Relational Differential Dynamic Logic

Juraj Kolčák1,2, Ichiro Hasuo1,3, Jérémy Dubut1,4, Shin-ya Katsumata1, David Sprunger1, Akihisa Yamada1
1National Institute of Informatics, Japan; 2LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France;
3The Graduate University for Advanced Studies (SOKENDAI), Japan; 4Japanese-French Laboratory for Informatics, Japan

Introduction

Cyber-physical systems (CPS) are becoming exceedingly common in industry, including numerous safety-critical applications (e.g., automated driving), making quality assurance of CPS in important issue. CPS are generally characterized by a union of continuous (physical) dynamics and discrete (digital) control, referred to as a hybrid system. Verification of hybrid systems is an intriguing challenge bringing together two research communities of control theory and formal methods focused on continuous and discrete dynamics respectively. To handle infinite state space, we turn to deductive verification. In particular, we make use of differential dynamic logic [2], a well established and powerful tool for reasoning about continuous dynamics.

Motivating Example

Our industry collaboration revealed an interesting application summarized in the following example. Imagine two cars with the following dynamics, where $x$, $x^2$, and $v$, $\dot{v}$ denote their positions and velocities, respectively, and an equality obstacle $\dot{x} = v$, $\dot{v} = 1$. $\ddot{x} = v^2$, $\ddot{v} = 2$.

Our aim is then to ascertain the following natural claim:

Starting from the initial state $x_0 = x_0^2 = v_0 = \dot{v}_0 = 0$ and observing the two dynamics above, we obtain $v_0 = x_0^2$ when $x = \dot{x} = 1$.

Figure 1 displays an ad-hoc proof of the claim using closed-form solutions of the dynamics. (The bold areas represent $x$, $\dot{x}$; Instead of global properties (e.g., solutions) of the dynamics, we want to rely on local reasoning. Local reasoning is well established in differential dynamic logic dl, in particular via differential invariant (DI). In fact, one can view (DI) as a continuous version of loop invariants. It turns out, however, that local relational reasoning, e.g. with two differential dynamics, is not easy in dl.

Our Contribution

We introduce time stretch functions as means of reasoning about relational invariant properties. We built time stretch functions into a proof rule which allows us to synchronise the two dynamics on an essential property (e.g. the distance travelled). Left with single differential dynamics, we unlock the full potential of invariant reasoning in dl. Along with the time stretch (TS) rule, we propose several other rules for relational reasoning. Although the rules appear new to us, they might be derivable in the comprehensive calculus in [1].

We showcase our proof rules in several case studies. Our methods are especially well suited for non-linear properties (e.g. for test case evaluation), or formally relating concrete models and their abstractions.

Differential Dynamic Logic

In a nutshell, dl is a modal logic with the modal operators realised as hybrid programs. Hybrid programs allow mixing discrete operators such as assignment, $x := 0$ or test, $\text{true}$, with differential dynamics $[x = f(x), \dot{x} = 0]$.

The modality about to any state reachable by $\dot{x} = f(x)$ provided $Q(x)$ holds at all times. Refer to literature [1] for full exposure to dl.

Relational Formulas

We expect the relational property to be captured by relational differential dynamics formulas (RDD formulas) $\vec{x} = f(\vec{x}) \land Q(\vec{x})$.

Or on a shorthand with $\delta \vec{x} := f(\vec{x}) \land Q(\vec{x})$ and $\delta^2 \vec{x} := f(\vec{x}) \land Q(\vec{x})$:

$\{ \vec{x} \mid \delta^2 \vec{x} \}$ $B$

where $\vec{x}$ and $\vec{z}$ are disjoint and $E, B$ are formulas on both $\vec{x}$ and $\vec{z}$. An RDD formula is intended as "after any execution of $\vec{x} = f(\vec{x}) \land Q(\vec{x})$ and $\delta^2 \vec{x} = f(\vec{x}) \land Q(\vec{x})$ concurrently, e.g. allowing different amount of time for the two dynamics, and synchronisation on $E$, $B$ must hold." We assume the following equality form of the test $E \equiv g(x) = g(\dot{x})$.

