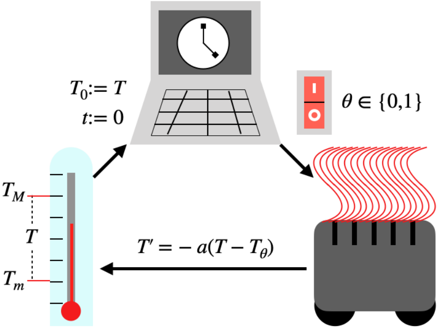
Differential Hoare Logics and Refinement Calculi for Hybrid Systems with Isabelle/HOL

Simon Foster[1] Jonathan Julián Huerta y Munive[2] Georg Struth[2]

University of York, UK

University of Sheffield, UK

Verification of Hybrid Systems



$$
\begin{aligned}
\text{dynamics} &= T' = -a(T - T_\theta) \\
\text{pre} &= T_m \leq T \leq T_M \\
\text{pos} &= T_m \leq T \leq T_M \\
\text{control} &= t := 0 \,;\, T_0 := T \,;\, \ldots \\
\text{therm} &= (\text{control} \,;\, \text{dynamics})^*
\end{aligned}
$$

hybrid program

$\{\text{pre}\}\ \text{therm}\ \{\text{pos}\}$

correctness spec

Previous Work

- Isabelle/HOL verification components for hybrid programs that
  - benefit from huge, impressive libraries of topology, analysis, ODEs;
  - based on MKA;
  - work with weakest liberal preconditions;
  - support various verification procedures for systems of ODEs, and
  - are correct by construction.

- Yet, simpler solutions suffice for program verification:
  - Hoare logic is enough for verification condition generation, and
  - Morgan's refinement calculus suffices for program construction.

Do Hoare logic and refinement calculus suffice for hybrid program verification?

Main Contributions

Development of minimal proof systems for verification of hybrid systems:

1. rules of differential Hoare logic $\mathsf{d}\mathcal{H}$ based on KAT;
2. laws of differential refinement calculus $\mathsf{d}\mathcal{R}$ based on rKAT;
3. integration of lenses as the store model;
4. support invariant reasoning in the style of differential dynamic logic $\mathsf{d}\mathcal{L}$, and
5. tactics for automated verification condition generation.

https://github.com/yonoteam/CPSVerification

Kleene Algebras with Tests

Kleene Algebra

sequential composition

nondeterministic choice　　　　finite iteration

$$(K, +, ;, 0, 1, ^{*})$$

actions of programs　　　abort　　　skip

Tests

- $(B, +, ;, 0, 1, \neg)$ is a boolean algebra,
- use $\alpha, \beta \in K$ and $p, q \in B$ where $B \subseteq K$,
- $\textbf{if } p \textbf{ then } \alpha \textbf{ else } \beta = p \,;\, \alpha + \neg p \,;\, \beta$,
- $\textbf{while } p \textbf{ do } \alpha = (p \,;\, \alpha)^{*} \,;\, \neg p$, and
- $\{p\}\, \alpha\, \{q\} \leftrightarrow p \,;\, \alpha \leq \beta \,;\, q$

State Transformer Model

Programs are functions $S \to \mathcal{P}\, S$:

$$
\begin{aligned}
(\alpha + \beta)\, s &= \alpha\, s \cup \beta\, s, \\
(\alpha\, ;\, \beta)\, s = (\alpha \circ_K \beta)\, s &= \bigcup \{\beta\, s' \mid s' \in \alpha\, s\}, \\
0\, s &= \emptyset, \\
1\, s &= \{s\}, \\
(\neg p)\, s &= \begin{cases} \{s\}, & \text{if } p\, s = \emptyset, \\ \emptyset, & \text{otherwise}, \end{cases} \\
\alpha^*\, s &= \bigcup_{n \geq 0} \alpha^n\, s,
\end{aligned}
$$

where $\alpha^0\, s = 1\, s$ and $\alpha^{n+1} = \alpha^n \circ_K \alpha$.

$$\{p\}\, \alpha\, \{q\} \leftrightarrow (\forall s_1.\ p\, s_1 \to (\forall s_2.\ s_2 \in \alpha\, s_1 \to q\, s))$$

What about Assignments?

