



**Carnegie
Mellon
University**

Facilitating Concurrency in Hybrid Programs

DECEMBER 10, 2019

15-824: Logical Foundations of Cyber-Physical Systems (Fall 2019)

Haithem Turki
Long Pham



Languages shape our thinking

Shadenfreude

- Malicious enjoyment of the misfortunes of others. (Oxford English Dictionary)



Overview

- Motivation and Background
- Theory
- Implementation
- Future Work



Motivation and Background

Cyber-physical systems (CPSs) are inherently composite

Interaction between CPSs

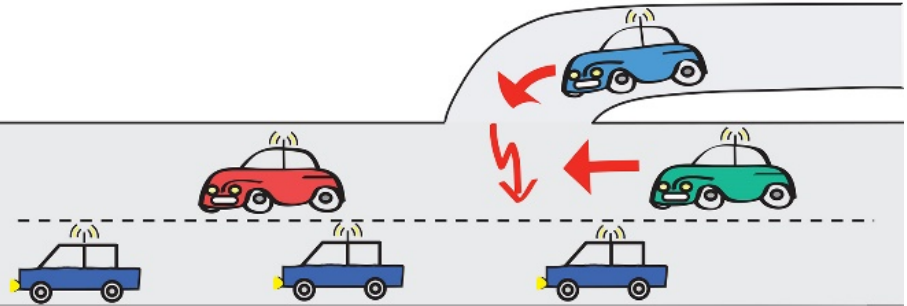


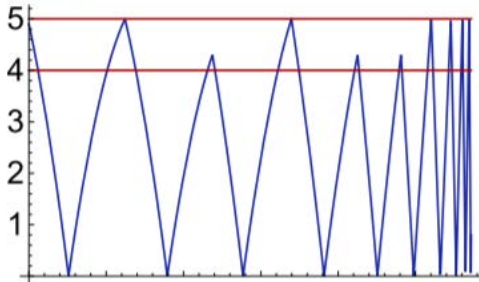
Image courtesy of <http://lfcps.org/course/lfcps19.html>

Interaction within a CPS

```

Proposition (Quantum can play ping-pong safely)
 $0 \leq x \wedge x \leq 5 \wedge v \leq 0 \wedge g > 0 \wedge 1 \geq c \geq 0 \wedge f \geq 0 \rightarrow$ 
 $[[\{x' = v, v' = -g \wedge x \geq 0 \wedge x \leq 5\} \cup \{x' = v, v' = -g \wedge x \geq 5\}];$ 
 $\text{if}(x=0) v := -cv \text{ else if}(4 \leq x \leq 5 \wedge v \geq 0) v := -fv]^*(0 \leq x \leq 5)$ 
    
```

Proof @invariant($0 \leq x \leq 5 \wedge (x = 5 \rightarrow v \leq 0)$)



Just can't implement ...

Image courtesy of <https://lfcps.org/lfcps/slides/09-time-slides.pdf>

Inputs, outputs, and parallel composition



C. A. R. (Tony) Hoare,
Communicating Sequential Processes, 1978

Image courtesy of Microsoft Research



Inputs, outputs, and parallel composition

Consumer = order!coffee -> how-much?x -> pay!x -> get?y -> STOP

Barista = order?y -> how-much!2 -> pay?x -> get!y -> STOP

Consumer || Barista = order!coffee -> how-much!2 -> pay!2 -> get!coffee -> STOP



Inputs, outputs, and parallel composition

Consumer = order!coffee -> how-much?x -> get?y -> pay!x -> STOP

Barista = order?y -> how-much!2 -> pay?x -> get!y -> STOP

Consumer || Barista = order!coffee -> how-much!2 -> STOP



Adding concurrency to hybrid programs

Our Contributions

- Adding channels to hybrid programs
- Providing trace semantics
- Devising a sequentialization algorithm
- Extending KeYmaera X to provide a proof-of-concept implementation supporting our augmented syntax



