idDL2DL: translating interval specifications to $d\mathcal{L}$

Jaime Santos$^1$    Daniel Figueiredo$^2$

$^1$University of Minho, Braga, Portugal & $^2$CIDMA – University of Aveiro, Portugal

26th International Workshop on Algebraic Development Techniques
Aveiro, June 28–30, 2022
Outline

- Differential dynamic logic ($d\mathcal{L}$).
  KeYmaera X

- Interval arithmetics.

- Interval paradigm in $d\mathcal{L}$.
  Syntax
  Semantics

- idDL2DL.
  Example

- Conclusions and future work
Differential dynamic logic ($d\mathcal{L}$).

Differential dynamic logic ($d\mathcal{L}$) was developed by Platzer and is able to describe both continuous and discrete (hybrid) dynamics. It is a dynamic logic with a first order structure which considers two kinds of atomic programs:

- Discrete Jump Sets ($x := a$)
- Continuous Evolutions ($x' = f(x_1, ..., x_n)$)

We can obtain hybrid behaviors if we combine both kinds of hybrid programs.

This logic can be used to specify safety-critical systems.
Differential dynamic logic ($d\mathcal{L}$).

In $d\mathcal{L}$ language, we admit a set $V$ of logical variables and a set $\Sigma$ of signatures. In $\Sigma$ we have:

- Predicate symbols (for instance $=$, $<$ and $>$).
- Function symbols (for instance $+$, $-$, $/$ and $\cdot$) which includes constants (functions with arity 0) as 0 and 1.
- State variables whose interpretation is not rigid (can vary during the evolution model). We denote by $\Sigma_{fl} \subseteq \Sigma$ the set of state variables.

Thus, we define the set of terms ($Trm(V, \Sigma)$) as the smaller set that contains both $V$, $\Sigma_{fl}$ and such that $f(t_1, \ldots, t_n) \in Trm(V, \Sigma)$ for any $f \in \Sigma$ with arity $n$ (possibly 0) and any $t_1, \ldots, t_n \in Trm(V, \Sigma)$. 
Differential dynamic logic ($d\mathcal{L}$).

The set of first-order formulas of $d\mathcal{L}$, $\text{Fml}_{\text{FOL}}(\mathcal{V}, \Sigma)$, is the least set that contains $p(t_1, \ldots, t_n)$, $\bot$, $\top$, $\varphi \lor \psi$, $\varphi \land \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$, $\neg \varphi$, $\exists x \varphi$, $\forall x \varphi$ for any predicate $p$ of arity $n$, any terms $t_i$, any $x \in \mathcal{V}$ and any $\varphi$, $\psi \in \text{Fml}_{\text{FOL}}(\mathcal{V}, \Sigma)$.

The set of hybrid programs, $\text{HP}(\mathcal{V}, \Sigma)$, is the least set that contains $(a_1 := t_1, \ldots, a_n = t_n)$, $(a'_1 = t_1, \ldots, a'_n = t_n \land \chi)$, $?\chi$, $\alpha; \beta$, $\alpha \cup \beta$, $\alpha^*$ for any $a_1, \ldots, a_n \in \Sigma_{fl}$, any $t_1, \ldots, t_n \in \text{Trm}(\mathcal{V}, \Sigma)$, any $\chi \in \text{Fml}_{\text{FOL}}(\mathcal{V}, \Sigma)$ and any $\alpha, \beta \in \text{HP}(\mathcal{V}, \Sigma)$.

The set of formulas of $d\mathcal{L}$, $\text{Fml}(\mathcal{V}, \Sigma)$, is the least set that contains $\text{Fml}_{\text{FOL}}(\mathcal{V}, \Sigma)$ and such that $[\alpha] \varphi$, $\langle \alpha \rangle \varphi \in \text{Fml}(\mathcal{V}, \Sigma)$ for any $\alpha \in \text{HP}(\mathcal{V}, \Sigma)$ and any $\varphi \in \text{Fml}(\mathcal{V}, \Sigma)$.
Differential dynamic logic ($d\mathcal{L}$).

A valuation $val_{l,\eta}(v, \cdot) : Fml(V, \Sigma) \to \{true, false\}$ is a function such that:

- $l$ is an interpretation and it is defined into $\Sigma \setminus \Sigma_{fl}$, the set of predicate and function symbols.
- $v$ is a state and $v : \Sigma_{fl} \to \mathbb{R}$.
- $\eta$ is called an assignment and $\eta : V \to \mathbb{R}$.