Lenses

- Variables are lenses $x = (A, S, get_x, put_x)$ where

$$get_x : S \to A \text{ and } put_x : S \to A \to S$$

- A is a variable type while S is the source
- They satisfy the axioms

$$\begin{aligned} get_x\ (put_x\ s\ v) &= v \\ put_x\ (put_x\ s\ u)\ v &= put_x\ s\ v, \\ put_x\ s\ (get_x\ s) &= s. \end{aligned}$$

Semantics $S \to \mathcal{P}\, S$ for assignments is

$$(x := e)\, s = \{put_x\, s\, (e\, s)\}$$

Verification Rules

- Traditional Hoare logic:

$$
\begin{aligned}
p_1 \le p_2 \wedge \{p_2\}\,\alpha\,\{q_2\} \wedge q_2 \le q_1 &\rightarrow \{p_1\}\,\alpha\,\{q_1\}\\
\{p\}\,\alpha\,\{r\} \wedge \{r\}\,\beta\,\{q\} &\rightarrow \{p\}\,\alpha\,;\beta\,\{q\},\\
\{r\,;p\}\,\alpha\,\{q\} \wedge \{\neg r\,;p\}\,\beta\,\{q\} &\rightarrow \{p\}\,\mathbf{if}\ r\ \mathbf{then}\ \alpha\ \mathbf{else}\ \beta\,\{q\},\\
\{r\,;p\}\,\alpha\,\{p\} &\rightarrow \{p\}\,\mathbf{while}\ r\ \mathbf{do}\ \alpha\,\{\neg r\,;p\},\\
&\quad\ \{\lambda s.\ q\,(put_x\,s\,(e\,s))\}\,x := e\,\{q\}.
\end{aligned}
$$

- Adapted to regular programs

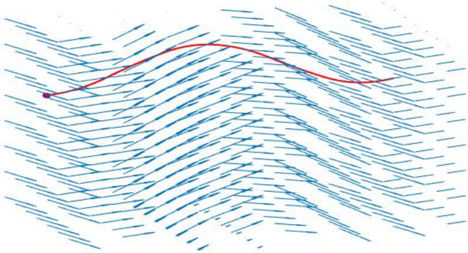
$$
\begin{aligned}
&\{p\}\,\mathbf{skip}\,\{p\},\\
&\{p\}\,\mathbf{abort}\,\{q\},\\
\{p\}\,\alpha\,\{q\} \wedge \{p\}\,\beta\,\{q\} \rightarrow\ &\{p\}\,\alpha+\beta\,\{q\},\\
\{p\}\,\alpha\,\{p\} \rightarrow\ &\{p\}\,\mathbf{loop}\ \alpha\,\{p\},
\end{aligned}
$$

where **loop** $\alpha = \alpha^*$, **skip** $= 1$, and **abort** $= 0$.

What about ODEs?

Vector Field

$$X' \ t = f \ t \, (X \ t)$$



where

$$X : T \subseteq \mathbb{R} \to S \qquad f : T \to S \to S \qquad X \, 0 = s$$

$$\text{orbit} : s \mapsto \{X \, t \mid t \in T\}$$

Semantics for ODEs

- Solutions to initial value problems (IVPs)

$$\mathrm{Sols}\, f\ T\ s = \{X : T \to S \mid (\forall t \in T.\ X'\, t = f\ t\, (X\ t) \wedge X\, 0 = s\}$$

- Guarded orbit

$$\mathrm{orbit}_G^X\ s = \{X\ t \mid t \in T \wedge (\forall \tau \in [0, t].\ G\,(X\ \tau))\}$$

- Semantics $S \to \mathcal{P}\, S$ for assignments are

$$(x' = f\ \&\ G)\, s = \bigcup \{\mathrm{orbit}_G^X\ s \mid X \in \mathrm{Sols}\, f\ T\ s\}$$

- The corresponding rule of inference is

$$\{\lambda\, s.\ \forall t \in T.\ (\forall \tau \in [0, t].\ G\,(X\ t)) \to Q\,(X\ t)\}\ (x'{=}f\ \&\ G)\ \{Q\}$$

easy to obtain if there is a unique solution $X : T \to S$ to the IVPs associated to each s and the vector field f

Invariants in d$\mathcal{H}$

- I is an invariant for f iff $\{I\}\, x'{=}f \,\&\, G\, \{I\}$, or equivalently

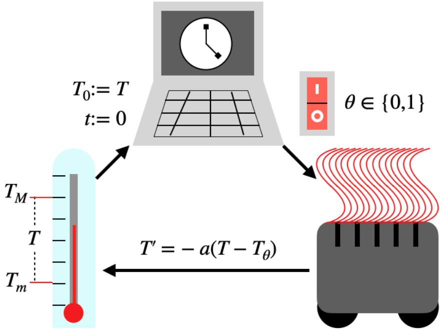
$$\bigcup(\mathcal{P}\,(x'{=}f \,\&\, G)\, I) \subseteq I,$$

