstraction method to the flowpipe construction technique implemented in Flow*. Since we only reason the safety at discrete time steps for the hybrid system, to be fair, the tests on Flow* are implemented based on its C++ API instead of the user interface which proves the safety for the whole time interval. Since the abstraction is an overapproximation of the original hybrid system, if the abstraction is safe then so is the original

Table 1. Properties for the *Car* benchmark

#	Property	#	Property
1	$\Box v \geq 9$	13	$\Box m = 5 \wedge time \geq 5 \rightarrow \theta \leq 0.6$
2	$\Box v \leq 25$	14	$\Box(x \in [0, 500] \rightarrow y \leq x + 10)$
3	$\Box m = 1 \rightarrow \theta \leq 0.9$	15	$\Box(x \in [100, 500] \rightarrow 4y \geq 3.5x - 350)$
4	$\Box m = 1 \wedge time \geq 10 \rightarrow \theta \geq 0.6$	16	$\Box(x \in [0, 100] \rightarrow y \geq -10)$
5	$\Box m = 2 \rightarrow \theta \leq 0.9$	17	$\Box(x \in [500, 1000] \rightarrow y \leq 450)$
6	$\Box m = 2 \wedge time \geq 10 \rightarrow \theta \leq 0.1$	18	$\Box(x \in [500, 1000] \rightarrow y \geq 380)$
7	$\Box m = 2 \rightarrow \theta \geq -0.1$	19	$\Box(x \in [1000, 1500] \rightarrow y + 0.7x \geq 1100)$
8	$\Box m = 3 \rightarrow \theta \geq -0.9$	20	$\Box(x \in [1000, 1500] \rightarrow y + 0.7x \leq 1200)$
9	$\Box m = 3 \wedge time \geq 10 \rightarrow \theta \leq -0.6$	21	$\Box(x \in [1500, 2000] \rightarrow y \geq -50)$
10	$\Box m = 4 \rightarrow \theta \leq 0.9$	22	$\Box(x \in [1500, 2000] \rightarrow y \leq 100)$
11	$\Box m = 4 \wedge time \geq 20 \rightarrow \theta \geq 0.6$	23	$\Box(x \in [2000, 2500] \rightarrow y \geq 0.8x - 1600)$
12	$\Box m = 5 \rightarrow \theta \geq 0.4$	24	$\Box(x \in [2000, 2500] \rightarrow y \leq 0.8x - 1500)$

model. Hence, the comparison is based on proving the safety of the same system. If the safety can not be proved, we compare the number of steps where a counterexample is found, the larger the number, the better the result. Besides, we also provide comparisons between compositionally and monolithically computing relational abstractions, in the first way, we only perform subdivision in the dimensions of interface variables.

5.1 Benchmarks and Results

We present our benchmarks as well as the experiments as follows.

Single car. The dynamics of a single car is given in our motivating example. We try to prove the 24 safety properties shown in Table 1 in which m is the mode of the heading controller, $time$ counts the number of control steps, i.e., time steps, which is 0.01. In Flow*, we use the step size 0.01, the TM order 4, the cutoff threshold 10^{-10} and the precision 100 for floating-point numbers. The experimental results are given in Table 3. We also notice that by increasing the TM order and reducing the step size can not apparently improve the results due to the large uncertainties. For relational abstraction, we try to compute results according to different accuracies. Table 2 shows the experimental results. The system has 3 linear variables and 3 nonlinear variables. If we consider the linear variables form a component and the nonlinear variables form another one, then the interface variables are v and θ. It can be seen that the compositional method requires much less time on the same accuracy. Table 3 presents the time costs of proving the 24 properties on the compositional abstraction from Test #2 in Table 2. The results are better than the direct method in Flow* in most of the tests.

Car platoon of 3 cars. We consider the model of a platoon of 3 single cars each of which has the dynamics in the previous benchmark. The model is inspired by a continuous time system proposed by Chen et al. [15]. The 1st and the 3rd cars are driven manually, whereas the controller for the 2nd car chooses a velocity control that maintains its position in the middle of the two cars. Each car implements the same heading control law as used in the motivating example. But rather than consider changes in the road angle, we simply consider a single segment straight road with $\theta_1 = 0$.

