

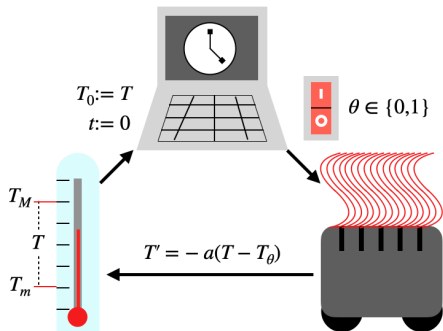
Differential Hoare Logics and Refinement Calculi for Hybrid Systems with Isabelle/HOL

Simon Foster¹ Jonathan Julián Huerta y Munive² Georg Struth²

University of York, UK

University of Sheffield, UK

Verification of Hybrid Systems



dynamics = $T' = -a(T - T_\theta)$

pre = $T_m \leq T \leq T_M$

pos = $T_m \leq T \leq T_M$

control = $t := 0 ; T_0 := T ; \dots$

therm = (control ; dynamics)*

pre \leq [therm] pos

hybrid program

correctness spec

Previous Work

- Modular and extensible semantic 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

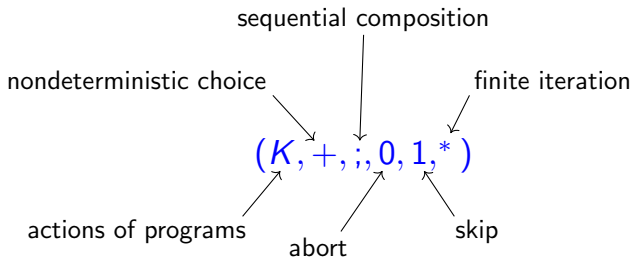
Development of minimalist proof systems for hybrid system verification:

1. differential Hoare logic $d\mathcal{H}$ based on KAT
2. differential refinement calculus $d\mathcal{R}$ based on $rKAT$
3. integration of lenses as the store model
4. invariant reasoning in the style of differential dynamic logic $d\mathcal{L}$
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 $S \rightarrow \mathcal{P} S$:

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

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

$$0 s = \emptyset$$

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

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

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

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

What about Assignments?

Lenses

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

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

- 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 $S \rightarrow \mathcal{P} S$ for assignments is

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

Verification Rules

- Traditional Hoare logic:

$$p_1 \leq p_2 \wedge \{p_2\} \alpha \{q_2\} \wedge q_2 \leq q_1 \rightarrow \{p_1\} \alpha \{q_1\}$$

$$\{p\} \alpha \{r\} \wedge \{r\} \beta \{q\} \rightarrow \{p\} \alpha ; \beta \{q\}$$

$$\{r; p\} \alpha \{q\} \wedge \{\neg r; p\} \beta \{q\} \rightarrow \{p\} \text{ if } r \text{ then } \alpha \text{ else } \beta \{q\}$$

$$\{r; p\} \alpha \{p\} \rightarrow \{p\} \text{ while } r \text{ do } \alpha \{ \neg r; p \}$$

$$\{ \lambda s. q(\text{put}_x s(e)) \} x := e \{q\}$$

- Extended with $d\mathcal{L}$'s hybrid programs

$$\{p\} \text{ skip } \{p\}$$

$$\{p\} \text{ abort } \{q\}$$

$$\{p\} \alpha \{q\} \wedge \{p\} \beta \{q\} \rightarrow \{p\} \alpha + \beta \{q\}$$

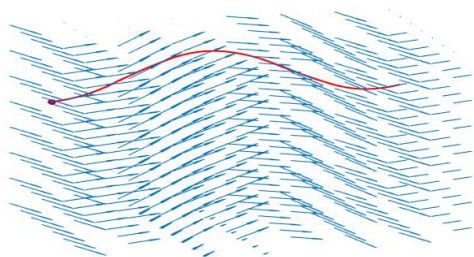
$$\{p\} \alpha \{p\} \rightarrow \{p\} \text{ loop } \alpha \{p\}$$

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

What about ODEs?

