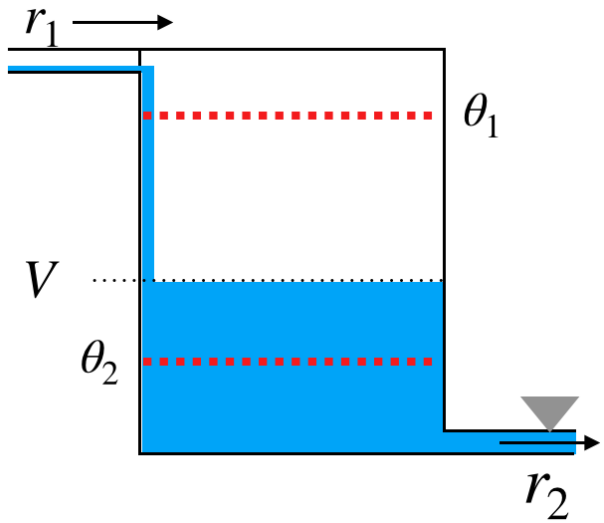
Verifying Hybrid Systems with Modal Kleene Algebra

Jonathan Julián Huerta y Munive Georg Struth

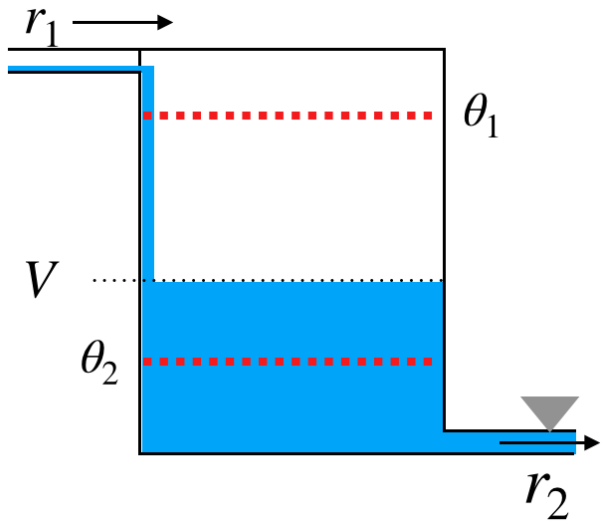
University of Sheffield, UK

Hybrid System Verification



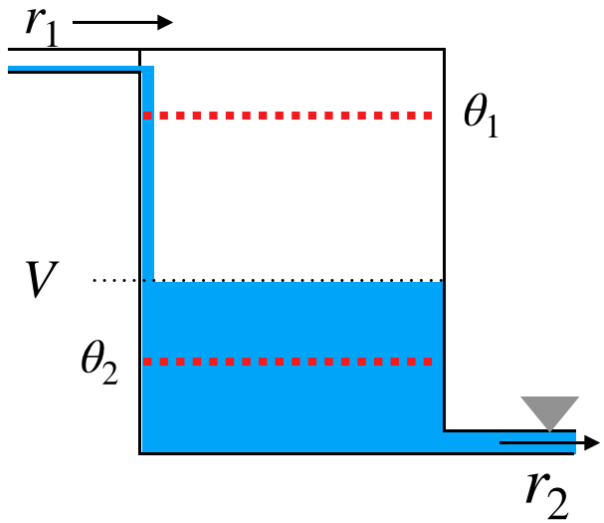
$$
\begin{aligned}
\text{dynamics} &= V' = r_1 - r_2 \quad r_1' = f \quad r_2' = 0 \\
\text{safe} &= \theta_2 \leq V \leq \theta_1 \\
\text{control} &= r_2 := c \\
\text{tank} &= (\text{control}; \text{dynamics})^* \\
\text{safe} &\leq [\text{tank}]\,\text{safe}
\end{aligned}
$$

