Hybrid I/O Automata

Nancy Lynch, MIT
Roberto Segala, University of Verona
Frits Vaandrager, University of Nijmegen

http://www.cs.kun.nl/~fvaan

I/O Automata (Lynch & Tuttle, '87; Jonsson '87)

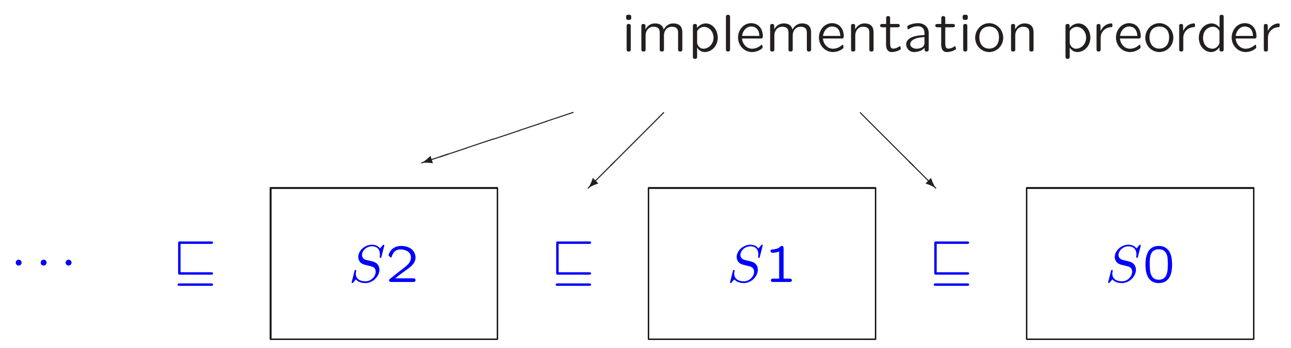
Purpose

Formal model for specification+verification of distributed algorithms

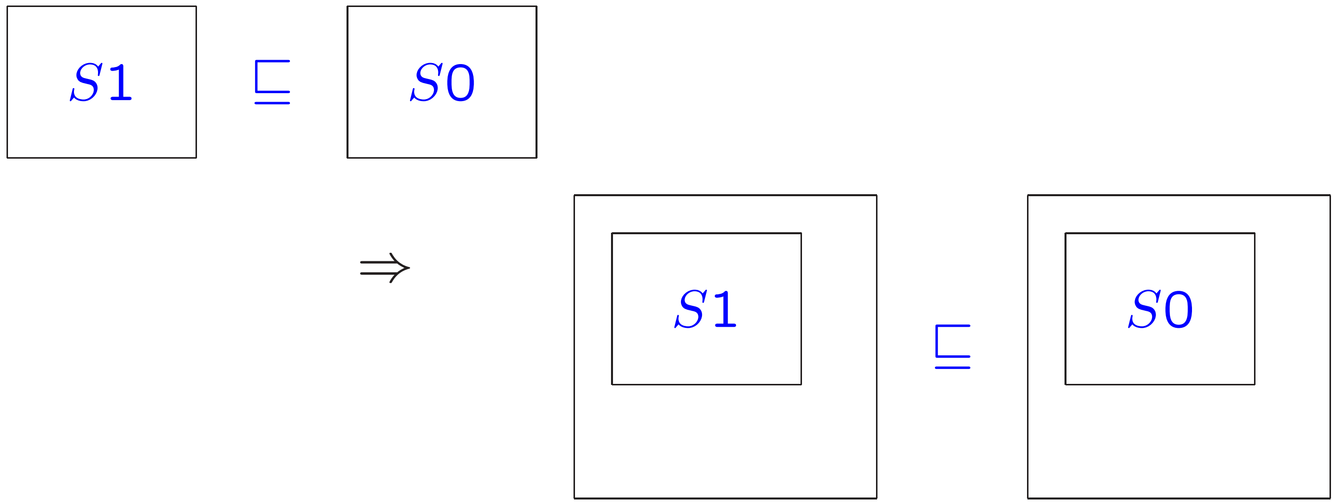
Characteristics:

- Both system and specification modelled as transition system
- Language inclusion as implementation relation
  ($\Rightarrow$ stepwise refinement!)
- Compositionality
- Distinction between input and output actions
- Fairness/liveness
- Assertional reasoning (invariants, simulations, etc)
- Extensions deal with real-time, hybrid, and probabilistic aspects

Stepwise Refinement



Compositionality



Extensions and Restrictions of IOA model

(S= Safe, F=Fair, L=Live, T=Timed, H=Hybrid, P=Probabilistic)

$$\begin{tikzcd}
 & & & \text{PIOA} & & & & \text{LIOA} \arrow[dddd] \\
\text{PA} \arrow[rr] \arrow[dddd] & & \text{SPIOA} \arrow[ur] \arrow[dddd] & & & & \text{FIOA} \arrow[ur] & \\
 & & & & & \text{IOA} \arrow[uull] \arrow[ur] & & \\
 & \text{A} \arrow[uul] \arrow[rrr] \arrow[ddd] & & & \text{SIOA} \arrow[uull] \arrow[ur] \arrow[ddd] & & & \\
 & & & & & & & \text{LTIOA} \\
\text{PTA} \arrow[rr] & & \text{PTIOA} & & & & & \\
 & \text{TA} \arrow[ul] \arrow[rrr] \arrow[d] & & & \text{TIOA} \arrow[ull] \arrow[uurrr] \arrow[d] & & & \\
 & \text{HA} \arrow[rrr] & & & \text{HIOA} & & &
\end{tikzcd}$$

I/O Distinction and Input Enabling

Advantages

- helps to avoid mistakes in specifications
- simple semantics in terms of traces
  (no need for failure pairs as in CSP)
- fairness/liveness becomes easier

Disadvantages

- less expressive
  (handshake needed to encode single CSP synchronization)
- process algebra becomes more difficult