$Sta(\Sigma)$ is the set of all states.

$$val_{l,\eta}(v, x) = \eta(x), \text{ for any } x \in V.$$  

$$val_{l,\eta}(v, u) = v(u), \text{ for any } u \in \Sigma_{fl}.$$  

For any function symbol $f$ and terms $t_i$,  

$$val_{l,\eta}(v, f(t_1, \ldots, t_n)) = l(f)(val_{l,\eta}(v, t_1), \ldots, val_{l,\eta}(v, t_n)).$$
Differential dynamic logic ($d\mathcal{L}$).

For any predicate symbol $p$ and terms $t_i$,
\[
\text{val}_{I,\eta}(v, p(t_1, ..., t_n)) = true \iff l(p)(\text{val}_{I,\eta}(v, t_1), ..., \text{val}_{I,\eta}(v, t_n)) = true.
\]
\[
\text{val}_{I,\eta}(v, \bot) = false.
\]
\[
\text{val}_{I,\eta}(v, \top) = true.
\]
\[
\text{val}_{I,\eta}(v, \neg \phi) = true \iff \text{val}_{I,\eta}(v, \phi) = false.
\]
\[
\text{val}_{I,\eta}(v, \phi \lor \psi) = true \iff \text{val}_{I,\eta}(v, \phi) = true \text{ or } \text{val}_{I,\eta}(v, \psi) = true.
\]
\[
\text{val}_{I,\eta}(v, \phi \land \psi) = true \iff \text{val}_{I,\eta}(v, \phi) = true \text{ and } \text{val}_{I,\eta}(v, \psi) = true.
\]
\[
\text{val}_{I,\eta}(v, \phi \rightarrow \psi) = true \iff \text{val}_{I,\eta}(v, \phi) = false \text{ or } \text{val}_{I,\eta}(v, \psi) = true.
\]
\[
\text{val}_{I,\eta}(v, \phi \leftrightarrow \psi) = true \iff \text{val}_{I,\eta}(v, \phi \rightarrow \psi) = true \text{ and } \text{val}_{I,\eta}(v, \psi \rightarrow \phi) = true.
\]
Differential dynamic logic (DDL).

\[
\text{val}_I,\eta(v, \exists x \varphi) = \text{true} \iff \text{val}_I,\eta[x\mapsto d](v, \varphi) = \text{true} \quad \text{for some } d \in \mathbb{R}
\]

\[
\text{val}_I,\eta(v, \forall x \varphi) = \text{true} \iff \text{val}_I,\eta[x\mapsto d](v, \varphi) = \text{true} \quad \text{for any } d \in \mathbb{R}
\]

\[
\text{val}_I,\eta(v, \langle \alpha \rangle \varphi) = \text{true} \iff \text{val}_I,\eta(w, \varphi) = \text{true} \quad \text{for some state } w \text{ that can be reached from } v \text{ executing the hybrid program } \alpha
\]

\[
\text{val}_I,\eta(v, [\alpha] \varphi) = \text{true} \iff \text{val}_I,\eta(w, \varphi) = \text{true} \quad \text{for any state } w \text{ that can be reached from } v \text{ executing the hybrid program } \alpha
\]

We write \( I, \eta, v \models \varphi \iff \text{val}_I,\eta(v, \varphi) = \text{true} \) and we say that \( \varphi \) is satisfied in \( I, \eta, v \).

We say that a formula \( \varphi \) is valid and we write \( \models \varphi \) if \( I, \eta, v \models \varphi \) for any \( I, \eta, v \).
Differential dynamic logic ($dL$).

- $H$ - initial height of the ball;
- $h$ - actual height of the ball;
- $g$ - gravitational acceleration;
- $c$ - elasticity constant (between 0 and 1);
- $v$ - vertical velocity.

The evolution of the system can be described by the following hybrid program:

$$bounce \equiv (h' = v, v' = -g \land h \geq 0; (?h > 0 \cup (?h = 0; v := -cv)))*$$

$$0 < h < H \rightarrow [bounce]h < H$$
Differential dynamic logic ($\mathcal{DL}$).

KeYmaera X

$\mathcal{DL}$ has a proof calculus which is proven to be sound, although not complete.

KeYmaera X is a semiautomatic prover for $\mathcal{DL}$. It applies the proof calculus of $\mathcal{DL}$ and asks for help when it is not able to close a proof.

If KeYmaera is not able to close a proof it may be either the case that the input formula is not valid. KeYmaera provides the formula(s) which was not able to prove.