The velocity control inputs u_1, u_3 for the 1st and the 3rd car respectively are treated as uncertainties in relational abstraction. For each car, the input is ranging in $[-2, -0.5]$ when the car velocity is larger than 22, $[-2, 2]$ when the velocity is in $[18, 22]$, and $[0.5, 2]$ when the velocity is below 18.

Table 2. Experiments of computing relational abstractions. Legend: δ: time step for fixed-step abstraction; $w_{\max}$: accuracy tolerance; Var_L: # of linear variables; Var_N: # of nonlinear variables; Var_I: selected interface variables; Sub: # of subdivisions.

				# of variables		Compositional			Monolithic	
No.	Benchmark	δ	$w_{\max}$	Var_L	Var_N	Var_I	T (sec)	Sub	T (sec)	Sub
1	Single car	0.05	0.5	3	3	v, θ	0.05	1	0.07	1
2	Single car	0.1	0.5				0.8	4	13	64
3	Single car	0.1	1.5				0.1	1	0.2	1
4	Single car	0.1	0.2				3.5	22	>1200	$\geq 8^6$
5	Artificial pancreas	0.5	1	3	9	G_p, G_t	4.9	4	>1200	$\geq 2^{12}$
6	Artificial pancreas	0.5	2				1	1	1	1
7	Artificial pancreas	5	10				563	4	>1200	$\geq 2^{12}$
8	Artificial pancreas	0.5	0.05				7.6	28	>1200	$\geq 2^{12}$

The controller for the middle car chooses the velocity as

$$\begin{aligned} u_2 = \quad & \max(-5, \min(5, \hat{u}_2)) \quad \text{such that} \\ \hat{u}_2 \in \quad & -0.3 * (v_2 - 20) + 0.3 * \begin{pmatrix} (x_3 - x_2 - 10 + [-0.1, 0.1]) \\ -(x_2 - x_1 - 10 + [-0.1, 0.1]) \end{pmatrix} \end{aligned}$$

The safety properties are defined as follows. *Property 1:* $\square(x_2 - x_1 > 2)$, i.e., the 1st and the 2nd will never be too close. *Property 2:* $\square(x_3 - x_2 > 2)$, i.e., the 2nd and the 3rd will never be too close.

Since each car can be abstracted independently, we take the compositional abstraction result of Test #2 in Table 2. The time costs for proving the two properties are presented in Table 3. For Flow*, we use the same computational setting for the single car model, and the check the safety of the reachable set at discrete steps. The tool can not compute a valid overapproximation that reaches $x \geq 500$ due to the large time-varying uncertainties. We tried much higher TM order and much smaller step size, but no apparent improvement can be made. The relational abstraction method is slightly better than Flow* but quite sensitive to the solver in use. Again, both of Flow* and the abstraction method prove the safety on the same system, since the abstraction is an overapproximation of the original system at discrete steps.

Car platoon of 7 cars. We extend the previous model to 7 cars. The 1st, 3rd, 5th and 7th cars are driven manually, and the rest of the cars are controlled similarly to the 2nd car in the previous model. If no successive two cars are within the distance less than 2 than the system is safe. The experimental results are presented in Table 3 with the abstractions from Table 2. Notice that when the abstraction # 4 is taken, the whole abstraction has 154 subdivisions.

