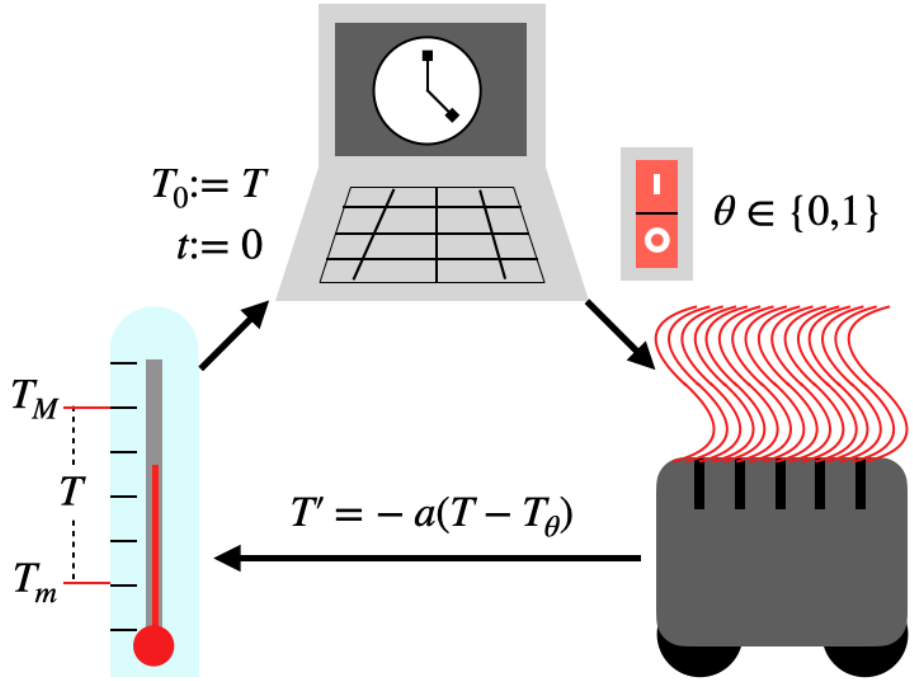
Differential Hoare Logics and Refinement Calculi for Hybrid Systems with Isabelle/HOL

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

University of York, UK

University of Sheffield, UK

Verification of Hybrid Systems



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

Previous Work

- Framework for verification of hybrid programs in a general purpose proof assistant:
  - implemented in Isabelle/HOL
  - benefits from huge, impressive libraries of topology, analysis, ODEs
  - based on MKA
  - works with weakest liberal preconditions
  - supports various verification procedures for systems of ODEs
  - correct by construction

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

Do these calculi suffice for hybrid program verification?

Main Contributions

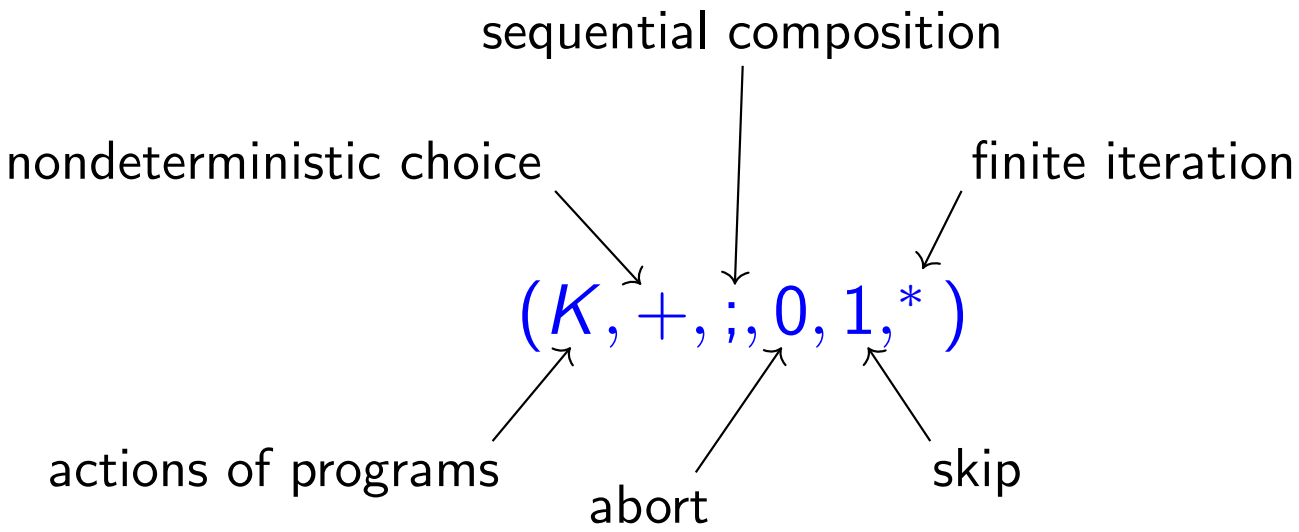
Development of minimalist proof systems for hybrid system verification:

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

`https://github.com/yonoteam/CPSVerification`

Kleene Algebras with Tests

Kleene Algebra



Tests

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

State Transformer Model

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

$$(\alpha + \beta)\, s = \alpha\, s \cup \beta\, s$$

$$(\alpha\,;\beta)\, s = (\alpha \circ_K \beta)\, s = \bigcup \{\beta\, s' \mid s' \in \alpha\, s\}$$

$$0\, s = \emptyset$$

$$1\, s = \{s\}$$

$$\alpha^*\, s = \bigcup_{n \geq 0} \alpha^n\, s \text{ where } \alpha^0\, s = 1\, s \text{ and } \alpha^{n+1} = \alpha^n \circ_K \alpha$$

$$(\neg p)\, s = \begin{cases} \{s\}, & \text{if } p\, s = \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

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

What about Assignments?

Lenses

- Variables are lenses $x = (A, S, get_x, put_x)$ denoted $x : A \Longrightarrow S$ 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

$$get_x\ (put_x\ s\ v) = v$$
$$put_x\ (put_x\ s\ u)\ v = put_x\ s\ v$$
$$put_x\ s\ (get_x\ s) = s$$

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

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

Verification Rules

- Traditional Hoare logic:

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

- Adapted to regular programs

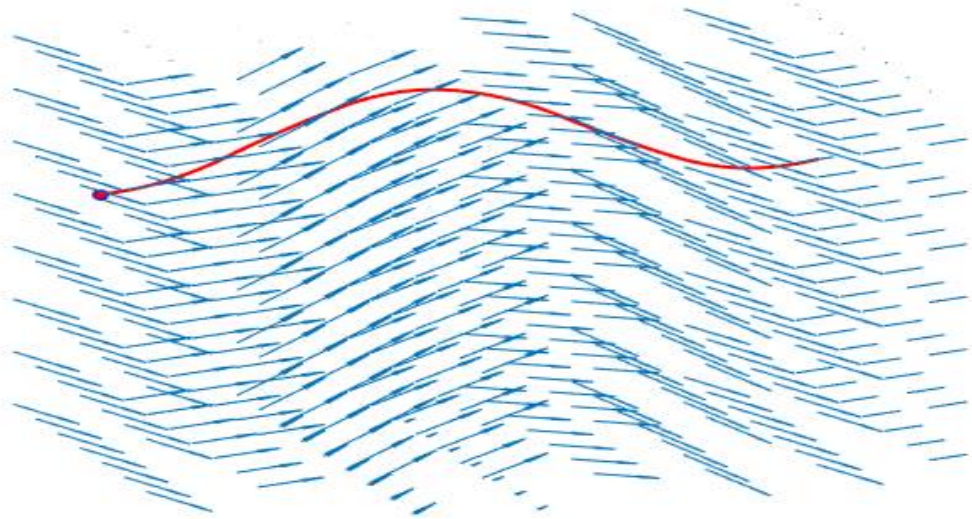
$$
\begin{aligned}
&\to \{p\}\,\textbf{skip}\,\{p\} \\
&\to \{p\}\,\textbf{abort}\,\{q\} \\
\{p\}\,\alpha\,\{q\} \wedge \{p\}\,\beta\,\{q\} &\to \{p\}\,\alpha + \beta\,\{q\} \\
\{p\}\,\alpha\,\{p\} &\to \{p\}\,\textbf{loop } \alpha\,\{p\}
\end{aligned}
$$