Hybrid System Verification



$$\begin{aligned}
\text{dynamics} &= V' = r_1 - r_2 \quad r_1' = f \quad r_2' = 0 \\
\text{safe} &= \theta_2 \leq V \leq \theta_1 \\
\text{control} &= r_2 := c \\
\text{tank} &= (\text{control}; \text{dynamics})^* \qquad \text{Hybrid Program} \\
\text{safe} &\leq [\text{tank}]\,\text{safe} \qquad \text{Verification Condition}
\end{aligned}$$

Hybrid System Verification



$$\begin{aligned}
\text{dynamics} &= V' = r_1 - r_2 \quad r_1' = f \quad r_2' = 0 \\
\text{safe} &= \theta_2 \leq V \leq \theta_1 \\
\text{control} &= r_2 := c \\
\text{tank} &= (\text{control}; \text{dynamics})^* \\
\text{safe} &\leq [\text{tank}]\,\text{safe}
\end{aligned}$$

Kleene Algebra

Modal Kleene Algebra

Hybrid System Verification

- Hybrid systems verification requires integrating:
  - Dynamical systems (ODEs,. . .)
  - Program verification formalisms (automata, modal logics, . . .)
- Differential dynamic logic (d$\mathcal{L}$):
  - Extension of dynamic logic to hybrid programs
  - Well supported by the KeYmaera X tool
- Mathematical components in Isabelle proof assistant:
  - Verification components based on modal Kleene algebra (MKA)
  - Impressive theory stack—from topology and measure theory to analysis and ODEs

Can they be integrated?

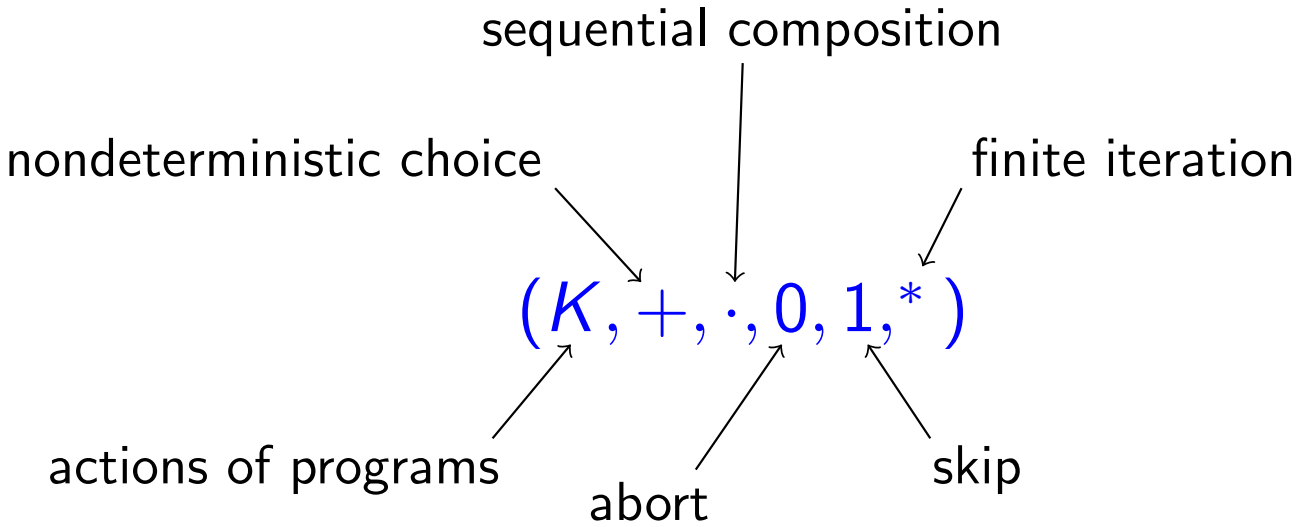
Main Contributions

MKA-based verification components for hybrid systems in Isabelle inspired by d$\mathcal{L}$:

1. Relational flow model of MKA suitable for hybrid systems
2. Predicate transformer semantics for hybrid programs
3. First Isabelle verification component: equational reasoning and wlp
4. Second Isabelle verification component for differential invariants
5. Derivation of domain-specific d$\mathcal{L}$-style inference rules

Components correct by construction

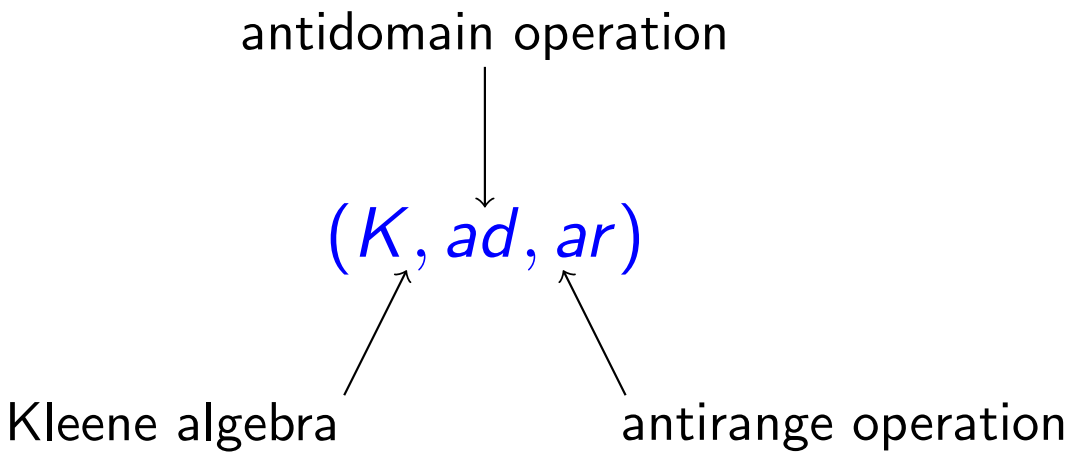
Kleene Algebras



Relational model

- $(\mathcal{P}(S \times S), \cup, ;, \emptyset, Id, ^*)$ forms relation KA over S.

Modal Kleene Algebras



- $ad\, x$ models states from which x can't be executed.

Relational model

- $ad\, R = \{(a, a) \mid \neg\exists b.\ (a, b) \in R\}$

Modal Kleene Algebras

Algebra of propositions

- $(ad[K], +, \cdot, ad, 0, 1)$ forms boolean algebra in K.
- $\bar{p} = ad\, p$ models complementation.
- $p, q \in ad[K]$ yield tests/assertions.

Algebra of programs

- **if** p **then** x **else** $y = px + \bar{p}y$
- **while** p **do** $x = (px)^* \bar{p}$

Algebra of Predicate Transformers

- $wlp\ x\ p = [x]p = ad\,(x\,(ad\,p))$.
- $p \leq [x]q$ partial correctness specification.

Integrating Discrete State Dynamics

Store

- Functions $s : V \to E$ from variables to values.
- Assignments $(v := e) = \{(s, s[v \mapsto e\ s]) \mid s \in E^V\}$.
- wlp law: $[v := e]Q = \lambda s.\ Q\, s[v \mapsto e\ s]$.

Modular Approach

- MKA is polymorphic in Isabelle.
  - Relation MKA is an instance of MKA.
  - Relational store MKA is an instance of relation MKA.
- We can plug in other store models.

What about ODEs?