Artificial pancreas. We consider the artificial pancreas control systems that automatically adjust the delivery of the hormone insulin to patient with type-1 diabetes to control their blood glucose levels in a closed loop. The plant model for the artificial pancreas benchmarks capture the human

insulin-glucose regulation system using a nonlinear ODE with 10 state variables shown below

$$
\begin{aligned}
\dot{X} &= -0.0278X + 0.0278(18.2129I_p - 100.25)\\
\dot{I}_{\text{sc1}} &= 0.0142I_{\text{sc1}} - 0.0078I_{\text{sc2}} + \boldsymbol{u_I}(t)\\
\dot{I}_{\text{sc2}} &= 0.0152I_{\text{sc1}} - 0.0078I_{\text{sc2}}\\
\dot{G}_t &= -0.0039(3.2267 + 0.0313X)G_t(1 - 0.0026G_t +\\
&\quad 2.5097\text{e-}6G_t^2) + 0.0581G_p - 0.0871G_t\\
\dot{G}_p &= 3.7314 - 0.0047G_p - 0.0121I_d\\
&\quad -0.0581G_p + 0.0871G_t + \boldsymbol{u_m}(t)\\
\dot{I}_l &= -0.4219I_l + 0.225I_p\\
\dot{I}_p &= -0.315I_p + 0.1545I_l + 1.9\text{e-}3I_{\text{sc1}} + 7.8\text{e-}3I_{\text{sc2}}\\
\dot{I}_1 &= -0.0046(I_1 - 18.2129I_p)\\
\dot{I}_d &= -0.0046(I_d - I_1)\\
\dot{G}_s &= 0.1(0.5521G_p - G_s)
\end{aligned}
$$

The model has two inputs $\boldsymbol{u_I}(t)$ a controlled input from the insulin glucose controller and $\boldsymbol{u_m}(t)$, an uncontrolled meal input that introduces glucose due to the patient's meal. A detailed explanation and rationale for the model and the various model parameters are available elsewhere [18, 33]. The system as a whole is partitioned into two components using a dependency graph analysis:

$$Z_1 : \{X, I_{\text{sc1}}, I_{\text{sc2}}, I_p, I_l, I_1, I_d, \boldsymbol{u_I}\},\ Z_2 : \{G_t, G_p, G_s, \boldsymbol{u_m}\}$$

We note that Z_1 is linear whereas Z_2 is nonlinear. The interface variables G_t, G_p are selected as the variables whose ranges are to be decomposed to obtain a relational abstraction.

The meal input is modeled by a simple nondeterministic component that selects an upper bound on the amount of glucose appearing in the blood stream due to the meal at various times after the ingestion of the meal. The ranges for the nondeterministic meal input $u_m(t)$ is defined according to different time intervals: $u_m \in [0, 20]$ for $t \in [0, 30]$, $u_m \in [0, 10]$ for $t \in (30, 80]$, $u_m \in [0, 4]$ for $t \in (80, 360]$, $u_m \in [0, 2]$ for $t \in (360, 400]$, $u_m \in [0, 1]$ for $t \in (400, 500]$, and $u_m \in [0, 0]$ for $t \in (500, +\infty)$.

We consider three different controllers for deciding $u_I(t)$, the insulin input to the model. C_1, C_2 are a multi-basal artificial pancreas controller described in the recent work of Chen et al. [14]. C_3 is a PID control algorithm with saturation and anti-windup compensation, as described by Weinzimer et al. [45] (see also Cameron et al. [9]). Controllers based on this algorithm have undergone advanced stage clinical trials and form the basis for a product.

We consider two safety properties for our study. *Property 1:* The blood glucose level of the patient must not fall below 70 `mg/dl` (the clinical limit for hypoglycemia onset), i.e, $\Box(G_s \geq 70)$. *Property 2:* The blood glucose levels of the patient cannot exceed 300 `mg/dl` (the clinical limit for the onset of ketoacidosis), i.e, $\Box(G_s \leq 300)$. Since the system model here is with large uncertainties, we would like to find the largest step number below which the system is safe.

Table 2 gives the computation costs for generating relational abstractions. Because of the large uncertainties, we have to consider large tolerance $w_{\max}$, and still, compositional abstraction is much better than the monolithic one due to the small number of interface variables. For the safety proof, we always use the step size 0.2, the TM order 4, the cutoff threshold 10^{-8} and the precision 256 in Flow*. The results are better than using IC3 in most cases, however worse than those from using BMC. Although the time costs from Flow* seem to be better, the tool can not prove the safety for more than 7 steps. As we tried the step size 0.002 and TM order 6, the tool still found a counterexample at the 7th step after more than 1800s computation time.