Applications

1. Distributed algorithms!
2. Distributed operating systems
3. Database concurrency control
4. etc. etc.

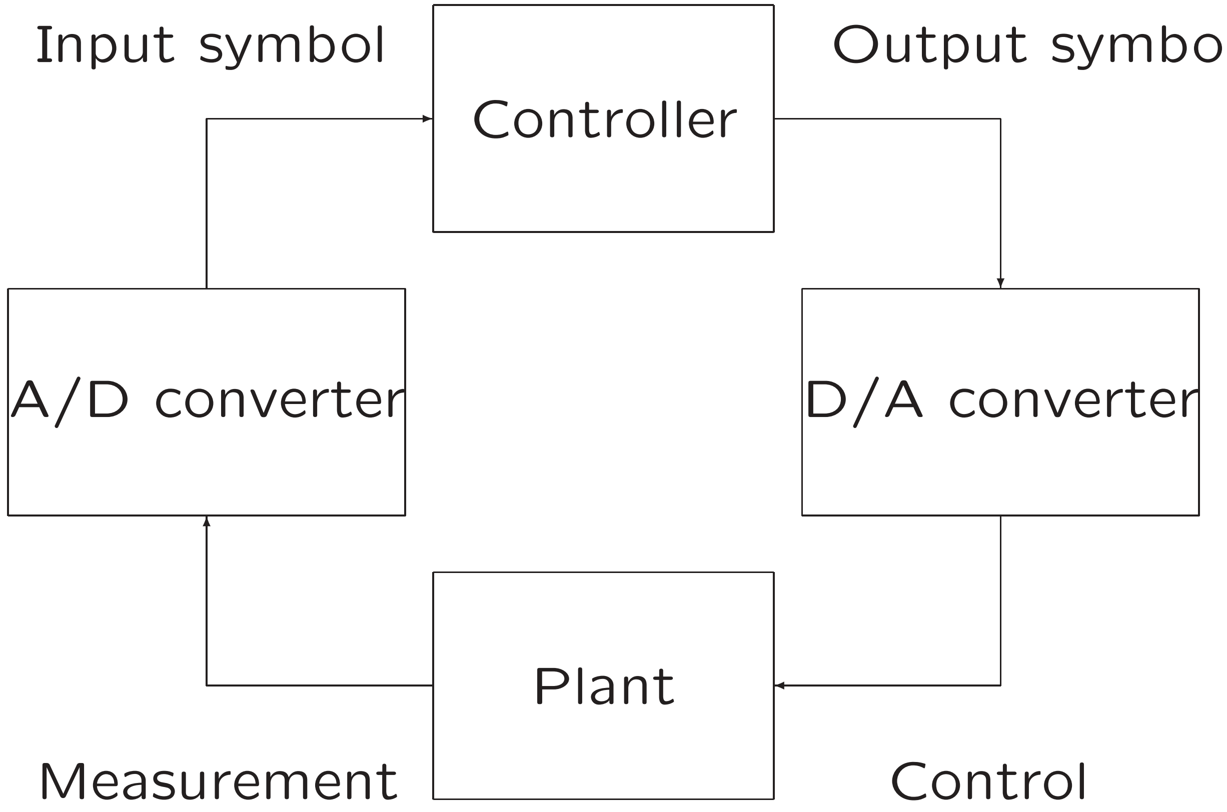
Background

In a timed automaton, all clocks proceed with the same rate in each location, i.e. $\dot{x} = 1$ for all clocks x in each location.

We may relax this condition and allow for (continuous) variables that evolve with arbitrary dynamics that may also depend on the location (see e.g. Maler, Manna & Pnueli, 1990).

The resulting structures are commonly called hybrid automata (HA). Variables of a HA may represent, a drifting clock, the pressure in a tank, the speed of a car, the temperature in a room, the position of a robot hand, the voltage on a wire, etc.

HAs appear to be an appropriate modelling formalism to support design and analysis of hybrid control systems:



In this lecture, I will focus on the following fundamental issues:

- What is the observable behavior of a HA? What does it mean for one HA to implement another?

- Compositionality

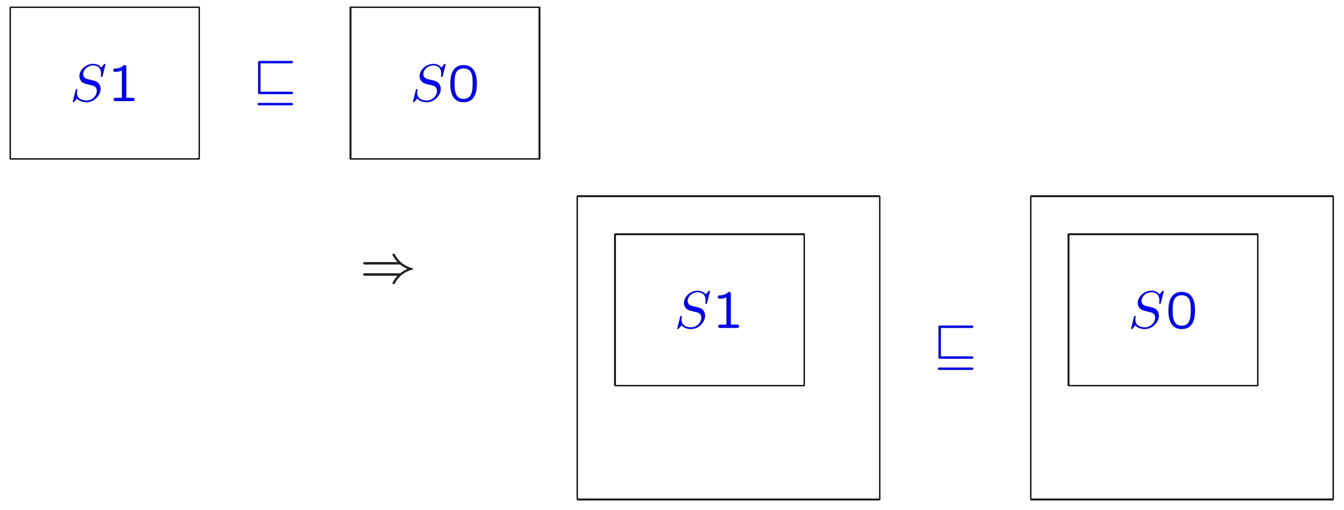
- Receptivity

This is all joint work with Nancy Lynch & Roberto Segala, improving/extending earlier results published in 1996 and 2001.

Stepwise Refinement

$$\begin{tikzcd}
 & & & & \text{implementation preorder} \arrow[dll] \arrow[dl] \arrow[dr] & & \\
\cdots & \sqsubseteq & \boxed{S2} & \sqsubseteq & \boxed{S1} & \sqsubseteq & \boxed{S0}
\end{tikzcd}$$

Compositionality



Terminology

The issues that I want to address in my talks are best studied at the semantic level. The objects in the semantic world that we define and study will be called hybrid automata, even though this leads to confusion with the syntactic objects with the same name. For the semantic objects, hybrid transition systems probably would have been a better name, just like I/O automata should probably have been called I/O transition systems.

Time

We assume a time axis T, which is a subgroup of $(\mathsf{R}, +)$, the real numbers with addition. We assume that every infinite, monotone, bounded sequence of elements of T has a limit in T.

Examples: the real numbers, the integers, $\{0\}$.

An interval J is a nonempty, convex subset of T.

Types We assume a universal set V of variables.