Theory



Local variables

Consumer = `x:=1; pay!x; x:=0`

Barista = `pay?x; x:=x+1`

Global variables

$$\begin{aligned} \{t' = -1\} \parallel \{t' = -3\} &= \{t' = -1\}; \{t' = -4\}; \{t' = -3\} \\ &\cup \{t' = -1\}; \{t' = -4\}; \{t' = -1\} \\ &\cup \{t' = -3\}; \{t' = -4\}; \{t' = -1\} \\ &\cup \{t' = -3\}; \{t' = -4\}; \{t' = -3\}. \end{aligned}$$

Syntax of concurrent hybrid programs

$\alpha, \beta ::= x := e$	assignment
$?Q$	test
$x' = f(x) \ \& \ Q$	ordinary differential equation; ODE
$\alpha \cup \beta$	nondeterministic choice
$\alpha; \beta$	sequential composition
α^*	nondeterministic finite repetition
$\alpha \parallel \beta$	parallel composition
$c!e$	blocking channel write
$c?x$	blocking channel read

Trace semantics

Definition 3.2 (Action). *An action is one of the following:*

1. *Assignment: $x := e$;*
2. *Test: $?Q$;*
3. *Ordinary differential equation: $(\{x' = f(x) \ \& \ Q\}, r)$, where $r \in \mathbb{R}$ denotes the duration of continuous evolution;*
4. *Input: $c?x$, where $c \in \mathcal{C}$ is a channel name and $x \in \mathcal{V} = \mathcal{V}_\ell \cup \mathcal{V}_g$ is a variable symbol.*
5. *Output: $c!e$, where $c \in \mathcal{C}$ is a channel name and e is a polynomial term.*

Trace semantics

Definition 3.5 (Interleaving of sequences of actions). *Suppose we are given a state $v \in \text{State}(\mathcal{V})$ and two sequences $\sigma = (\sigma_1, \dots, \sigma_n)$ and $\rho = (\rho_1, \dots, \rho_m)$ of actions. The interleaving of σ and ρ with respect to v , denoted by $v \vdash (\sigma \parallel \rho)$, is a set of sequences of actions inductively defined below. For brevity, whenever there is symmetry, we only present one of two dual cases.*

Trace semantics

4. If $\sigma_1 = c_1!e$ and $\rho_1 = c_2?x$, then

$$v \vdash \sigma \parallel \rho = c_1!u \circ (v' \vdash (\sigma_2, \dots, \sigma_n) \parallel (\rho_2, \dots, \rho_n)),$$

where $c_1 \equiv c_2$, $u = v[[e]]$, and v' is the result of applying $[x := u]$ to state v .



Implementation

Current Implementation

Authoring Parallel Programs

- Extended existing KeYmaera X [1] theorem prover
- Augmented KeYmaera X parser and lexer to handle necessary syntax
- Added new *Parallel* program type
- Propagated new *Channel* concept throughout existing KeYmaera X codebase

airConditioner

```
1 Definitions
2   Real Thot;
3   Real Tcold;
4 End.
5
6 ProgramVariables
7   Real t;
8 End.
9
10 Problem
11   (Thot > Tcold)
12   ->
13   <{{{?(t > Thot); {t'=-1}} ++ {?(t < Tcold); {t'=3}} || {{?(t > Thot); {t'=-3}} ++ {?(t < Tcold); {t'=1}}}} & t}>
14   (t <= Thot & t >= Tcold)
15 End.
```

Authoring parallel programs in KeYmaera X web interface

[1] <http://www.ls.cs.cmu.edu/KeYmaeraX/>

Current Implementation

Proofs for Parallel Programs

- Implemented subset of the core and derived axioms needed to prove the safety of parallel programs
- Full set of axioms still to be implemented
- But able to complete proofs of small examples in current implementation