- We obtain the following rules

$$
\begin{aligned}
p \le i \wedge \{i\}\,\alpha\,\{i\} \wedge i \le q \ &\rightarrow \{p\}\,\alpha\ \textbf{inv}\ i\,\{q\},\\
\{i\}\,\alpha\,\{i\} \wedge \{j\}\,\alpha\,\{j\} &\rightarrow \{i\,;j\}\,\alpha\,\{i\,;j\},\\
\{i\}\,\alpha\,\{i\} \wedge \{j\}\,\alpha\,\{j\} &\rightarrow \{i+j\}\,\alpha\,\{i+j\},\\
p \le i \wedge \{i\,;t\}\,\alpha\,\{i\} \wedge \neg r\,; i \le q \ &\rightarrow \{p\}\,\textbf{while}\ r\ \textbf{do}\ \alpha\ \textbf{inv}\ i\,\{q\},\\
p \le i \wedge \{i\}\,\alpha\,\{i\} \wedge i \le q \ &\rightarrow \{p\}\,\textbf{loop}\ \alpha\ \textbf{inv}\ i\,\{q\},\\
p \le i \wedge i \text{ is inv. for } f \wedge (G\,; i) \le q \ &\rightarrow \{p\}\,x' = f \,\&\, G\ \textbf{inv}\ i\,\{q\}.
\end{aligned}
$$

where operationally α **inv** $i = \alpha$.

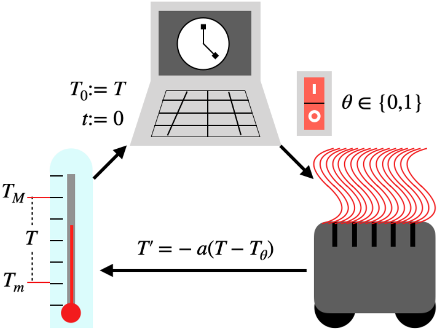
Formalisation of the Thermostat



Lenses $\Pi[n] = (\mathbb{R}, \mathbb{R}^{\{0,1,2,3\}}, \lambda s.\ s\ n, \lambda s\ t.\ s[n \mapsto t])$ give us variables

abbreviation T :: *real* $\Longrightarrow$ *real^4* **where** $T \equiv \Pi[0]$
abbreviation t :: *real* $\Longrightarrow$ *real^4* **where** $t \equiv \Pi[1]$
abbreviation T_0 :: *real* $\Longrightarrow$ *real^4* **where** $T_0 \equiv \Pi[2]$
abbreviation ϑ :: *real* $\Longrightarrow$ *real^4* **where** $\vartheta \equiv \Pi[3]$

Formalisation of the Thermostat



Provide vector field and unique solution

abbreviation $f\ a\ c \equiv [T \mapsto_s -(a * (T - c)),\ T_0 \mapsto_s 0,\ \vartheta \mapsto_s 0,\ t \mapsto_s 1]$

abbreviation $\varphi\ a\ c\ \tau \equiv [T \mapsto_s -exp(-a * \tau) * (c - T) + c,$
$T_0 \mapsto_s T_0,\ \vartheta \mapsto_s \vartheta,\ t \mapsto_s \tau + t]$

Verification of the Thermostat

abbreviation $G\ T_m\ T_M\ a\ L \equiv$
$\mathbf{U}(t \le -\ (ln\ ((L{-}(if\ L{=}0\ then\ T_m\ else\ T_M))/(L{-}T_0)))/a)$