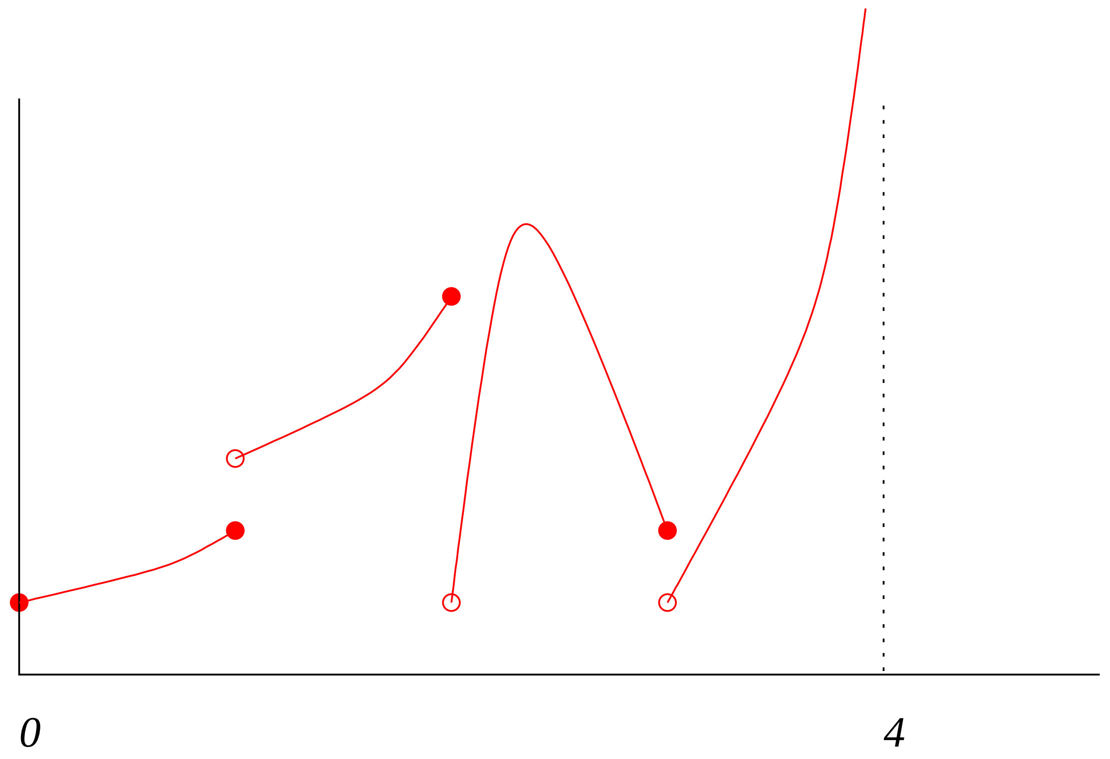
Each variable v has a static type $type(v)$, which is the set of values it may take.

In addition we assume a dynamic type $dtype(v)$, which is a set of functions from left-closed intervals of T to $type(v)$ that is closed under time shift, subinterval and pasting.

The pasting operations glues together a countable number of functions which all (possibly except for the last one) have a right-closed domain. At borderpoints value of leftmost function is taken.

Examples: (closure of) constant functions, continuous functions, differentiable functions, smooth functions, integrable functions, smooth functions with range $[-1, 1]$...

Example Element of Dynamic Type



Alternatives to pasting closure:
“stuttering” events [LSVW96] or superdense computations [Pnueli94].

Trajectories

Let V be a set of variables and J a left-closed interval of T with left endpoint equal to 0. Then a J-trajectory for V is a function $\tau : J \to val(V)$, such that for each $v \in V$, $\tau \downarrow v \in dtype(v)$.

Lemma
The set of trajectories for V together with the prefix ordering $\leq$, is an algebraic cpo.

A hybrid automaton (HA) is a tuple $\mathcal{A} = (W, X, Q, \Theta, E, H, D, \mathcal{T})$ with

- W and X disjoint sets of external resp internal variables.
  We call a valuation $\mathbf{x}$ for X a state and write $V \triangleq W \cup X$.
- $Q \subseteq val(X)$ a set of states and $\Theta \subseteq Q$ a nonempty set of start states.
- E and H sets of external resp internal actions.
  We write $A \triangleq E \cup H$ and let $a, a', a_1, a_2, \ldots$ range over A.
- $\mathcal{D} \subseteq Q \times A \times Q$ a set of discrete transitions.
  We write $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$ for $(\mathbf{x}, a, \mathbf{x}') \in D$.
- A set $\mathcal{T}$ of trajectories for V such that $\tau(t) \lceil X \in Q$ for all $\tau \in \mathcal{T}$ and $t \in \mathsf{T}$. We require that $\mathcal{T}$ is closed under prefix, suffix and countable concatenation.

Notation

In examples, unless specified otherwise, we take the time domain to be the set of real numbers.

If not specified, we assume the set of states Q equals the set $val(X)$ of all valuations of internal variables.

Notation

We specify sets of trajectories using differential and algebraic equations (or inclusions).

A trajectory satisfies algebraic equation $v = e$ if the constraint on the variables expressed by this equation holds for each point on the trajectory.

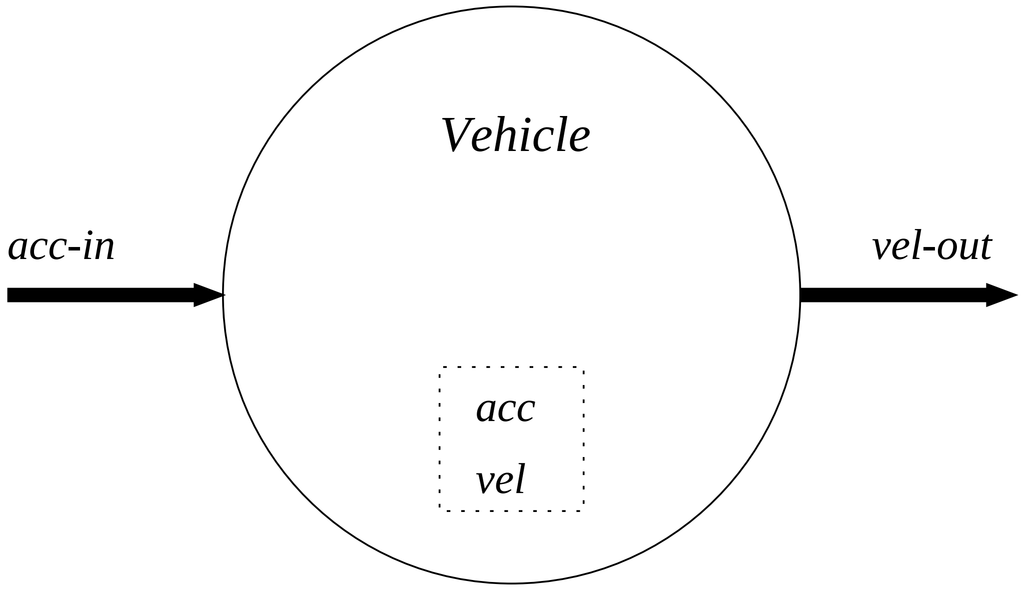
Trajectory τ satisfies differential equation $\dot{v} = e$ if, for every $t \in dom(\tau)$,

$$v(t) = v(0) + \int_0^t e(t')dt'$$

(cf "weak solutions" of Polderman and Willems).

Algebraic/differential inclusions are dealt with similarly.

Example HA $\mathit{Vehicle}$ follows a suggested acceleration approximately, to within an error of $\epsilon \geq 0$.