5.2 Discussion

We briefly discuss our observations on the applicability of the relational abstraction method presented in the paper.
1. The proposed approach can better handle some nontrivial verification tasks. In some cases, we are even able to find inductive invariants through IC3, thanks to the relational abstraction.
2. The main focus of our paper was on building the abstraction compositionally, focussing especially on handling compositions with linear subsystems effectively. This applies to many systems, built from such compositions. Otherwise, the explosion in the number of subdivisions is unavoidable in a general nonlinear system.
3. Another key advantage is that we can subdivide over the interface variables between components rather than the entire state space while maintaining the same accuracy.
4. We note that once the abstract model is built, checking properties remains challenging using existing linear arithmetic SMT solvers that underlie model checkers such as NuXMV. First, a large number of subdivisions may be a computational burden for the model checker: a promising future direction is to consider subdivisions on the fly, adapting our approach to a SAT-modulo ODE approach used in solvers such as dReach and iSAT [25, 28, 30]. Furthermore, existing SMT solvers use infinite precision arithmetic to reason about the floating point coefficients of the abstractions. This reasoning is expensive, and the solving time would be greatly improved by adopting floating-point arithmetic.

6 RELATED WORK

The formal verification of nonlinear hybrid systems is an important problem that has led to numerous approaches. In light of this paper, we broadly classify them into the approaches that conclude reachability directly based on the system dynamics vs. those based on an abstraction of the system. The approach in this paper belongs to the latter category. Examples of the first category include deductive verification to prove properties [37], flowpipe construction [2, 12, 26, 31], simulation-based falsification [1, 21], level set methods [34] bounded-model checking [24, 25, 27, 28, 40] and a number of abstraction-based techniques discussed below.

There are many types of abstraction-based approaches for hybrid systems that are classified based on the type of abstraction sought: (a) abstraction to finite state systems vs. infinite state abstractions; and (b) separate abstraction of a plant and control vs. a joint abstraction of the closed-loop hybrid system as a whole.

Predicate abstraction approaches studied by Alur et al. focus on deriving finite-state abstractions using an initial set of predicates provided by the user [3, 4]. Further predicates are derived using counter-example guided refinement. Predicate abstractions are advantageous since they yield finite state systems that can be efficiently checked when the number of predicates is small. However, due to the exponential growth in the number of states as a function of the number of predicates added, these approaches suffer from state-space explosion especially for nonlinear dynamics. In contrast, our approach does not rely on predicates and constructs an infinite state abstraction in the form of discrete linear relations.

Another form of abstraction called *hybridization* involves the simplification of nonlinear dynamics by simpler dynamics such as affine inclusions or piecewise linear dynamics [19, 44]. This approach converts a nonlinear dynamics into piecewise affine or polyhedral inclusions by considering subdivisions of the state-space. An important difference between our approach and hybridization is that whereas hybridization abstracts the dynamics, our approach abstracts the solutions of the dynamics. As a result, our abstraction continues to be valid even when the dynamics leaves the partition corresponding to the initial condition. Whereas, hybridization can be applied up front to

Table 3. Verification results. Legend: *Model*: name of the verified benchmark. For the artificial pancreas model, we specify the abstraction that we used in each line (#4, #5, #6 in Table 2); *Property*: # of the property in verification; *Safe*: Y - the property is proved, N - a (possible) counterexample is found, U - unknown; T: time cost for verification, **TO**: > 600 seconds; *Abs. steps:* # of steps explored in the model checker; *Steps*: the corresponding # of the discrete steps of the system; Notice that if the result is unknown, then the system is safe up to this number of steps; –: Flow* fails to compute a valid overapproximation.