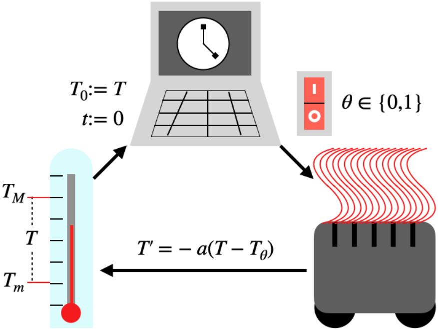
abbreviation $I\ T_m\ T_M \equiv \mathbf{U}(T_m \le T \wedge T \le T_M \wedge (\vartheta = 0 \vee \vartheta = 1))$

abbreviation $ctrl\ T_m\ T_M \equiv$
$(t ::= 0); (T_0 ::= T);$
$(IF\ (\vartheta = 0 \wedge T_0 \le T_m + 1)\ THEN\ (\vartheta ::= 1)\ ELSE$
$IF\ (\vartheta = 1 \wedge T_0 \ge T_h - 1)\ THEN\ (\vartheta ::= 0)\ ELSE\ skip)$

abbreviation $dyn\ T_m\ T_M\ a\ T_u\ \tau \equiv$
$IF\ (\vartheta = 0)\ THEN\ x\acute{} = f\ a\ 0\ \&\ G\ T_m\ T_M\ a\ 0\ on\ \{0..\tau\}\ UNIV\ @\ 0$
$ELSE\ x\acute{} = f\ a\ T_u\ \&\ G\ T_m\ T_M\ a\ T_u\ on\ \{0..\tau\}\ UNIV\ @\ 0$

abbreviation $therm\ T_m\ T_M\ a\ L\ \tau \equiv$
$LOOP\ (ctrl\ T_m\ T_M\ ;\ dyn\ T_m\ T_M\ a\ L\ \tau)\ INV\ (I\ T_m\ T_M)$

Verification of the Thermostat



lemma *thermostat-flow*:
  assumes $0 < a$ **and** $0 \leq \tau$ **and** $0 < T_m$ **and** $T_M < T_u$
  shows $\{I\ T_m\ T_M\}$ *therm* $T_m\ T_M\ a\ T_u\ \tau\ \{I\ T_m\ T_M\}$
  apply (*hyb-hoare* **U**$(I\ T_m\ T_M \wedge t{=}0 \wedge T_0 = T)$)
  prefer *4* **prefer** *8* **using** *local-flow-therm assms* **apply** *force*+
  using *assms therm-dyn-up therm-dyn-down* **by** *rel-auto'*

Differential Refinement Calculus $\mathsf{d}\mathcal{R}$

Extend KAT with refinement operation $[-,-] : B \times B \to K$ such that

$$\{p\}\,\alpha\,\{q\} \leftrightarrow \alpha \leq [p,q].$$

Obtain traditional Morgan Style Refinement laws:

$$\begin{aligned}
\textbf{skip} &\leq [p,p], \\
\textbf{abort} &\leq [p,q], \\
[p',q'] &\leq [p,q], && \text{if } p \leq p' \text{ and } q' \leq q, \\
[p,r]\,;[r,q] &\leq [p,q], \\
[p,q] + [p,q] &\leq [p,q], \\
\textbf{if } t \textbf{ then } [t\,;p,q] \textbf{ else } [\neg t\,;p,q] &\leq [p,q], \\
\textbf{while } t \textbf{ do } [t\,;p,p] &\leq [p,\neg t\,;p], \\
\textbf{loop } [p,p] &\leq [p,p].
\end{aligned}$$

More Refinement Laws

- Laws for assignments $(x := e) \leq [\lambda s.\ Q\,(put_x\, s\,(e\, s)), Q]$.
- Laws for evolution commands where $X'\,t = f\,(X\,t)$ and $X\,0 = s$

$$(x' = f\ \&\ G) \leq [\lambda s \in S. \forall t \in T.\ (\forall \tau \in [0, t].\ G\,(X\,\tau)) \rightarrow Q\,(X\,t), Q].$$

- Monotonoic laws and laws with invariants

$$\begin{array}{rl}
\textbf{if}\ t\ \textbf{then}\ \alpha_1\ \textbf{else}\ \beta_1 \leq \textbf{if}\ t\ \textbf{then}\ \alpha_2\ \textbf{else}\ \beta_2, & \text{if } \alpha_1 \leq \alpha_2 \text{ and } \beta_1 \leq \beta_2; \\
\textbf{while}\ t\ \textbf{do}\ \alpha_1 \leq \textbf{while}\ t\ \textbf{do}\ \alpha_2, & \text{if } \alpha_1 \leq \alpha_2; \\
\textbf{loop}\ \alpha_1 \leq \textbf{loop}\ \alpha_2, & \text{if } \alpha_1 \leq \alpha_2; \\
\textbf{while}\ t\ \textbf{do}\ \alpha\ \textbf{inv}\ i \leq [p, q] & \text{if } p \leq i\ ;\ t \text{ and } \alpha \leq [i, i] \text{ and } \neg t\ ;\ i \leq q; \\
\textbf{loop}\ \alpha\ \textbf{inv}\ i \leq [p, q] & \text{if } p \leq i \text{ and } \alpha \leq [i, i] \text{ and } i \leq q.
\end{array}$$