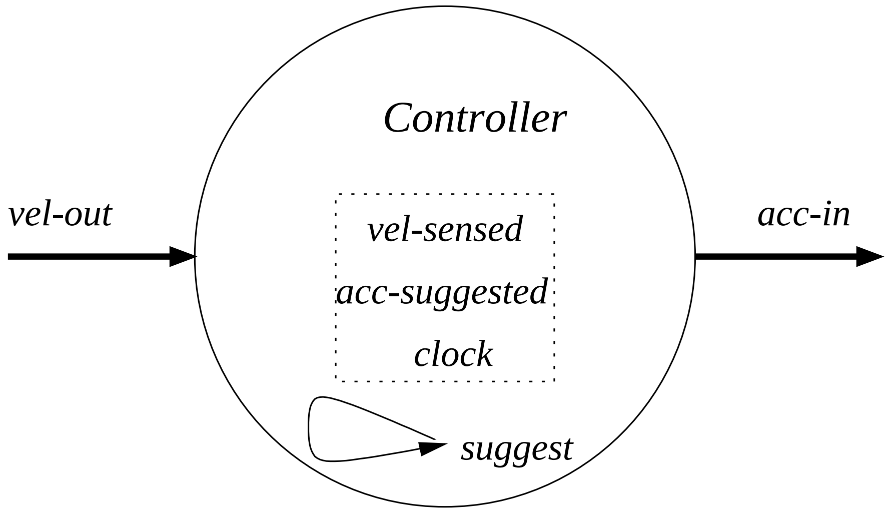
$W = \{\mathit{acc\text{-}in}, \mathit{vel\text{-}out}\}$, $X = \{\mathit{vel}, \mathit{acc}\}$, Θ assigns 0 to both state variables, and E, H and D are empty.

Example (cnt) All variables have type R. The dynamic type of the variables vel, $vel\text{-}out$, and $acc\text{-}in$ is the (pasting closure of the) set of continuous functions. The dynamic type of acc is the set of integrable functions. Set $\mathcal{T}$ consists of all trajectories that satisfy:

$$\begin{aligned} \dot{vel} &= acc \\ acc(t) &\in [acc\text{-}in(t) - \epsilon, acc\text{-}in(t) + \epsilon] \quad \text{for } t > 0 \\ vel\text{-}out &= vel \end{aligned}$$

(No constraints on values input variables in initial state of trajectories.)

Example HA $Controller$ suggests accelerations for a vehicle, with the intention of ensuring that the vehicle's velocity does not exceed a pre-specified velocity vmax.



Q is the set of valuations of X in which $clock \leq$ d, where d is a constant satisfying vmax $\geq \epsilon$ d. Θ assigns 0 to all state variables. $E = \emptyset$ and $H = \{suggest\}$.

Example (cnt)

All variables are of type R. The dynamic types of $vel\text{-}out$, $vel\text{-}sensed$, $acc\text{-}in$, and $clock$ are the (pasting closure of the) set of continuous functions, and $acc\text{-}suggested$ is a discrete variable.

Set D consists of the $suggest$ steps specified by:

$$\begin{aligned} clock &= \mathsf{d} \\ vel\text{-}sensed + (acc\text{-}suggested' + \epsilon)\mathsf{d} &\leq \mathsf{vmax} \\ clock' &= 0 \\ vel\text{-}sensed' &= vel\text{-}sensed \end{aligned}$$

Example (cnt)

Set $\mathcal{T}$ consists of all trajectories that satisfy:

$$\begin{aligned}
\dot{\mathit{acc\text{-}suggested}} &= 0 \\
\dot{\mathit{clock}} &= 1 \\
\mathit{vel\text{-}sensed}(t) &= \mathit{vel\text{-}out}(t) \qquad\qquad \text{for } t > 0 \\
\mathit{acc\text{-}in} &= \mathit{acc\text{-}suggested}
\end{aligned}$$

Executions and traces

An execution fragment of a hybrid automaton $\mathcal{A}$ is a sequence $\alpha = \tau_0\, a_1\, \tau_1\, a_2\, \tau_2 \ldots$, where (1) each τ_i is a trajectory in $\mathcal{T}$, and (2) if τ_i is not the last trajectory in α then $\tau_i.lstate \overset{a_{i+1}}{\rightarrow} \tau_{i+1}.fstate$.
An execution fragment α is defined to be an execution if its first state is a start state.

If α is an execution fragment, then the trace of α, denoted by $trace(\alpha)$, is obtained by (1) first projecting all trajectories of α on the variables in W, then (2) removing the actions in H, and finally (3) concatenating all adjacent trajectories.

We define a trace of $\mathcal{A}$ to be the trace of an execution of $\mathcal{A}$.

Implementation

Hybrid automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable if they have the same external interface, that is, if $W_1 = W_2$ and $E_1 = E_2$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable then we say that $\mathcal{A}_1$ implements $\mathcal{A}_2$, denoted by $\mathcal{A}_1 \leq \mathcal{A}_2$, if the traces of $\mathcal{A}_1$ are included among those of $\mathcal{A}_2$.

Example

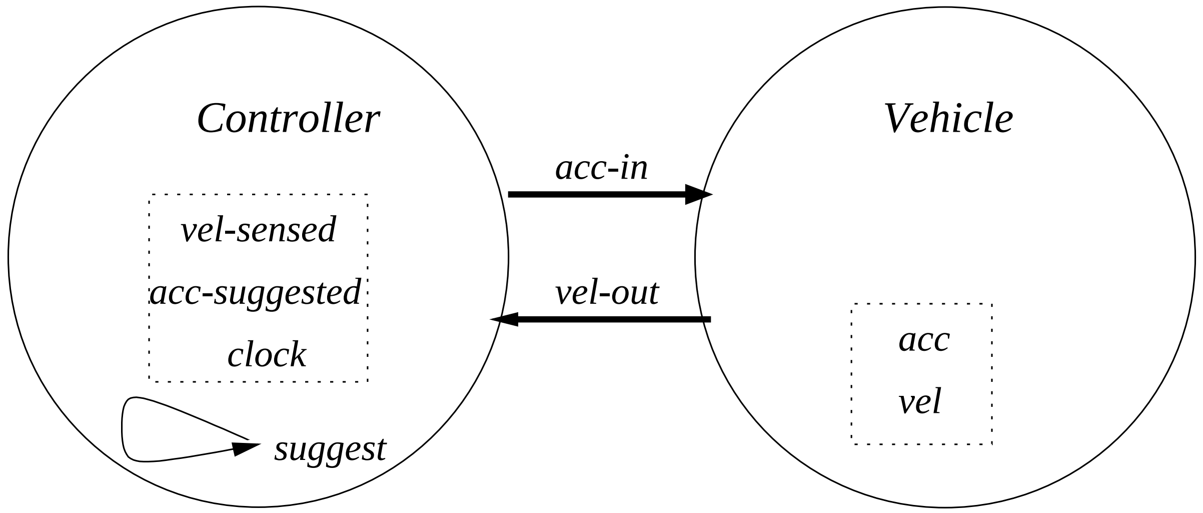
Denote the $\mathit{Vehicle}$ HA by $\mathit{Vehicle}(\epsilon)$, making the uncertainty parameter explicit. Assume $0 \leq \epsilon_1 \leq \epsilon_2$. We claim that $\mathit{Vehicle}(\epsilon_1) \leq \mathit{Vehicle}(\epsilon_2)$. We can show this by demonstrating that the identity mapping is a simulation relation.