This features can be used to provide safety conditions.

KeYmaera X calls Mathematica for solving complex differential equations. However, proving strategies are embedded in the software in order to reduce the dependence of external software.
Interval arithmetics.

In general, when working with differential equations, small perturbations in the initial conditions can lead to great changes in the continuous evolutions.

This turns to be a relevant issue in real life systems since it is impossible to measure exact values for variables like distance and velocity.

The solution proposed is to consider an interval version of \( dL \), i.e., a version of \( dL \) where the logical and state variable are interpreted as closed intervals \([a, b]\) instead of reals.

Sicun Gao developed a language and a tool to reason about these systems using intervals (dReach). However, not all properties can be proved in that tool.
Interval arithmetics.

Let $\mathcal{I} (\mathbb{R})$ be the set of closed intervals of $\mathbb{R}$ with the form $[a, b]$, for $a, b \in \mathbb{R}$ with $a \leq b$. Moore introduced the usual operations for the set $\mathcal{I} (\mathbb{R})$. These are called interval arithmetics.

- $[a, b] + [c, d] = [a + c, b + d]$
- $-[a, b] = [-b, -a]$
- $[a, b] \cdot [c, d] = [\min(P), \max(P)]$ where $P = \{a \cdot c, a \cdot d, b \cdot c, b \cdot d\}$
- $[a, b]^{-1} = [\frac{1}{b}, \frac{1}{a}]$ provided that $0 \in ]a, b[$

Given a real function, $f : \mathbb{R}^n \to \mathbb{R}$, we can obtain $f^\mathcal{I} (\mathbb{R}) : \mathcal{I} (\mathbb{R})^n \to \mathcal{I} (\mathbb{R})$.

**Correctness:** $x_1 \in A_1, \ldots, x_n \in A_n \implies f(x_1, \ldots, x_n) \in f^\mathcal{I} (\mathbb{R})(A_1, \ldots, A_n)$

**Optimality:** $f_1^\mathcal{I} (\mathbb{R}), g$ are correct $\implies f_1^\mathcal{I} (\mathbb{R})(A_1, \ldots, A_n) \subseteq g(A_1, \ldots, A_n)$

In general, we can define $f^\mathcal{I} (\mathbb{R})$ such that $f^\mathcal{I} (\mathbb{R})(X) = [\inf_{x \in X} f(x), \sup_{x \in X} f(x)]$. 
Interval paradigm in $d\mathcal{L}$.

Syntax

We define variables as intervals instead of reals number for the variables of $d\mathcal{L}$.

An interval can be assigned to a variable in the following way:

$$x := [a, b] \equiv (x \geq a \land x \leq b)$$

Real values still can be assigned as degenerated intervals $x = [a, a]$.

The syntax of the interval version of $d\mathcal{L}$ is “the same” but we adapt it to make it more user friendly when working with intervals or uncertainty.

idDL2DL is a tool which reads an “interval formula” and automatically translates it into $d\mathcal{L}$ base syntax and runs KeYmaera in order to try to prove it.
Interval paradigm in $d\mathcal{L}$.

Semantics.

- $I$ is a usual interpretation (but for the interval context).
- $v$ is a state and $v : \Sigma_{fl} \rightarrow \mathcal{I}(\mathbb{R})$.
- $\eta$ is called an assignment and $\eta : V \rightarrow \mathcal{I}(\mathbb{R})$.

\[
\text{val}_{I,\eta}(v, x) = \eta(x), \text{ for any } x \in V.
\]

\[
\text{val}_{I,\eta}(v, u) = v(u), \text{ for any } u \in \Sigma_{fl}.
\]

For any function symbol $f$ and terms $t_i$,
\[
\text{val}_{I,\eta}(v, f(t_1, ..., t_n)) = I(f)^{\mathcal{I}(\mathbb{R})}(\text{val}_{I,\eta}(v, t_1), ..., \text{val}_{I,\eta}(v, t_n)).
\]
Intervalar paradigm in $d\mathcal{L}$.

Semantics.

Given an $n$-ary predicate $P$ over reals, we define $P^{I(\mathbb{R})}$, a predicate over $I(\mathbb{R})$ such that $P^{I(\mathbb{R})}(A_1, ..., A_n) = true \iff \forall a_1 \in A_1, ..., a_n \in A_n, P(a_1, ..., a_n) = true$

**Example**

$A \leq^{I(\mathbb{R})} B \iff \forall a \in A, \forall b \in B, a \leq b$

For any predicate symbol $P$ and terms $t_i$, $\text{val}_{I,\eta}(v, P(t_1, ..., t_n)) = I(P)^{I(\mathbb{R})}(\text{val}_{I,\eta}(v, t_1), ..., \text{val}_{I,\eta}(v, t_n))$

The interpretation for quantifiers and the remaining Boolean and modal operators is defined as expected.
Intervalar paradigm in $d\mathcal{L}$. 