Refinement of the Thermostat

abbreviation $dyn\ T_m\ T_M\ a\ T_u\ \tau \equiv$
$IF\ (\vartheta = 0)\ THEN\ x' = f\ a\ 0\ \&\ G\ T_m\ T_M\ a\ 0\ on\ \{0..\tau\}\ UNIV\ @\ 0$
$ELSE\ x' = f\ a\ T_u\ \&\ G\ T_m\ T_M\ a\ T_u\ on\ \{0..\tau\}\ UNIV\ @\ 0$

lemma *R-therm-down*:
assumes $a > 0$ **and** $0 \leq \tau$ **and** $0 < T_m$ **and** $T_M < T_u$
shows $[\vartheta = 0 \wedge I\ T_m\ T_M \wedge t = 0 \wedge T_0 = T,\ I\ T_m\ T_M] \geq$
$(x' = f\ a\ 0\ \&\ G\ T_m\ T_M\ a\ 0\ on\ \{0..\tau\}\ UNIV\ @\ 0)$
apply(*rule local-flow.R-g-ode-ivl*[*OF local-flow-therm*])
using *therm-dyn-down*[*OF assms*(1,3), *of* - T_M] *assms* **by** *rel-auto'*

lemma *R-therm-up*:
assumes $a > 0$ **and** $0 \leq \tau$ **and** $0 < T_m$ **and** $T_M < T_u$
shows $[\neg\ \vartheta = 0 \wedge I\ T_m\ T_M \wedge t = 0 \wedge T_0 = T,\ I\ T_m\ T_M] \geq$
$(x' = f\ a\ T_u\ \&\ G\ T_m\ T_M\ a\ T_u\ on\ \{0..\tau\}\ UNIV\ @\ 0)$
apply(*rule local-flow.R-g-ode-ivl*[*OF local-flow-therm*])
using *therm-dyn-up*[*OF assms*(1) - - *assms*(4), *of* T_m] *assms* **by** *rel-auto'*

Refinement of the Thermostat

abbreviation $\mathit{ctrl}\ T_m\ T_M \equiv$
$(t ::= 0); (T_0 ::= T);$
$(\mathit{IF}\ (\vartheta = 0 \wedge T_0 \leq T_m + 1)\ \mathit{THEN}\ (\vartheta ::= 1)\ \mathit{ELSE}$
$\mathit{IF}\ (\vartheta = 1 \wedge T_0 \geq T_h - 1)\ \mathit{THEN}\ (\vartheta ::= 0)\ \mathit{ELSE\ skip})$

lemma *R-therm-time*: $[I\ T_m\ T_M, I\ T_m\ T_M \wedge t = 0] \geq (t ::= 0)$
by (*rule R-assign-law*, *pred-simp*)

lemma *R-therm-temp*:
$[I\ T_m\ T_M \wedge t = 0, I\ T_m\ T_M \wedge t = 0 \wedge T_0 = T] \geq (T_0 ::= T)$
by (*rule R-assign-law*, *pred-simp*)

lemma *R-thermostat-flow*:
assumes $a > 0$ **and** $0 \leq \tau$ **and** $0 < T_m$ **and** $T_M < T_u$
shows $[I\ T_m\ T_M, I\ T_m\ T_M] \geq \mathit{therm}\ T_m\ T_M\ a\ T_u\ \tau$
by (*refinement*;(*rule R-therm-time*)?,(*rule R-therm-temp*)?,
(*rule R-assign-law*)?, (*rule R-therm-up*[*OF assms*])?,
(*rule R-therm-down*[*OF assms*])?) *rel-auto'*

Conclusions

- Used modular semantic framework in Isabelle/HOL to
  - derive a minimal logic $\mathrm{d}\mathcal{H}$ for verification of hybrid programs,
  - obtain refinement components via the laws of $\mathrm{d}\mathcal{R}$,
- Lenses provide
  - a more algebraic program store,
  - better parsing: nicer syntax,
- Future work:
  - Explore total correctness,
  - Adversarial dynamics like in differential game logic,
  - Code generation for verified executable code,
  - Integrate with a CAS that supplies solutions and invariants, leaving the certification work to Isabelle.

https://github.com/yonoteam/CPSVerification