	Abstraction verification using SMT								Flow*		
	BMC				IC3						
Model / Property	**Safe**	**T** (sec)	**Steps**	**Abs. steps**	**Safe**	**T** (sec)	**Steps**	**Abs. steps**	**Safe**	**T** (sec)	**Steps**
Car/1	N	8.4	17	35	N	264.8	18	37	N	0.3	2
Car/2	U	71	7974	15965	Y	0.24	3	6	N	4.1	29
Car/3	N	6.8	8	17	U	TO	9	19	N	8.9	47
Car/4	N	0.2	9	18	N	2.3	7	15	N	1.1	10
Car/5	U	331	11	23	U	TO	11	23	F	–	–
Car/6	U	440	16	33	U	TO	15	31	F	–	–
Car/7	U	536	9	19	U	TO	10	21	F	–	–
Car/8	U	29.0	14	29	U	TO	13	26	F	–	–
Car/9	U	23.0	11	23	U	TO	14	29	F	–	–
Car/10	U	68.0	7974	15948	Y	0.04	1	3	F	–	–
Car/11	U	68.0	7974	15797	Y	0.03	1	3	F	–	–
Car/12	U	68.0	7974	15948	Y	0.01	1	2	F	–	–
Car/13	U	68.0	7974	15797	Y	0.01	1	2	F	–	–
Car/14	N	256.9	7	15	U	TO	6	13	N	2.6	21
Car/15	U	434.0	9	19	U	TO	14	29	N	6.8	40
Car/16	N	169.2	7	15	U	TO	6	13	N	1.8	16
Car/17	U	366.0	7	15	U	TO	8	17	F	–	–
Car/18	U	291.0	10	21	U	TO	14	29	F	–	–
Car/19	U	592.0	11	23	U	TO	16	32	F	–	–
Car/20	U	380	8	17	U	TO	13	27	F	–	–
Car/21	U	68.0	7974	15857	Y	0.01	1	2	F	–	–
Car/22	U	69.0	7974	15857	Y	0.01	1	2	F	–	–
Car/23	U	70.0	7974	15857	Y	0.02	1	2	F	–	–
Car/24	U	70.0	7974	15857	Y	0.02	1	2	F	–	–
Platoon/1	N	475.7	2	5	N	9.5	1	3	N	3.6	3
Platoon/2	N	9.4	1	3	U	TO	2	4	N	3.2	3
7 Car Platoon, #2	U	25.8	3	7	U	TO	3	7	N	8.2	3
7 Car Platoon, #4	U	103	2	4	U	TO	1	3			
Artificial Pancreas, C1, #5 / 1	U	179	20	203	U	TO	1	12	N	16	7
Artificial Pancreas, C1, #6 / 1	U	252	2	24	U	TO	0	4			
Artificial Pancreas, C1, #7 / 1	U	431	11	22	U	TO	3	6			
Artificial Pancreas, C1, #8 / 1	U	313.7	1	14	U	TO	0	5			
Artificial Pancreas, C2, #5 / 1	U	179	20	203	U	TO	1	12	N	17	7
Artificial Pancreas, C2, #6 / 1	U	251	2	24	U	TO	0	5			
Artificial Pancreas, C2, #7 / 1	U	303	11	22	U	TO	3	7			
Artificial Pancreas, C2, #8 / 1	U	231.8	1	14	U	TO	1	6			
Artificial Pancreas, C3, #5 / 1	U	185	20	203	U	TO	1	12	N	13	7
Artificial Pancreas, C3, #6 / 1	U	541	9	19	U	TO	0	5			
Artificial Pancreas, C3, #7 / 1	U	148	8	16	U	TO	2	5			
Artificial Pancreas, C3, #8 / 1	U	453.2	1	13	U	TO	0	5			

first create an abstraction and then verify it, most successful approaches pursue hybridization on the fly. However, in this paper, we do not compute the abstractions on-the-fly in order to allow us to use existing tools like NuXMV off-the-shelf.