System of Equations

$$\begin{bmatrix} x_1'\ t \\ x_2'\ t \\ \vdots \\ x_n'\ t \end{bmatrix} = \begin{bmatrix} f_1\ ((x_1\ t), \ldots, (x_n\ t)) \\ f_2\ ((x_1\ t), \ldots, (x_n\ t)) \\ \vdots \\ f_n\ ((x_1\ t), \ldots, (x_n\ t)) \end{bmatrix}$$

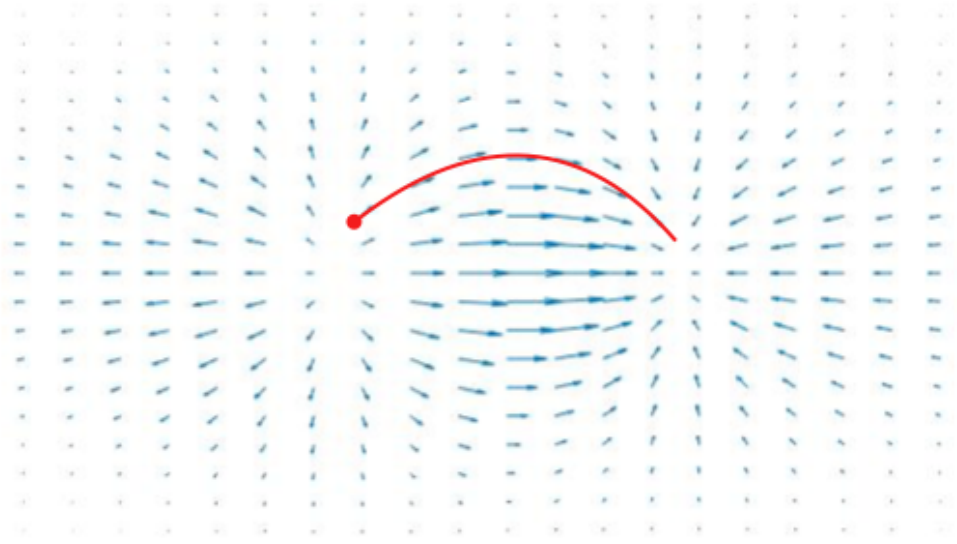
where

$$x_i : \mathbb{R} \to \mathbb{R} \qquad f_i : \mathbb{R}^n \to \mathbb{R}$$

What about ODEs?

Vector Field

$$x'\ t = f\ (x\ t)$$



where

$$x : \mathbb{R} \to \mathbb{R}^n \qquad f : \mathbb{R}^n \to \mathbb{R}^n$$

What about ODEs?

Picard-Lindelöf

- For Lipschitz continuous $f : S \subseteq \mathbb{R}^n \to \mathbb{R}^n$, each initial condition (t_0, s) admits a unique solution φ_s:

$$\varphi_s'\ t = f\ (\varphi_s\ t)$$

- Induces flow $\varphi : T \subseteq \mathbb{R} \to S \to S$ such that $\varphi_s = (\lambda t.\ \varphi\ t\ s)$.

Hybrid Store

- Functions $s : V \to \mathbb{R}$.
- Evolution statement $(x' = f) = \{(s, \varphi\ t\ s) \mid s \in \mathbb{R}^V \wedge t \geq t_0\}$.
- wlp law: $[x' = f]\ Q = \lambda s. \forall t \geq t_0.\ Q\ (\varphi\ t\ s)$.

Provided $\varphi : T \subseteq \mathbb{R} \to S \to S$ is a flow for $f : S \subseteq \mathbb{R}^V \to \mathbb{R}^V$.

First Verification Component

lemma *wp-g-orbit*:
assumes *local-flow f S T L φ*
shows $[(x' = f)\ T\ S\ @\ t0\ \&\ G]\ Q = \lambda s. \forall\, t{\geq}t0.\ (\forall\, r{\in}\{t0..t\}. G\ (\varphi\ r\ s)) {\longrightarrow} Q\ (\varphi\ t\ s))$

...

lemma *wp-assign* [*simp*]: $[v ::= e]\ Q = \lambda s.\ Q\ (s\ (v := e\ s))$

...

lemma *fbox-cond* [*simp*]: $[\mathit{if}\ p\ \mathit{then}\ x\ \mathit{else}\ y\ \mathit{fi}]\ q = (d\ p \cdot [x]\ q) + (\mathit{ad}\ p \cdot [y]\ q)$