Semantics.

$v$ can be reached from $w$ executing $(x_1 := t_1, ..., x_n := t_n)$ if, for any $i \in \{1, ..., n\}$, $v(x_i) = val_{I,\eta}(w, t_i)$ and $v|_{\Sigma_f\setminus\{x_1, ..., x_n\}} = w|_{\Sigma_f\setminus\{x_1, ..., x_n\}}$

$v$ can be reached from $w$ executing $(x'_1 = t_1, ..., x'_n = t_n \& \chi)$ if $v|_{\Sigma_f\setminus\{x_1, ..., x_n\}} = w|_{\Sigma_f\setminus\{x_1, ..., x_n\}}$ and, for each $i \in \{1, ..., n\}$ and for some fixed $t' \geq 0$, $v(x_i) = \bar{f}_i(t')(val_{I,\eta}(w, x_1), ..., val_{I,\eta}(w, x_n))$ where:

- $(x_1(t), ..., x_n(t)) = (f_1(t), ..., f_n(t))(a_1, ..., a_n)$ is the solution of the differential equation for initial conditions $x_i(0) = a_i$
- for any $t \in [0, t']$ which generates a state $u$ obtained from $v$ according to the same differential equations and initial conditions, $val_{I,\eta}(u, \chi) = true$.

The valuation is extended to all formulas in the same way as before.
Intervalar paradigm in $\mathcal{DL}$. Semantics.

Example of a continuous evolution restricted to the domain $[a, d]$. 
The goal of idDL2DL is to be a user friendly tool, whose language was designed for interval/uncertainty contexts.

The tool translates a formula in interval syntax to KeYmaera X syntax.

The translated formula can then be used as input of KeYmaera X to check if the property holds.

It is implemented in python.

https://github.com/JaimePSantos/idDL2DL/blob/main/README.md
Biological regulatory network are systems composed of several cell component such as proteins, mRNA, gens, and other organs.

The cell metabolism can be described by the concentration of their components, whose dynamic are described using a system of differential equations.

Some components induce the production of another (such as mRNA encoding the production of a protein) while others can inhibit the creation of other components (such as a protein which stops a gene from being translated into mRNA).

The resulting system of differential equations is complex and nonlinear. Preliminary analysis are performed with simplified models such as PieceWise Linear (PWL).
PWL models divide the original state space into several proper open domains and corresponding boundaries.

Within each domain, the dynamics of the system are simplified using a linear differential equation.

When the trajectory of the system’s evolution reaches a boundary, the system considers that a new domain has been reached and the new corresponding linear differential equations guide the system’s evolution.

The system is composed of continuous evolutions and discrete events (the change of domain).

The great uncertainty associated to a cell’s component concentration requires an interval approach in order to robustly prove safety conditions.
idDL2DL

Example

\[
\begin{align*}
x' &= 5 \frac{x^2}{x^2 + 2^2} \cdot \frac{2^2}{y^2 + 2^2} - x \\
y' &= 3 \frac{x^2}{x^2 + 4^2} - y
\end{align*}
\]

<table>
<thead>
<tr>
<th>(x' = -x)</th>
<th>(y' = -y)</th>
<th>(0 \leq x &lt; 2)</th>
<th>(2 &lt; x &lt; 4)</th>
<th>(4 &lt; x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y' = -y)</td>
<td>(2 &lt; y)</td>
<td>(2 &lt; y)</td>
<td>(2 &lt; y)</td>
<td>(2 &lt; y)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x' = -x)</th>
<th>(y' = -y)</th>
<th>(0 \leq x &lt; 2)</th>
<th>(2 &lt; x &lt; 4)</th>
<th>(4 &lt; x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x' = 5 - x)</td>
<td>(y' = -y)</td>
<td>(2 &lt; x &lt; 4)</td>
<td>(2 &lt; x &lt; 4)</td>
<td>(2 &lt; x)</td>
</tr>
<tr>
<td>(y' = 3 - y)</td>
<td>(0 &lt; y &lt; 2)</td>
<td>(0 &lt; y &lt; 2)</td>
<td>(0 &lt; y &lt; 2)</td>
<td>(0 &lt; y &lt; 2)</td>
</tr>
</tbody>
</table>
Example