Finally, the idea of a relational abstraction has been well-studied for linear systems, often for different purposes [35, 42, 47]. Tiwari and Sankaranarayanan consider the construction of relational abstractions $R(\vec{x}', \vec{x})$ that relate the current state of the system to any future state at an arbitrary time in the future [42]. However, computing these relations is often very difficult, requiring assumptions about the form of the dynamics. Further, these relations are imprecise for the purposes of verifying time triggered systems. Subsequently, Zutshi et al. proposed timed relational abstractions that relate $R(\vec{x}(\delta), \vec{x}(0))$ for a fixed time step δ [47], specifically for time-triggered controller verification. This

is the same form of relationalization used in this paper for nonlinear systems. Mover et al. proposed a time-aware scheme that computes relations of the form $R(\vec{x}(t), \vec{x}(0), t)$ with time t as an explicit argument [35]. In this paper, time aware relations of a more general form are considered as an intermediate step towards computing fixed time step relations. Podelski and Wagner propose *binary reachability relations* that are special relational abstractions for proving stability properties [39].

Our approach uses a decomposition based on a dependency graph wherever a "monolithic" system model is provided. A similar approach has been applied in the context of flowpipe construction through a similar decomposition [15]. A similar decomposition has also been used for stability analysis [43] and computing differential invariants [38]. If, however, the system is already provided as a composition of blocks, a decomposition can be avoided. Dragomir et al. provide a compositional semantics of Simulink block diagrams through a composition of "transformers" that have a similar flavor of the relational abstractions proposed here [22]. Whereas Dragomir et al. use a simple Euler scheme for integrator blocks, the purpose of this paper is to provide a sound mathematical abstraction for ODEs. Furthermore, Dragomir et al. use the ISABELLE theorem prover to handle properties of nonlinear functions arising from ODEs, whereas this paper uses Taylor models to abstract such functions through linear relations. The compositional construction of abstractions has also been considered in other approaches [29, 41]. However, the latter approaches rely on properties of the components such as input-to-state stability to enable a composition.

7 CONCLUSION

We have thus presented a relational abstraction approach for nonlinear systems that is compositional, using efficient SAT/SMT-based verification techniques for analyzing the resulting abstraction. Our experimental evaluation shows promising results, proving properties that the tool Flow* fails to establish. Our future work will explore the trade off between the precision of the relational abstraction and the computational cost of the verification procedure.

Acknowledgment. This work was supported in part by the US National Science Foundation (NSF) under award number CPS-1446900 and the Air Force Research Laboratory (AFRL). All opinions expressed are those of the authors and not necessarily of the NSF or AFRL.

REFERENCES

[1] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. 2013. Probabilistic Temporal Logic Falsification of Cyber-Physical Systems. *Trans. on Embedded Computing Systems (TECS)* 12, 2s (2013), 95:1–95:30.

[2] M. Althoff. 2015. An Introduction to CORA 2015. In *Proc. of ARCH'15 (EPiC Series in Computer Science)*, Vol. 34. EasyChair, 120–151.

[3] R. Alur, T. Dang, and F. Ivancic. 2003. Counter-Example Guided Predicate Abstraction of Hybrid Systems. In *Proc. of TACAS'03 (LNCS)*, Vol. 2619. Springer, 208–223.

[4] R. Alur, T. Dang, and F. Ivancic. 2003. Progress on Reachability Analysis of Hybrid Systems Using Predicate Abstraction. In *Proceedings of the 6th International Workshop on Hybrid Systems: Computation and Control (HSCC'03) (LNCS)*, Vol. 2623. Springer, 4–19.

[5] C. Baier and J.-P. Katoen. 2008. *Principles of Model Checking*. MIT Press.

[6] M. Berz. 1999. *Modern Map Methods in Particle Beam Physics*. Advances in Imaging and Electron Physics, Vol. 108. Academic Press.

[7] M. Berz and K. Makino. 1998. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. *Reliable Computing* 4 (1998), 361–369. Issue 4.

[8] A. R. Bradley. 2011. SAT-based Model Checking Without Unrolling. In *Proc. VMCAI'11 (Lecture Notes in Computer Science)*, Vol. 6538. Springer-Verlag, 70–87.