...

lemma *fbox-whilei*:
assumes $d\ p \leq d\ i$
and $d\ i \cdot \mathit{ad}\ t \leq d\ q$
and $d\ i \cdot d\ t \leq [x]\ i$
shows $d\ p \leq [\mathit{while}\ t\ \mathit{inv}\ i\ \mathit{do}\ x\ \mathit{od}]\ q$

...

lemma *fbox-seq* [*simp*]: $[x \cdot y]\ q = [x]\ [y]\ q$

Differential Invariants

Differential Invariant

$$I \subseteq [y'{=}f \,\&\, G]\ I$$

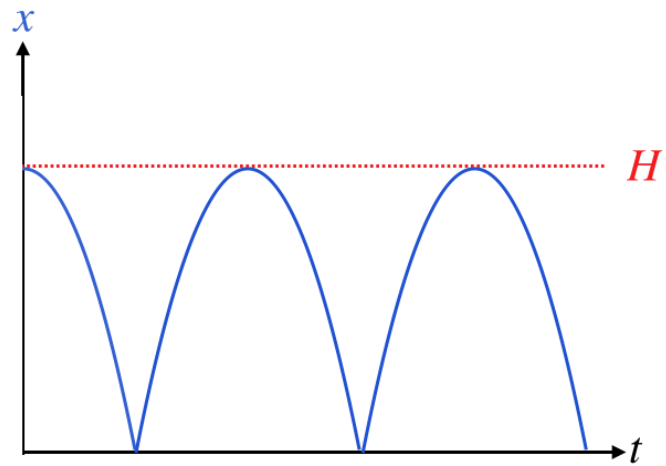
Bouncing Ball

PRE:$0 \leq x \wedge x = H \wedge v = 0 \wedge g > 0$

$[((x' = v,\ v' = -g,\ \text{with } 0 \leq x);$

if $x = 0$ **then** $v := -v$ **else skip**$)^*]$

POST:$0 \leq x \wedge x \leq H$



Proof with invariant can be easier than solving ODE.

Differential Invariants

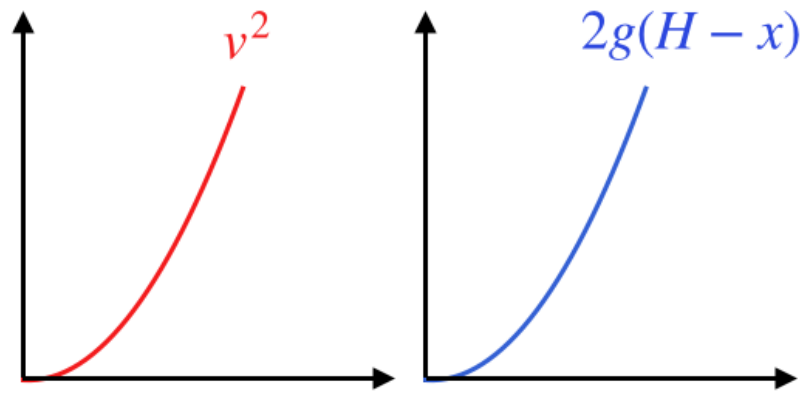
Bouncing Ball Invariant

$$I \;=\; v^2 = 2g(H - x)$$

$$\frac{I \subseteq [x' = v,\ v' = -g]\ (I)'}{I \subseteq [x' = v,\ v' = -g]\ I}$$

Same rates

$$\begin{aligned}(I)' &= (v^2)' = (2g(H-x))' \\ &= 2vv' = -2gx' \\ &= -2vg = -2gv \\ &= \text{true}\end{aligned}$$



Second Verification Component

Formalised in Isabelle:

Induction

$$\frac{\Gamma \vdash I \qquad G \vdash (I')[\theta/x'] \qquad I \vdash Q}{\Gamma \vdash [x'=\theta \,\&\, G]\ Q}\, dI$$