The screenshot shows the KeYMaera X web interface. At the top, there is a menu bar with options: Prop, Explore, Unfold, Simplify, Undo, Edit, Browse, Model, Propositional, and Quantifier. Below the menu bar, a proof is displayed. The proof starts with a goal \vdash and a hypothesis $Thot > Tcold \rightarrow$. The goal is $(t \leq Thot \wedge t \geq Tcold)$. The proof steps are: $\{ ?t > Thot; \{t' = -1\} \cup ?t < Tcold; \{t' = 3\} \}$, followed by a step labeled '1:' with $\{ ?t > Thot; \{t' = -3\} \cup ?t < Tcold; \{t' = 1\} \}$, and finally t . Below the proof, a list of suggested parallel axioms is shown:

$\langle ++ \mid \rangle$ parChoiceLd	$\langle \{a; Ub\} \forall v\{d\} \wedge I \rangle p \leftrightarrow \langle a; \forall v\{d\} \wedge I \rangle p \vee \langle b; \forall v\{d\} \wedge I \rangle p$
$\langle \mid ++ \rangle$ parChoiceRd	$\langle a; \forall v\{b; Ud\} \wedge I \rangle p \leftrightarrow \langle a; \forall v\{b\} \wedge I \rangle p \vee \langle a; \forall v\{d\} \wedge I \rangle p$
chaseAt	chaseAt
$\langle \cdot \rangle d$ diamondd	$(a)P \leftrightarrow \neg[a]\neg P$

At the bottom of the interface, there is a search bar with the text "Search for lemmas".

Parallel axioms suggested in KeYMaera X web interface

Current Implementation

parAssign: Proof ▶ Auto ✎ Prop 🔍 Explore 📄 Unfold ✖ Simplify ↶ Undo ✎ Edit 🗂 Browse 📧 Model Pr

Proof: ✓ All goals closed

Provable(==> x=10&y=2->[{x:=x*2;||{y:=y-1;} & c}]x+y=21 proved)

Tactic to Reproduce the Proof

```
implyR(1) ; parAb(1) ; composeb(1.0) ; composeb(1.1) ; assignb(1.0.1) ; assignb(1.0) ; andR(1) ; <(
  QE,
  assignb(1) ; assignb(1) ; QE
)
```

Completed proof in KeYmaera X web interface

VR

1. $\langle \{ ?t \rangle \text{That}; \{ ! = -1 \} \parallel ?t \rangle \text{That}; \{ ! = -3 \} \& ! \rangle \langle ! \leq \text{That} \wedge ! \geq \text{TCold} \rangle$
2. $\langle \{ ?t \rangle \text{That}; \{ ! = -1 \} \parallel ?t \langle \text{TCold}; \{ ! = 1 \} \& ! \rangle \rangle \langle ! \leq \text{That} \wedge ! \geq \text{TCold} \rangle$
3. $\langle \{ ?t \langle \text{TCold}; \{ ! = 3 \} \parallel ?t \rangle \text{That}; \{ ! = -3 \} \& ! \rangle \rangle \langle ! \leq \text{That} \wedge ! \geq \text{TCold} \rangle$
4. $\langle \{ ?t \langle \text{TCold}; \{ ! = 3 \} \parallel ?t \langle \text{TCold}; \{ ! = 1 \} \& ! \rangle \rangle \rangle \langle ! \leq \text{That} \wedge ! \geq \text{TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle \text{TCold}; \{ ! = 3 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle \text{TCold}; \{ ! = 3 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

VR $\text{That} \triangleright \text{TCold}$ \vdash $\langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \triangleright \text{That}; \{ ! = -3 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle \vee \langle ! \langle ?t \triangleright \text{That}; \{ ! = 1 \text{Atrue} \} \vee \{ ?t \langle \text{TCold}; \{ ! = 1 \text{Atrue} \} \} \wedge ! \rangle \langle ! \text{SThatA2TCold} \rangle$

Proof tree in KeYmaera X web interface



Future Work

- Providing full implementation of our extension in KeYmaera X
- Providing a formal definition of equality based on trace semantics
- Formally proving the sequentialization axioms
- Introduce more constructs to make calculus for concurrent hybrid programs more expressive
 - Example: wake up a stalled program when an ODE of another program reaches a certain state

Thank you!