[9] F. Cameron, G. Fainekos, D. M. Maahs, and S. Sankaranarayanan. 2015. Towards a Verified Artificial Pancreas: Challenges and Solutions for Runtime Verification. In *Proc. of RV'15 (LNCS)*, Vol. 9333. 3–17.

[10] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. 2014. The nuXmv Symbolic Model Checker. In *CAV (Lecture Notes in Computer Science)*, Vol. 8559. 334–342.

[11] X. Chen. 2015. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models.* Ph.D. Dissertation. RWTH Aachen University.

[12] X. Chen, E. Ábrahám, and S. Sankaranarayanan. 2012. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. In *Proc. of RTSS'12*. IEEE Computer Society, 183–192.

[13] X. Chen, E. Ábrahám, and S. Sankaranarayanan. 2013. Flow*: An Analyzer for Non-linear Hybrid Systems. In *Proc. of CAV'13 (LNCS)*, Vol. 8044. Springer, 258–263.

[14] X. Chen, S. Dutta, and S. Sankaranarayanan. 2017. Formal Verification of a Multi-Basal Insulin Infusion Control Model. (2017). Cf. http://www.cs.colorado.edu/~srirams/projects/ap-verification-project-page.html.

[15] X. Chen and S. Sankaranarayanan. 2016. Decomposed Reachability Analysis for Nonlinear Systems. In *Proc. of the 37th IEEE Real-Time Systems Symposium (RTSS'16)*. IEEE Computer Society, 13–24.

[16] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. 2015. HyComp: An SMT-Based Model Checker for Hybrid Systems. In *Proc. of TACAS'15 (LNCS)*, Vol. 9035. Springer, 52–67.

[17] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. 2016. Infinite-state Invariant Checking with IC3 and Predicate Abstraction. *Form. Methods Syst. Des.* 49, 3 (Dec 2016), 190–218.

[18] C. Dalla Man, R. A. Rizza, and C. Cobelli. 2006. Meal simulation model of the glucose-insulin system. *IEEE Transactions on Biomedical Engineering* 1, 10 (2006), 1740–1749.

[19] T. Dang, C. Le Guernic, and O. Maler. 2009. Computing Reachable States for Nonlinear Biological Models. In *Proc. of CMSB'09 (LNCS)*, Vol. 5688. Springer, 126–141.

[20] L. M. de Moura and N. Bjørner. 2008. Z3: An Efficient SMT Solver. In *Proc. of TACAS'08 (LNCS)*, Vol. 4963. Springer, 337–340.

[21] A. Donzé. 2010. Breach: A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *CAV (Lecture Notes in Computer Science)*, Vol. 6174. Springer.

[22] Iulia Dragomir, Viorel Preoteasa, and Stavros Tripakis. 2016. Compositional Semantics and Analysis of Hierarchical Block Diagrams. In *SPIN'16 (Lecture Notes in Computer Science)*, Vol. 9641. Springer, 38–56.

[23] B. Dutertre and L. de Moura. 2006. The YICES SMT Solver. (2006). Cf. http://yices.csl.sri.com/tool-paper.pdf.

[24] A. Eggers, N. Ramdani, N. Nedialkov, and M. Fränzle. 2011. Improving SAT Modulo ODE for Hybrid Systems Analysis by Combining Different Enclosure Methods. In *Proc. of SEFM'11 (LNCS)*, Vol. 7041. Springer, 172–187.

[25] M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. 2007. Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. *JSAT—Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on SAT/CP Integration* 1 (2007), 209–236.

[26] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Proc. of CAV'11 (LNCS)*, Vol. 6806. Springer, 379–395.

[27] S. Gao, S. Kong, and E. M. Clarke. 2013. dReal: An SMT Solver for Nonlinear Theories over the Reals. In *Proc. CADE'13 (Lecture Notes in Computer Science)*, Vol. 7898. Springer, 208–214.