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)

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

- Guarded orbit

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

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

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

- 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

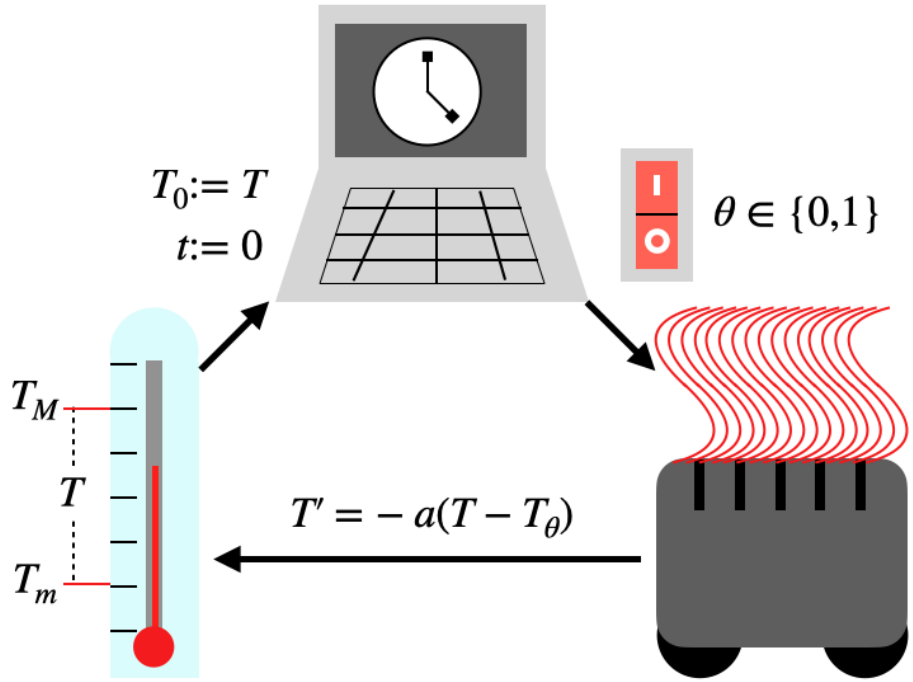
$$\bigcup(\mathcal{P}\,(x'{=}f \,\&\, G)\, I) \subseteq I$$