Hybrid automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if $H_1 \cap A_2 = H_2 \cap A_1 = \emptyset$ and $X_1 \cap V_2 = X_2 \cap V_1 = \emptyset$.
If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible then their composition $\mathcal{A}_1 \| \mathcal{A}_2$ is the structure (a HA in fact) $\mathcal{A} = (W, X, Q, \Theta, E, H, D, \mathcal{T})$ where

- $W = W_1 \cup W_2$ and $X = X_1 \cup X_2$.
- $Q = \{\mathbf{x} \in \mathit{val}(X) \mid \mathbf{x} \lceil X_1 \in Q_1 \wedge \mathbf{x} \lceil X_2 \in Q_2\}$.
- $\Theta = \{\mathbf{x} \in Q \mid \mathbf{x} \lceil X_1 \in \Theta_1 \wedge \mathbf{x} \lceil X_2 \in \Theta_2\}$.
- $E = E_1 \cup E_2$ and $H = H_1 \cup H_2$.
- For each $\mathbf{x}, \mathbf{x}' \in Q$ and each $a \in A$, $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$ iff for $i = 1, 2$, either (1) $a \in A_i$ and $\mathbf{x} \lceil X_i \xrightarrow{a}_i \mathbf{x}' \lceil X_i$, or (2) $a \notin A_i$ and $\mathbf{x} \lceil X_i = \mathbf{x}' \lceil X_i$.
- $\mathcal{T} \subseteq \mathit{trajs}(V)$ is given by $\tau \in \mathcal{T} \Leftrightarrow \tau \downarrow V_1 \in \mathcal{T}_1 \ \wedge \ \tau \downarrow V_2 \in \mathcal{T}_2$.

Example

Consider the $Vehicle$ and $Controller$ automata (for the same ϵ). These two HAs are compatible.



By means of a standard inductive proof one may establish that, for all reachable states of the composed system, $vel \leq$ vmax.

Compositionality

Theorem Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable HAs with $\mathcal{A}_1 \leq \mathcal{A}_2$. Suppose $\mathcal{B}$ is an HA that is compatible with each of $\mathcal{A}_1$ and $\mathcal{A}_2$. Then $\mathcal{A}_1 \| B$ and $\mathcal{A}_2 \| B$ are comparable and $\mathcal{A}_1 \| \mathcal{B} \leq \mathcal{A}_2 \| \mathcal{B}$.

Hiding

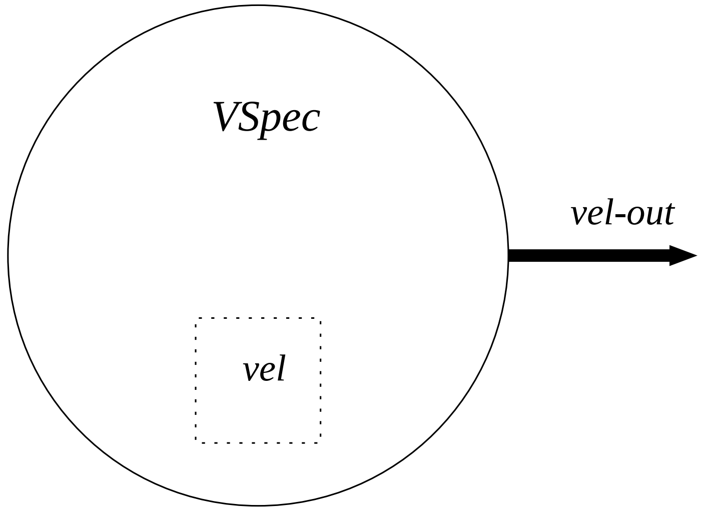
In [LSV02] we define two hiding operations for hybrid automata, $\mathsf{ActHide}(E, \mathcal{A})$ and $\mathsf{VarHide}(W, \mathcal{A})$, which hide actions resp variables. Both operations behave well wrt the trace implementation relation.

Example In the composition of the *Vehicle* and *Controller* HAs, we may hide the *acc-in* variable used for communication between the two components. Thus, we define

$$\mathcal{A} \;=\; \mathsf{VarHide}(\{\mathit{acc\text{-}in}\}, \mathit{Vehicle} \| \mathit{Controller}).$$

In the resulting automaton $\mathcal{A}$, the only external variable is *vel-out*.

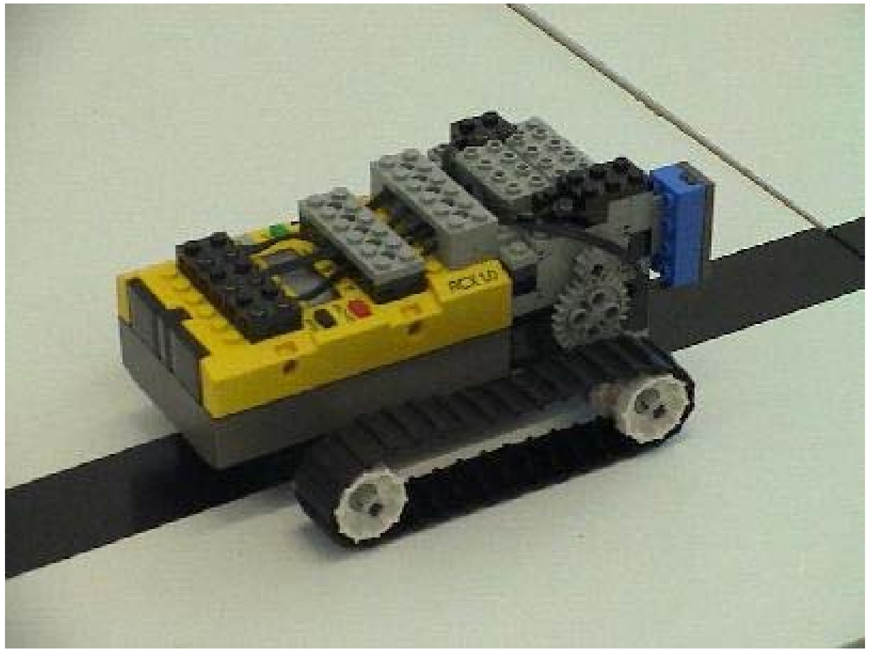
We may express the correctness of $\mathcal{A}$ by showing that it implements an abstract specification automaton $VSpec$ that simply represents the constraint that the vehicle's velocity is at most vmax.