[28] S. Gao, S. Kong, and E. M. Clarke. 2013. Satisfiability Modulo ODEs. In *Proceedings of the 13th International Conference on Formal Methods in Computer-Aided Design (FMCAD'13)*. IEEE, 105–112.

[29] Z. Huang and S. Mitra. 2014. Proofs from simulations and modular annotations. In *Proc. of HSCC'14*. ACM, 183–192.

[30] S. Kong, S. Gao, W. Chen, and E. M. Clarke. 2015. dReach: δ-Reachability Analysis for Hybrid Systems. In *Proc. of TACAS'15 (LNCS)*, Vol. 9035. Springer, 200–205.

[31] C. Le Guernic. 2009. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics.* Ph.D. Dissertation. Université Joseph Fourier.

[32] K. Makino and M. Berz. 2003. Taylor models and other validated functional inclusion methods. *J. Pure and Applied Mathematics* 4, 4 (2003), 379–456.

[33] C. Dalla Man, M. Camilleri, and C. Cobelli. 2006. A System Model of Oral Glucose Absorption: Validation on Gold Standard Data. *Biomedical Engineering, IEEE Transactions on* 53, 12 (2006), 2472 –2478.

[34] I. Mitchell and C. Tomlin. 2000. Level Set Methods for Computation in Hybrid Systems.. In *Proc. of HSCC'00 (LNCS)*, Vol. 1790. Springer, 310–323.

[35] S. Mover, A. Cimatti, A. Tiwari, and S. Tonetta. 2013. Time-aware relational abstractions for hybrid systems. In *EMSOFT'13*. 1–10.

[36] M. Neher, K. R. Jackson, and N. S. Nedialkov. 2006. On Taylor Model Based Integration of ODEs. *SIAM J. Numer. Anal.* 45 (2006), 236–262. Issue 1.

[37] A. Platzer. 2010. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.* Springer.

[38] A. Platzer and E. M. Clarke. 2009. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design* 35, 1 (2009), 98–120.

[39] A. Podelski and S. Wagner. 2007. A Sound and Complete Proof Rule for Region Stability of Hybrid Systems. In *Proc. of HSCC'07 (LNCS)*, Vol. 4416. Springer, 750–753.

[40] N. Ramdani and N. S. Nedialkov. 2011. Computing Reachable Sets for Uncertain Nonlinear Hybrid Systems using Interval Constraint-Propagation Techniques. *Nonlinear Analysis: Hybrid Systems* 5, 2 (2011), 149–162.
[41] M. Rungger and M. Zamani. 2015. Compositional construction of approximate abstractions. In *Proc. of HSCC'15*. ACM, 68–77.
[42] S. Sankaranarayanan and A. Tiwari. 2011. Relational Abstractions for Continuous and Hybrid Systems. In *Proc. of CAV'11 (LNCS)*, Vol. 6806. Springer, 686–702.
[43] D. Šiljak. 1978. *Large-scale dynamic systems: stability and structure.* North Holland.
[44] R. Testylier and T. Dang. 2013. NLTOOLBOX: A Library for Reachability Computation of Nonlinear Dynamical Systems. In *Proc. of ATVA'13 (LNCS)*, Vol. 8172. Springer, 469–473.
[45] S Weinzimer, G Steil, K Swan, J Dziura, N Kurtz, and W. Tamborlane. 2008. Fully Automated Closed-Loop Insulin Delivery Versus Semiautomated Hybrid Control in Pediatric Patients With Type 1 Diabetes Using an Artificial Pancreas. *Diabetes Care* 31 (2008), 934–939.
[46] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski. 2013. A Trajectory Splicing Approach to Concretizing Counterexamples for Hybrid Systems. In *IEEE Conf. on Decision and Control (CDC)*. IEEE Press.
[47] A. Zutshi, S. Sankaranarayanan, and A. Tiwari. 2012. Timed Relational Abstractions for Sampled Data Control Systems. In *Proc. of CAV'12 (LNCS)*, Vol. 7358. Springer, 343–361.