\[ bio_0 \equiv ?(x < 2); (x' = -x, y' = -y \& (x < 2)) \]

\[ bio_{10} \equiv ?(2 < x \land x < 4 \land 0 < y \land y < 2); (x' = 5 - x, y' = -y \& (2 < x \land x < 4 \land 0 < y \land y < 2)) \]

\[ bio_{20} \equiv ?(4 < x \land 0 < y \land y < 2); (x' = 5 - x, y' = 3 - y \& (4 < x \land 0 < y \land y < 2)) \]

\[ bio_{11} \equiv ?(2 < x \land x < 4 \land 2 < y); (x' = -x, y' = -y \& (2 < x \land x < 4 \land 2 < y)) \]

\[ bio_{21} \equiv ?(4 < x \land 2 < y); (x' = -x, y' = 3 - y \& (4 < x \land 2 < y)) \]

\[ bio \equiv (bio_0 \cup bio_{10} \cup bio_{20} \cup bio_{11} \cup bio_{21})^* \]

"when the concentrations \( x \) and \( y \) are around 5.5 and 3.5, the biological system will never reach a state where \( x < 2 \)"

\[ [x := [5, 6]; y := [3, 4] [bio] \rightarrow x > 2 \]
idDL2DL

Example

## idDL2DL v0.1

Input in interval ddL:

```
> [{x:=[5,6];y:=[3,4]}] [{ (x < 2); {x'=-x, y'=-y & x<2} 
  | (2<x AND x<4 AND 0<y AND y<2); { x' = 5-x , y'=-y & (2 < x AND x < 4 AND 0<y AND y<2)) 
  | (? ( 4 < x AND 0<y AND y<2 ) ; { x' =5-x , y'=3-y & ( 4<x AND 0 <y AND y <2 )} 
  | (? ( 2 < x AND x < 4 AND 2 < y ) ; { x' = -x , y' = -y & (2 < x AND x < 4 AND 2 < y)} 
  | (? ( 4 < x AND 2 < y ) ; { x' = -x , y'=3 - y & ( 4 < x AND 2 < y )}]**} (x>2)
```

Output in dDL:

```
> (5<=a & a<=6) & (3<=b & b<=4) 
  -> ( [ x := a; y := b; ] [ { ?( x < 2 ); { x' = -x , y' = -y & x < 2 } 
  | (2 < x & x < 4 & 0 < y & y < 2) ; { x' = 5 - x , y' = -y & (2 < x & x < 4 & 0 < y & y < 2) } 
  | (?) (4 < x & 0 < y & y < 2) ; { x' = 5 - x , y' = 3 - y & (4 < x & 0 < y & y < 2) } 
  | (2 < x & x < 4 & y < 2) ; { x' = -x , y' = -y & (2 < x & x < 4 & 2 < y) } 
  | (?) (4 < x & 2 < y) ; { x' = -x , y' = 3 - y & (4 < x & 2 < y) }]**} (x > 2))
```

## 2021-07-31 20:09:04.169821

## Execution time: 0.009601s

### Proof: ✔️ All goals closed

Provable ( —> (5<=a&a<=6)&3<=b&b<=4->[x:=a;y:=b;][{?x < 2;{x'=-x,y'=-y&x < 2}++?2 < x & x < 4&0 < y&y < 2;{x'=5-x,y'=-y&2 < x&x < 4&0 < y&y < 2}++?4 < x&0 < y&y < 2;{x'=5-x,y'=-y&4 < x&0 < y&y < 2}++?2 < x&x < 4&2 < y;{x'=-x,y'=-y&2 < x&x < 4&2 < y}++?4 < x&2 < y;{x'=-x,y'=-3-y&4 < x&2 < y}]*]x>2 proved)

### Tactic to Reproduce the Proof

auto
Conclusions and future work

This interval version of $d\mathcal{L}$ aims to ease the application of $d\mathcal{L}$ tools in contexts where intervals are necessary.

We make use of the proof calculus of $d\mathcal{L}$ to intervals.

KeYmaera X is a semi-prover developed for $d\mathcal{L}$. We provide a tool which traduces interval formulas into $d\mathcal{L}$ ones.

As example, we apply to piecewise linear models, which are used in biological regulatory networks to study the interaction between gens, proteins and other organels.

A complement to dReach, which is very useful for reachable properties.