Vector Field

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



where

$$X : T \subseteq \mathbb{R} \rightarrow S \quad f : T \rightarrow S \rightarrow S \quad X(0) = s$$

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

Semantics for ODEs

- Solutions to initial value problems (IVPs)

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

- Guarded orbit

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

- Semantics $S \rightarrow \mathcal{P} S$ for evolution commands are

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

- The corresponding rule of inference is

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

easy to obtain if there is a unique solution $X : T \rightarrow S$ to the IVPs associated to each s and the vector field f (Picard-Lindelöf's Theorem)

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

$$p \leq i \wedge \{i\} \alpha \{i\} \wedge i \leq q \rightarrow \{p\} \alpha \mathbf{inv} \ i \{q\}$$

$$\{i\} \alpha \{i\} \wedge \{j\} \alpha \{j\} \rightarrow \{i; j\} \alpha \{i; j\}$$

$$\{i\} \alpha \{i\} \wedge \{j\} \alpha \{j\} \rightarrow \{i+j\} \alpha \{i+j\}$$

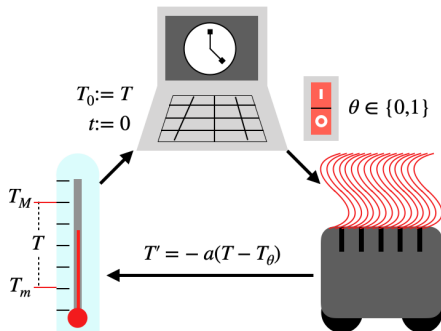
$$p \leq i \wedge \{i; t\} \alpha \{i\} \wedge \neg r; i \leq q \rightarrow \{p\} \mathbf{while} \ r \ \mathbf{do} \ \alpha \ \mathbf{inv} \ i \{q\}$$

$$p \leq i \wedge \{i\} \alpha \{i\} \wedge i \leq q \rightarrow \{p\} \mathbf{loop} \ \alpha \ \mathbf{inv} \ i \{q\}$$

$$p \leq i \wedge i \text{ is inv. for } f \wedge (G; i) \leq q \rightarrow \{p\} x' = f \& G \ \mathbf{inv} \ i \{q\}$$

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

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

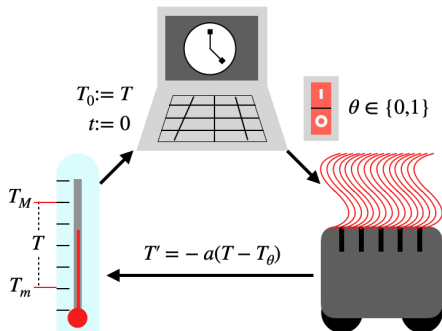
abbreviation $T :: \text{real} \Rightarrow \text{real}^4$ where $T \equiv \Pi[0]$

abbreviation $t :: \text{real} \Rightarrow \text{real}^4$ where $t \equiv \Pi[1]$

abbreviation $T_0 :: \text{real} \Rightarrow \text{real}^4$ where $T_0 \equiv \Pi[2]$

abbreviation $\vartheta :: \text{real} \Rightarrow \text{real}^4$ 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 $\varphi a c \tau \equiv [T \mapsto_s - \mathbf{exp} (-a * \tau) * (c - T) + c,$
 $T_0 \mapsto_s T_0, \vartheta \mapsto_s \vartheta, t \mapsto_s \tau + t]$

Verification of the Thermostat

abbreviation $G T_m T_M a L \equiv$

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

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 $ctrl T_m T_M \equiv$

$(t ::= 0); (T_0 ::= T);$

$(\text{IF } (\vartheta = 0 \wedge T_0 \leq T_m + 1) \text{ THEN } (\vartheta ::= 1) \text{ ELSE}$

$\text{IF } (\vartheta = 1 \wedge T_0 \geq T_h - 1) \text{ THEN } (\vartheta ::= 0) \text{ ELSE skip})$

abbreviation $dyn T_m T_M a T_u \tau \equiv$

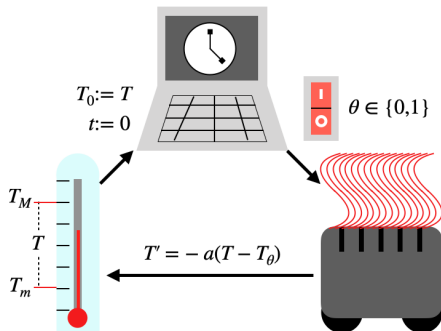
$\text{IF } (\vartheta = 0) \text{ THEN } x' = f a 0 \ \& \ G T_m T_M a 0 \ \text{on } \{0..\tau\} \ \text{UNIV } @ \ 0$