- Code available on GitHub: <https://github.com/hturki/KeYmaeraX-release/tree/parallel>

```
parTest.Proof ▶ Auto ▶ Prep ▶ M Explore ▶ Unfold ▶ Simplify ▶ Hide ▶ Edit ▶ Show ▶ Model ▶ Propositional - Quantifier - Hybrid Program - Differential Equation - Tools -
Proof: ✓ All goals closed
Provable( $\epsilon \Rightarrow \exists x \exists y \forall z \neg ((\exists x_0 \exists y_0 z_0) \wedge (|y_0 - z_0| < \epsilon \wedge |x_0 - y_0| < \epsilon)) \wedge \exists x \exists y \exists z \text{ proved}$ )
Tactic to Reproduce the Proof
[map] xk(1) ; parChoiceEnd(1) ; parChoiceEnd(1,0) ; orR(1) ; parChoiceEnd(3) ; orR(2) ; orR(1) ; parTest(1) ; parTest(2) ; parTest(3) ; parTest(4) ; orR(1) ; orR(1) ; orR(1) ; orR(1) ; com
posed(0) ; testd(0) ; testd(0,1) ; hideR(1 := "c?y < 1; x?y < 12") ; hideR(1 := "c?y < 1; x?z > x?y - 12") ; hideR(1 := "c?z?y < 3; x?y < 12") ; hideR(1 := "c?
x?y < 1; x?y < 12") ; hideR(1 := "c?y < 1; x?y > x?y - 12") ; hideR(1 := "c?x?y < 3; x?y < 12") ; QE
```

AR (87,1) ✓ QE (89,-1)

QE	x=10Ay=2	⊢	x ² + y-1 =21
AR	x=10Ay=2	⊢	x ² + y-1 =21 ∧ [y=1] [x=2] x+y=21
[=]	x=10Ay=2	⊢	[x=2] x+(y-1)=21 ∧ [y=1] [x=2] x+y=21
[=]	x=10Ay=2	⊢	[x=2] [y=1] x+y=21 ∧ [y=1] [x=2] x+y=21
[=]	x=10Ay=2	⊢	[x=2] [y=1] x+y=21 ∧ [x=2] x+y=21
[=]	x=10Ay=2	⊢	[x=2] [y=1] x+y=21 ∧ [y=1] x+y=21
parA	x=10Ay=2	⊢	[x=2] vV(y=1) ∧ [c] x+y=21
→R	x=10Ay=2	⊢	x=10Ay=2 → [(x=2] vV(y=1) ∧ c] x+y=21

Expand more details

x=10Ay=2	⊢	<?x>2.?y < 1? x+y=12v<?y < 1;?x>2? x+y=12
x=10Ay=2	⊢	<?x>2.?y < 3? x+y=12v<?y < 3;?x>2? x+y=12
x=10Ay=2	⊢	<?x>8.?y < 1? x+y=12v<?y < 1;?x>8? x+y=12
x=10Ay=2	⊢	<?x>8.?y < 3? x+y=12v<?y < 3;?x>8? x+y=12
x=10Ay=2	⊢	<?x>2.?y < 1? x+y=12v<?y < 1;?x>2? x+y=12
x=10Ay=2	⊢	<?x>2.?y < 3? x+y=12v<?y < 3;?x>2? x+y=12
x=10Ay=2	⊢	<?x>8.?y < 1? x+y=12v<?y < 1;?x>8? x+y=12
x=10Ay=2	⊢	<?x>8.?y < 3? x+y=12v<?y < 3;?x>8? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<?x>2.?y < 1? x+y=12v<?y < 1;?x>2? x+y=12
x=10Ay=2	⊢	<?x>2.?y < 3? x+y=12v<?y < 3;?x>2? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>2] vV(?y < 1) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>2] vV(?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1) ∧ c? x+y=12v<[?x>9] vV(?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>2] vV(?y < 1) ∧ c? x+y=12v<[?x>9] vV(?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1; u?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1; u?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1; u?y < 3) ∧ c? x+y=12
x=10Ay=2	⊢	<[?x>9] vV(?y < 1; u?y < 3) ∧ c? x+y=12