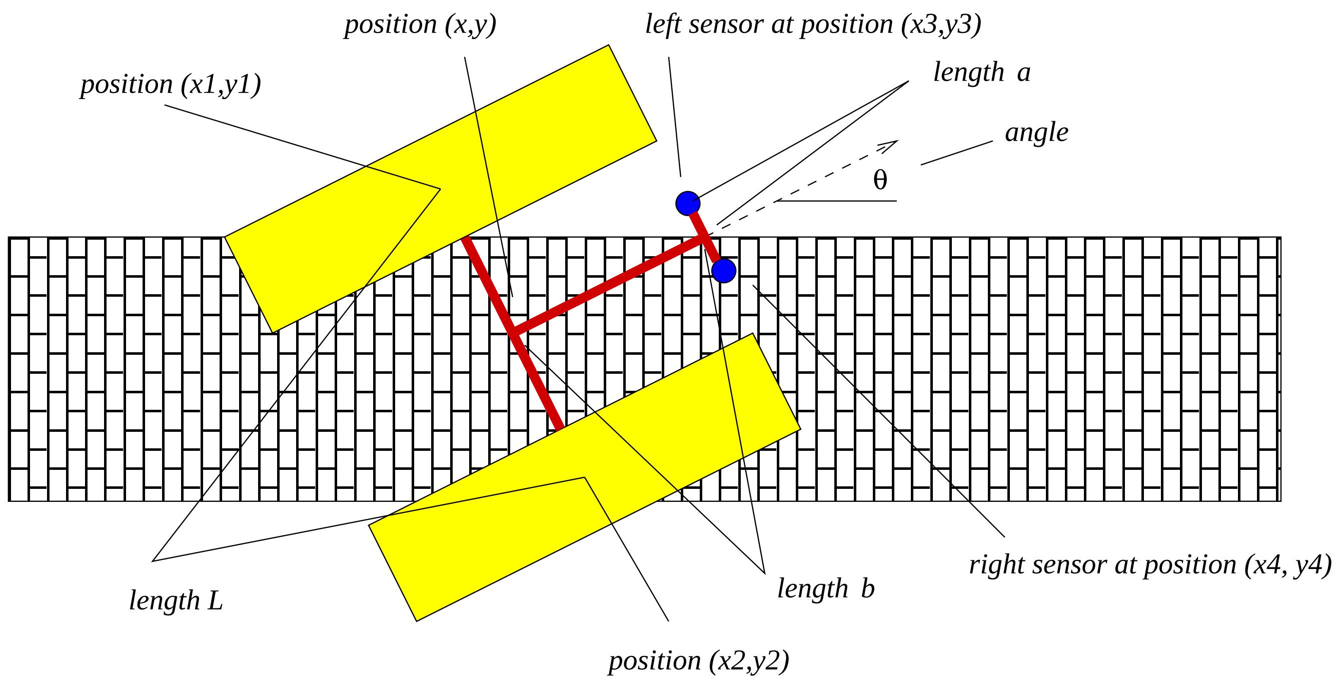
Q is the set of valuations of X in which $vel \leq$ vmax, $\Theta = Q$, $VSpec$ has no actions or discrete transitions. The trajectories of $VSpec$ are those that satisfy $vel\text{-}out = vel$, in each state.

Example: LEGO car

(joint work with Ansgar Fehnker and Miaomiao Zhang)



Operation of LEGO car



As long as sensor sees black background, opposite caterpillar moves forward.

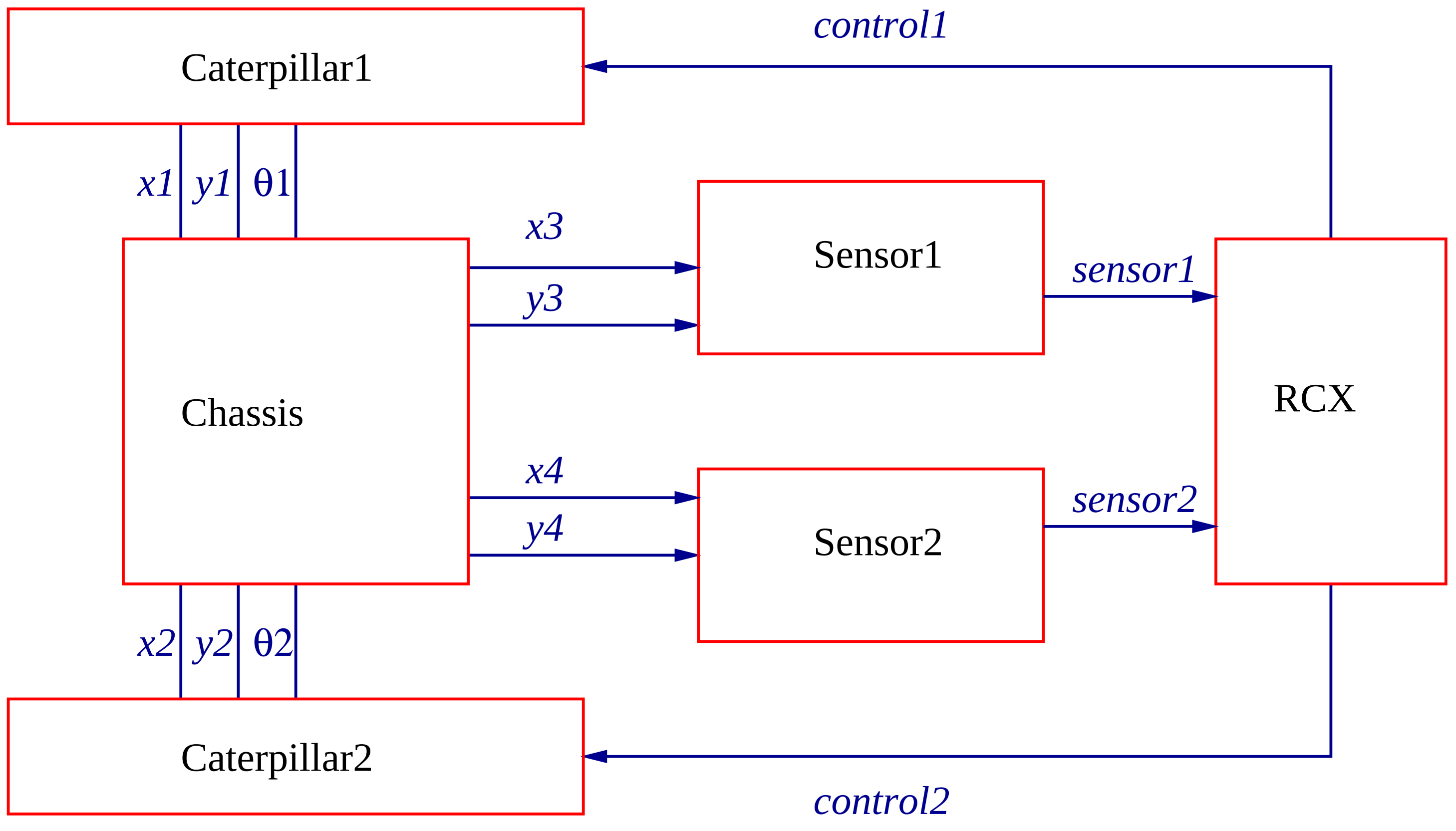
If it sees white background then opposite caterpillar moves backward.

Verification challenge

If orientation of the car differs too much from orientation of the black tape it may start bumping back and forth between different sides of the tape, and as a result even change the direction in which it moves.