- We obtain the following rules

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

where operationally $\alpha\ \textbf{inv}\ i = \alpha$

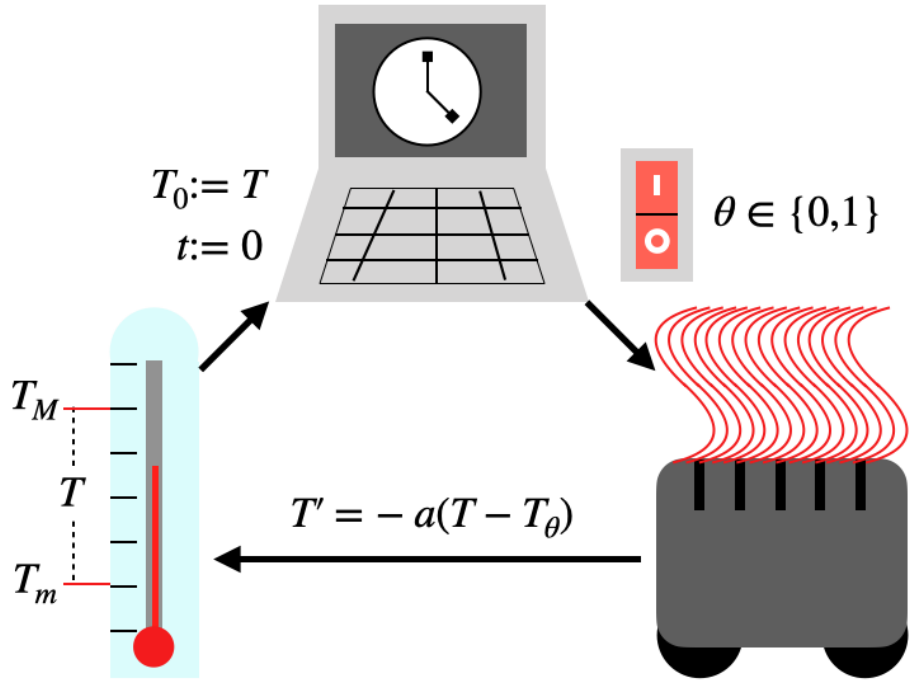
Formalisation of the Thermostat



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 $T :: real \Longrightarrow real\hat{}4$ **where** $T \equiv \Pi[0]$
abbreviation $t :: real \Longrightarrow real\hat{}4$ **where** $t \equiv \Pi[1]$
abbreviation $T_0 :: real \Longrightarrow real\hat{}4$ **where** $T_0 \equiv \Pi[2]$
abbreviation $\vartheta :: real \Longrightarrow real\hat{}4$ **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 $\varphi\ a\ c\ \tau \equiv [T \mapsto_s -\ \mathbf{exp}\ (-a * \tau) * (c - T) + c,$
$T_0 \mapsto_s T_0,\ \vartheta \mapsto_s \vartheta,\ t \mapsto_s \tau + t]$

Verification of the Thermostat

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

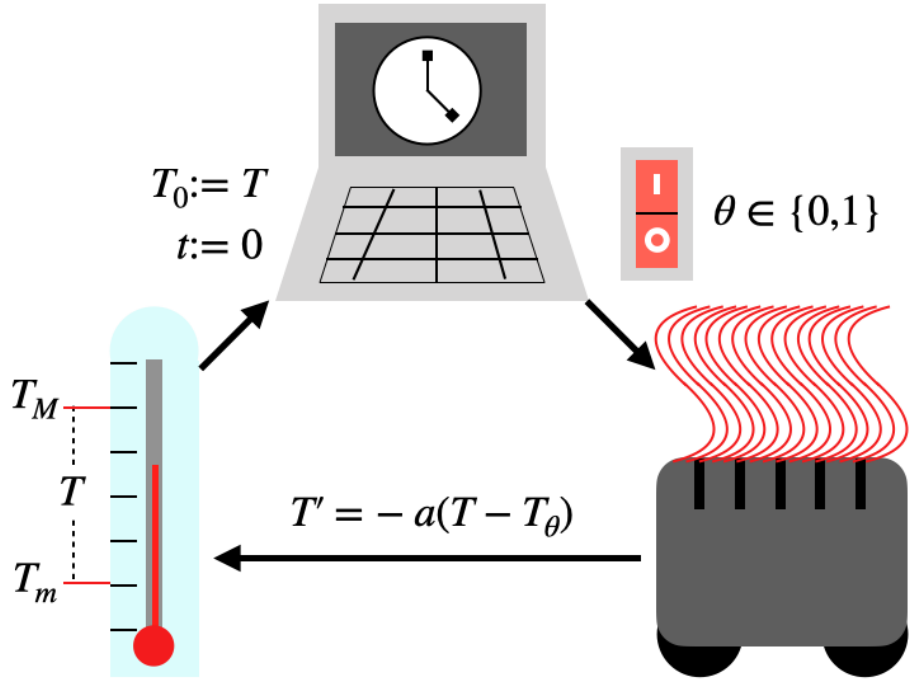
abbreviation $I\ T_m\ T_M \equiv \mathbf{U}(T_m \leq T \wedge T \leq T_M \wedge (\vartheta = 0 \vee \vartheta = 1))$

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

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

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 $\mathrm{d}\mathcal{R}$

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:

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

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

$$\textbf{if } t \textbf{ then } \alpha_1 \textbf{ else } \beta_1 \leq \textbf{if } t \textbf{ then } \alpha_2 \textbf{ else } \beta_2 \quad \text{if } \alpha_1 \leq \alpha_2 \text{ and } \beta_1 \leq \beta_2$$

$$\textbf{while } t \textbf{ do } \alpha_1 \leq \textbf{while } t \textbf{ do } \alpha_2 \quad \text{if } \alpha_1 \leq \alpha_2$$

$$\textbf{loop } \alpha_1 \leq \textbf{loop } \alpha_2 \quad \text{if } \alpha_1 \leq \alpha_2$$

$$\textbf{while } t \textbf{ do } \alpha \textbf{ inv } i \leq [p, q] \quad \text{if } p \leq i\ ;\ t \text{ and } \alpha \leq [i, i] \text{ and } \neg t\ ;\ i \leq q$$

$$\textbf{loop } \alpha \textbf{ inv } i \leq [p, q] \quad \text{if } p \leq i \text{ and } \alpha \leq [i, i] \text{ and } i \leq q$$

Conclusions

- Used modular semantic framework in Isabelle/HOL to
  - derive a minimalist 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 and other extensions of $\mathrm{d}\mathcal{L}$
  - Extension of related libraries of formalised mathematics
  - 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

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

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 $[I\ T_m\ T_M,\ I\ T_m\ T_M] \geq therm\ T_m\ T_M\ a\ T_u\ \tau$
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'*

Verification Rules of $\mathrm{d}\mathcal{H}$

$$\frac{}{\{q\}\,1\,\{q\}} \qquad \frac{}{\{\lambda s.\ q\,(put_x\,s\,(e\,s))\}\,x := e\,\{q\}} \qquad \frac{\{p\}\,\alpha\,\{r\} \quad \{r\}\,\beta\,\{q\}}{\{p\}\,\alpha;\beta\,\{q\}}$$

$$\frac{\{r \wedge p\}\,\alpha\,\{q\} \quad \{\neg r \wedge p\}\,\beta\,\{q\}}{\{p\}\,\textbf{if}\ r\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\,\{q\}} \qquad \frac{\{p \wedge r\}\,\alpha\,\{q\}}{\{p\}\,\textbf{while}\ r\ \textbf{do}\ \alpha\,\{\neg r \wedge q\}}$$

$$\frac{p \to r_1 \quad \{r_1\}\,\alpha\,\{r_2\} \quad r_2 \to q}{\{p\}\,\alpha\,\{q\}}$$

$$\frac{\forall s \in S.\exists! X : T \to S.\ X\,0 = s \wedge (\forall t \in T.\ X'\,t = f\,t\,(X\,t))}{\{\lambda s.\ s \in S \to \forall t \in T.\ (\forall \tau \in [0,t].\ G\,(X\,t)) \to q\,(X\,t)\}\,x' = f\ \&\ G\,\{q\}}$$

Extended verification Rules of $\mathrm{d}\mathcal{H}$

$$\frac{}{\{q\}\,1\,\{q\}} \qquad \frac{}{\{\lambda s.\ q\,(put_x\,s\,(e\,s))\}\,x := e\,\{q\}} \qquad \frac{}{\{p\}\,0\,\{q\}}$$

$$\frac{\{p\}\,\alpha\,\{q\} \quad \{p\}\,\beta\,\{q\}}{\{p\}\,\alpha + \beta\,\{q\}} \qquad \frac{\{p\}\,\alpha\,\{r\} \quad \{r\}\,\beta\,\{q\}}{\{p\}\,\alpha;\beta\,\{q\}}$$

$$\frac{p \to r_1 \quad \{r_1\}\,\alpha\,\{r_2\} \quad r_2 \to q}{\{p\}\,\alpha\,\{q\}} \qquad \frac{\{p\}\,\alpha\,\{p\}}{\{p\}\,\alpha^*\,\{p\}}$$

$$\frac{\forall s \in S.\exists! X : T \to S.\ X\,0 = s \wedge (\forall t \in T.\ X'\,t = f\,t\,(X\,t))}{\{\lambda s.\ s \in S \to \forall t \in T.\ (\forall \tau \in [0,t].\ G\,(X\,t)) \to q\,(X\,t)\}\,x' = f\ \&\ G\,\{q\}}$$