$\text{ELSE } x' = f a T_u \ \& \ G T_m T_M a T_u \ \text{on } \{0..\tau\} \ \text{UNIV } @ \ 0$

abbreviation $therm T_m T_M a L \tau \equiv$

$\text{LOOP } (ctrl T_m T_M ; dyn T_m T_M a L \tau) \ \text{INV } (I T_m T_M)$

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* $\mathbf{U}(I T_m T_M \wedge t=0 \wedge T_0 = T)$)

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 $d\mathcal{R}$

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

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

Obtain traditional Morgan Style Refinement laws:

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

More Refinement Laws

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

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

- Monotonic laws and laws with invariants

if t **then** α_1 **else** $\beta_1 \leq$ **if** t **then** α_2 **else** β_2 if $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$

while t **do** $\alpha_1 \leq$ **while** t **do** α_2 if $\alpha_1 \leq \alpha_2$

loop $\alpha_1 \leq$ **loop** α_2 if $\alpha_1 \leq \alpha_2$

while t **do** α **inv** $i \leq [p, q]$ if $p \leq i$; t and $\alpha \leq [i, i]$ and $\neg t$; $i \leq q$

loop α **inv** $i \leq [p, q]$ if $p \leq i$ and $\alpha \leq [i, i]$ and $i \leq q$

Conclusions

- Used modular semantic framework in Isabelle/HOL to
 - derive a minimalist logic $d\mathcal{H}$ for verification of hybrid programs
 - obtain refinement components via the laws of $d\mathcal{R}$
- Lenses provide
 - a more algebraic program store
 - better parsing: nicer syntax
- Future work:
 - Explore total correctness
 - Adversarial dynamics and other extensions of $d\mathcal{L}$
 - 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 $\text{dyn } T_m \ T_M \ a \ T_u \ \tau \equiv$

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

lemma *R-therm-down*:

assumes $a > 0$ and $0 \leq \tau$ and $0 < T_m$ and $T_M < T_u$

shows $[\vartheta = 0 \wedge I \ T_m \ T_M \ \wedge \ t = 0 \wedge T_0 = T, I \ T_m \ T_M] \geq$

$(x' = f \ a \ 0 \ \& \ G \ T_m \ T_M \ a \ 0 \ \text{on } \{0..\tau\} \ UNIV \ @ \ 0)$

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 $[\neg \vartheta = 0 \wedge I \ T_m \ T_M \ \wedge \ t = 0 \wedge T_0 = T, I \ T_m \ T_M] \geq$

$(x' = f \ a \ T_u \ \& \ G \ T_m \ T_M \ a \ T_u \ \text{on } \{0..\tau\} \ UNIV \ @ \ 0)$

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 $d\mathcal{H}$

$$\frac{}{\{q\} 1 \{q\}} \quad \frac{}{\{\lambda s. q(\text{put}_x s(es))\} x := e \{q\}} \quad \frac{\{p\} \alpha \{r\} \quad \{r\} \beta \{q\}}{\{p\} \alpha; \beta \{q\}}$$

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

$$\frac{p \rightarrow r_1 \quad \{r_1\} \alpha \{r_2\} \quad r_2 \rightarrow q}{\{p\} \alpha \{q\}}$$

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

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

$$\overline{\{q\} 1 \{q\}}$$

$$\overline{\{\lambda s. q(\text{put}_x s(es))\} x := e \{q\}}$$

$$\overline{\{p\} 0 \{q\}}$$

$$\frac{\{p\} \alpha \{q\} \quad \{p\} \beta \{q\}}{\{p\} \alpha + \beta \{q\}}$$

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

$$\frac{p \rightarrow r_1 \quad \{r_1\} \alpha \{r_2\} \quad r_2 \rightarrow q}{\{p\} \alpha \{q\}}$$

$$\frac{\{p\} \alpha \{p\}}{\{p\} \alpha^* \{p\}}$$

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