Time Stretch Function

In a sense, $B$ in RDD formulas is an relational invariant - or, literally speaking, a bisimulation. An RDD formula is in fact true if the test $E$, which relates elapsed times of the two dynamics, aligns with the simulation. Moreover, if $g$ and $\dot{g}$ are monotonic in time, we get a unique solution: $\delta^2 \vec{x} = f(\vec{x}) \land Q(\vec{x})$.

Here $\delta g\vec{x}$ is the Lie derivative of $g$ and $g(\dot{x})$ at $t$ the solution at time $t$. The above relationship allows us to "stretch" the time of the second "sharper" dynamics to match the first dynamics.

The time stretching can be followed by any established dl rules using single dynamics. We list the dl rules used in the case studies in the table below.

Selected dl Proof Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash a \equiv c$</td>
<td>$\Gamma$ is a deduction, $a$ is an atomic formula, $c$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash x \equiv x$</td>
<td>The variable $x$ is in the deduction $\Gamma$</td>
</tr>
<tr>
<td>$\Gamma \vdash a \equiv b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
</tbody>
</table>

References


Case Study: Cars with Constant Acceleration

For the first case study we consider a generalisation of the motivating example, where the accelerations are parameters, however, considered constant.

We thus consider the following dynamics $\delta^2 \vec{x}$ and initial condition $\Gamma$ for the cars. We also use $\delta^2 \vec{x}$ to denote the synchronised dynamics $\delta^2 \vec{x} = f(\vec{x}) \land Q(\vec{x})$.

Motivating Example

We expect the relational property to be captured by relational differential dynamics formulas (RDD formulas) $\vec{x} = f(\vec{x}) \land Q(\vec{x})$.

Or on a shorthand with $\delta \vec{x} := f(\vec{x}) \land Q(\vec{x})$ and $\delta^2 \vec{x} := f(\vec{x}) \land Q(\vec{x})$:

$\{ \vec{x} \mid \delta^2 \vec{x} \}$ $B$

where $\vec{x}$ and $\vec{z}$ are disjoint and $E, B$ are formulas on both $\vec{x}$ and $\vec{z}$. An RDD formula is intended as "after any execution of $\vec{x} = f(\vec{x}) \land Q(\vec{x})$ and $\delta^2 \vec{x} = f(\vec{x}) \land Q(\vec{x})$ concurrently, e.g. allowing different amount of time for the two dynamics, and synchronisation on $E$, $B$ must hold." We assume the following equality form of the test $E \equiv g(x) = g(\dot{x})$.

Time Stretch Function

In a sense, $B$ in RDD formulas is an relational invariant - or, literally speaking, a bisimulation. An RDD formula is in fact true if the test $E$, which relates elapsed times of the two dynamics, aligns with the simulation. Moreover, if $g$ and $\dot{g}$ are monotonic in time, we get a unique solution: $\delta^2 \vec{x} = f(\vec{x}) \land Q(\vec{x})$.

Here $\delta g\vec{x}$ is the Lie derivative of $g$ and $g(\dot{x})$ at $t$ the solution at time $t$. The above relationship allows us to "stretch" the time of the second "sharper" dynamics to match the first dynamics.

The time stretching can be followed by any established dl rules using single dynamics. We list the dl rules used in the case studies in the table below.

Selected dl Proof Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma \vdash a \equiv c$</td>
<td>$\Gamma$ is a deduction, $a$ is an atomic formula, $c$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash x \equiv x$</td>
<td>The variable $x$ is in the deduction $\Gamma$</td>
</tr>
<tr>
<td>$\Gamma \vdash a \equiv b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
<tr>
<td>$\Gamma \vdash a \leftarrow b$</td>
<td>$a$ is an atomic formula, $b$ is a complex term</td>
</tr>
</tbody>
</table>

References


Acknowledgements

Thanks are due to Kenji Yoshida, Yoshiyuki Shimizu and Takanori Seto from Maeda Motion Control for helpful discussion. The authors are supported by JST ERATO BAS10 Mechatronics for robot design project (No. JPMJER1610), RIT, J.K. is supported by ASR-FBS project "AlgebraCdt" ASR-16-CES-0031. HJ is supported by Grant-in-Aid No. KAKENHI 22H0562, JST.

These presented here are developed on arXiv: https://arxiv.org/abs/1903.00153