Under which assumptions on the initial orientation can we be sure that the car will always move in a forward direction? (We assume the tape is infinite)

Network of Hybrid Automata for LEGO car



Chassis

Internal Variables

x, y, θ: differentiable

External Variables

$x1, y1, \theta1, x2, y2, \theta2, x3, y3, x4, y4$: differentiable

Initial States

$\theta \in [-\alpha, \alpha] \wedge y \in [-B, B] \wedge PLS \in [-B, B] \wedge PRS \in [-B, B]$

where $PLS = y + b \sin\theta + a \cos\theta$ and $PRS = y + b \sin\theta - a \cos\theta$

Equations

$$
\begin{aligned}
\theta 1 &= \theta 2 = \theta \\
x1 &= x - \frac{1}{2}L\sin\theta \\
y1 &= y + \frac{1}{2}L\cos\theta \\
x2 &= x + \frac{1}{2}L\sin\theta \\
y2 &= y - \frac{1}{2}L\cos\theta \\
x3 &= x + b\cos\theta - a\sin\theta \\
y3 &= y + b\sin\theta + a\cos\theta \\
x4 &= x + b\cos\theta + a\sin\theta \\
y4 &= y + b\sin\theta - a\cos\theta
\end{aligned}
$$

Caterpillar Treads

External Variables

$x1, y1, \theta 1$: differentiable
$\mathit{control1}$: Boolean, discrete

Equations

$$
\begin{aligned}
\dot{x}1 &= \textbf{if} \quad \mathit{control1} \quad \textbf{then} \quad V \cos \theta 1 \quad \textbf{else} \quad -V \cos \theta 1 \\
\dot{y}1 &= \textbf{if} \quad \mathit{control1} \quad \textbf{then} \quad V \sin \theta 1 \quad \textbf{else} \quad -V \sin \theta 1
\end{aligned}
$$

Sensors

External Variables

$x3, y3$: differentiable
$\mathit{sensor1}$: discrete, $\{\mathit{black}, \mathit{white}\}$

Equations

$$\mathit{sensor1} \quad = \quad \textbf{if} \quad y3 \in [-B, B] \quad \textbf{then} \quad \mathit{black} \quad \textbf{else} \quad \mathit{white}$$

RCX

Internal Variables

c: differentiable, $c \leq t_{sample}$
$\mathit{sample1}$, $\mathit{sample2}$: discrete, enumerated type $\{\mathit{black}, \mathit{white}\}$

Initial states

$$c = 0 \;\wedge\; \mathit{sample1} = \mathit{sample2} = \mathit{black}$$

External Variables

$\mathit{sensor1}$, $\mathit{sensor2}$: discrete, enumerated type $\{\mathit{black}, \mathit{white}\}$
$\mathit{control1}$, $\mathit{control2}$: discrete Boolean variables

Internal transition

$$c \geq t_{sample} \wedge c' = 0 \wedge \mathit{sample1}' = \mathit{sensor1} \wedge \mathit{sample2}' = \mathit{sensor2}$$

Variables $\mathit{sample1}$ and $\mathit{sample2}$ remain constant along a trajectory.

Equations

$$\begin{aligned}
\dot{c} &= 1 \\
\mathit{control1} &= \textbf{if}\ \ \mathit{sample2} = \mathit{black}\ \ \textbf{then}\ \ \mathit{true}\ \ \textbf{else}\ \ \mathit{false} \\
\mathit{control2} &= \textbf{if}\ \ \mathit{sample1} = \mathit{black}\ \ \textbf{then}\ \ \mathit{true}\ \ \textbf{else}\ \ \mathit{false}
\end{aligned}$$

Four Modes Depending on Values *control1* and *control2*

$$\begin{array}{rcl}
\mathit{control1} \wedge \mathit{control2} & \Rightarrow & \dot{x} = V \cos\theta \wedge \dot{y} = V \sin\theta \wedge \dot{\theta} = 0 \\
\mathit{control1} \wedge \neg\mathit{control2} & \Rightarrow & \dot{x} = 0 \wedge \dot{y} = 0 \wedge \dot{\theta} = \dfrac{-2V}{L} \\
\neg\mathit{control1} \wedge \mathit{control2} & \Rightarrow & \dot{x} = 0 \wedge \dot{y} = 0 \wedge \dot{\theta} = \dfrac{2V}{L} \\
\neg\mathit{control1} \wedge \neg\mathit{control2} & \Rightarrow & \dot{x} = -V \cos\theta \wedge \dot{y} = -V \sin\theta \wedge \dot{\theta} = 0
\end{array}$$

Results I

Using a (self written) tool that over approximates the set of reachable states based on bounded polyhedra, Ansgar Fehnker was able to verify that, assuming that initially the car moves forward with an angle between -45 and 45 degrees:

1. The car always stays on the tape and never moves backward.
2. The right sensor gets never closer to the upper boundary of the tape than 2.1 mm.
3. If the car is in forward mode the car moves in the direction of the x-axis with at least 8.9 cm/s (speed of car is 13 cm/s).

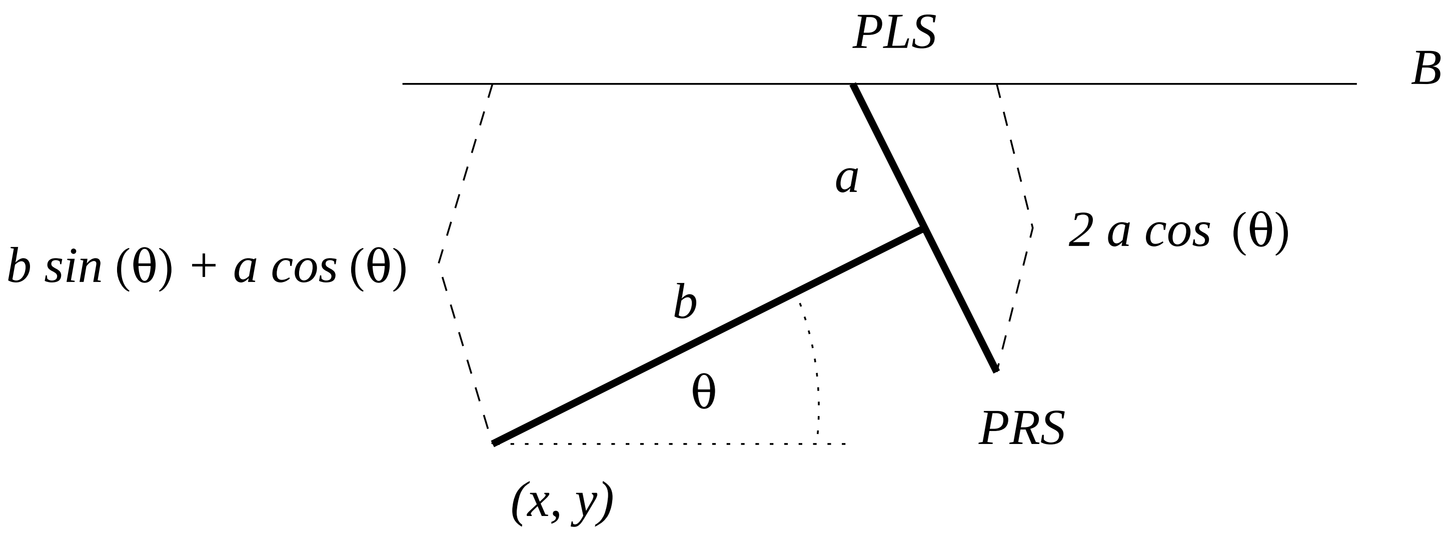
Experiments with the physical car confirm these results.