Cut

$$\frac{\Gamma \vdash [x'=\theta \,\&\, G]\ I \qquad \Gamma \vdash [x'=\theta \,\&\, G \wedge I]\ Q}{\Gamma \vdash [x'=\theta \,\&\, G]\ Q}\, dC$$

Weakening

$$\frac{G \vdash Q}{\Gamma \vdash [x' = \theta \,\&\, G]Q}\, dW$$

Bouncing Ball Again

$$\begin{array}{l}\text{PRE:} 0 \leq x \wedge x = H \wedge v = 0 \wedge g > 0 \\ \quad [((x' = v,\ v' = -g,\ \text{with } 0 \leq x); \\ \quad\ \textbf{if } x = 0 \textbf{ then } v := -v \textbf{ else skip})^*] \\ \text{POST:} 0 \leq x \wedge x \leq H\end{array}$$

Bouncing Ball Again

lemma *bouncing-ball*:

$$
\begin{aligned}
&\mathit{PRE}\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' = H \wedge s\ ''v'' = 0 \wedge s\ ''g'' > 0)\\
&((\mathit{ODEsystem}\ [(''x'',\ \lambda\ s.\ s\ ''v''),(''v'',\lambda\ s.\ -\ s\ ''g'')]\ \mathit{with}\ (\lambda\ s.\ 0 \leq s\ ''x''));\\
&(\mathit{IF}\ (\lambda\ s.\ s\ ''x'' = 0)\ \mathit{THEN}\ (''v'' ::= (\lambda\ s.\ -\ s\ ''v''))\ \mathit{ELSE}\ (\mathit{Id})\ \mathit{FI})^*\\
&\mathit{POST}\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' \leq H)
\end{aligned}
$$

Bouncing Ball Again

lemma *bouncing-ball*:
$PRE\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' = H \wedge s\ ''v'' = 0 \wedge s\ ''g'' > 0)$
$((ODEsystem\ [(''x'', \lambda\ s.\ s\ ''v''),(''v'',\lambda\ s.\ -\ s\ ''g'')]\ with\ (\lambda\ s.\ 0 \leq s\ ''x''));$
$(IF\ (\lambda\ s.\ s\ ''x'' = 0)\ THEN\ (''v'' ::= (\lambda\ s.\ -\ s\ ''v''))\ ELSE\ (Id)\ FI))^*$
$POST\ (\lambda\ s.\ 0 \leq s\ ''x'' \wedge s\ ''x'' \leq H)$

Proof Sketch

1. Provide loop-invariant: $0 \leq x \wedge 0 < g \wedge v^2 = 2g(H - x)$.
2. Prove precondition implies loop-invariant (automatic).
3. Prove body of loop maintains loop-invariant (interactive):
   - Introduce differential invariant with dC: $0 < g \wedge v^2 = 2g(H - x)$.
   - Discharge with dI.
   - Guard is postcondition, hence use dW.
4. Prove that loop-invariant implies postcondition (using arithmetic).

Relationship to d$\mathcal{L}$ and KeYmaera X

d$\mathcal{L}$ and KeYmaera X

- Special purpose sequent calculus with complex substitution rules.
- Uses restricted classes of ODE's (real arithmetic).
- Aims at automated industrial verification.

Hybrid MKA

- Equational reasoning and function updates instead of substitutions.
- Open approach limited only by HOL.
- Interactive and experimental.

Conclusions

- First $\mathrm{d}\mathcal{L}$ based hybrid verification components in proof assistant.
- Possible extensions:
  - Hoare-style approach based on KAT.
  - Refinement calculi.
  - Hybrid duration calculus.
- Future work:
  - Better integration of vector spaces and linear algebra.
  - Predicate transformer semantics using orbits and Kleisli arrows.
  - Better tactics and proof automation.
  - Extended case studies.

https://github.com/yonoteam/CPSVerification