Results II

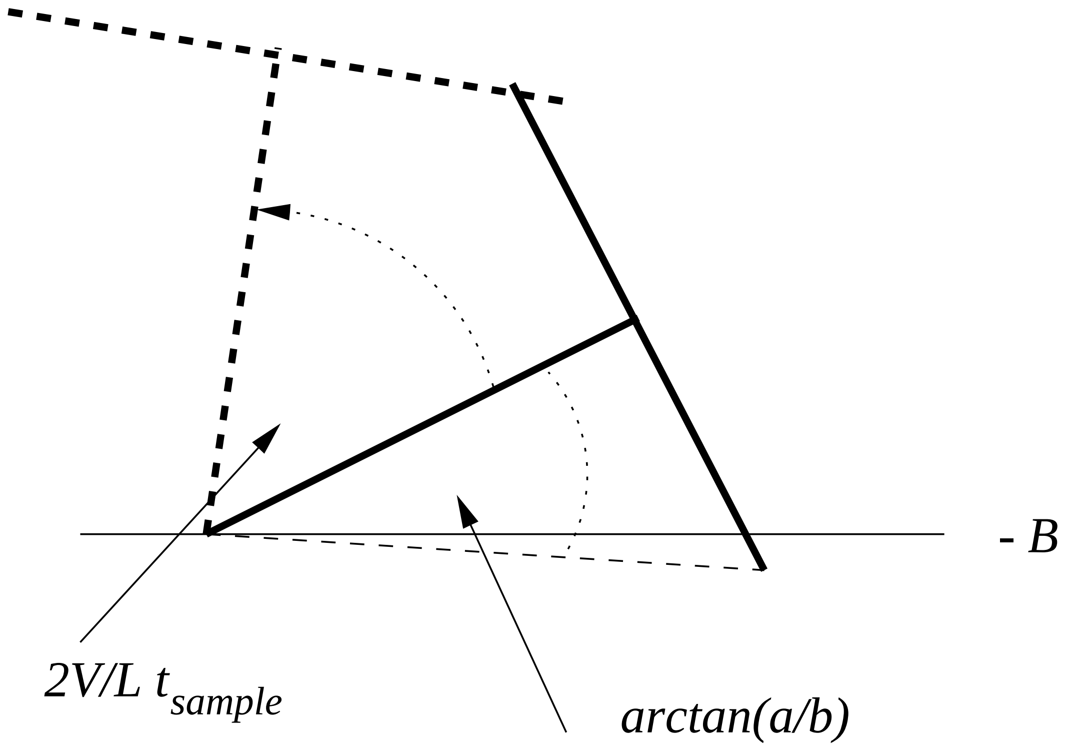
If the following constraints on the parameters hold, the car will never move backward, and infinitely often be in forward mode:

$$\begin{aligned}
\varphi_1 &= a\cos(\alpha) + b\sin(\alpha) \geq V\sin(\alpha)t_{sample} \\
\varphi_2 &= 2a\cos(\alpha) \geq V\sin(\alpha)t_{sample} \\
\varphi_3 &= \frac{2V}{L}t_{sample} + \arctan(\frac{a}{b}) \leq \alpha \\
\varphi_4 &= a\cos(\frac{V}{L}t_{sample}) + b\sin(\frac{V}{L}t_{sample}) \leq B
\end{aligned}$$

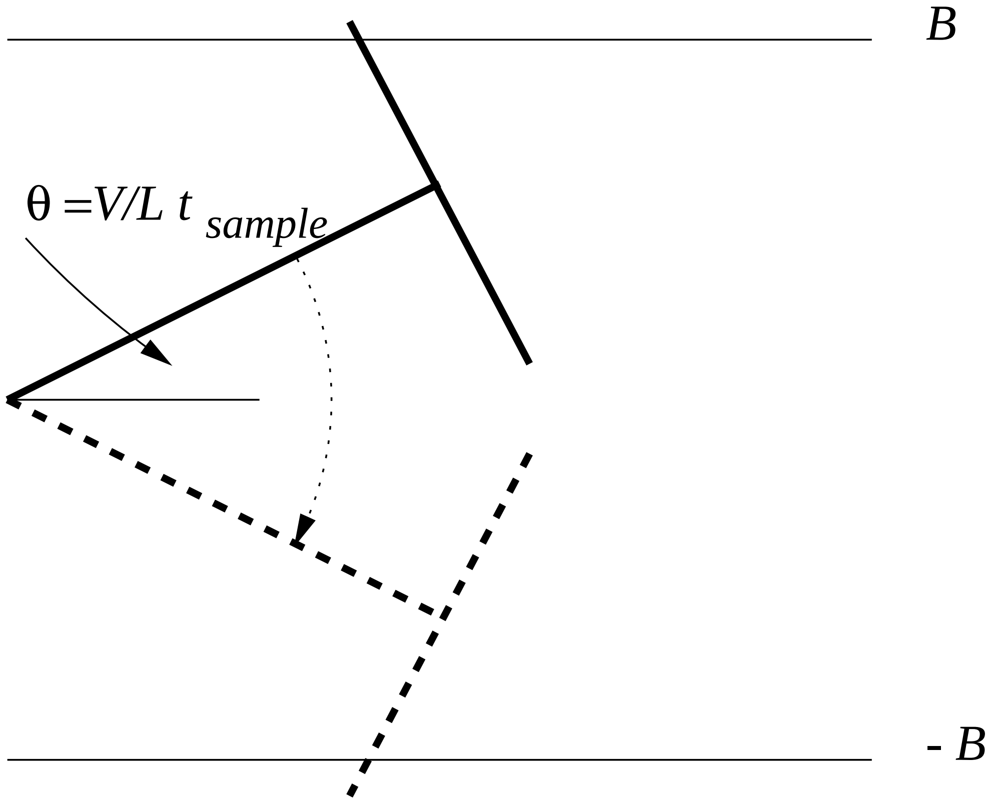
Why are constraints φ_1 and φ_2 needed?



Why is constraint φ_3 needed?



Why is constraint φ_4 needed?



Results III

Extending analysis to include disturbances is easy!!!

Hybrid I/O Automata

A hybrid I/O automaton (HIOA) $\mathcal{A}$ is a tuple $(\mathcal{H}, U, Y, I, O)$ where

- $\mathcal{H} = (W, X, Q, \Theta, E, H, D, \mathcal{T})$ is a hybrid automaton.
- U and Y partition W into input and output variables, resp.
  Variables in $Z \triangleq X \cup Y$ are called locally controlled; $V \triangleq W \cup X$.
- I and O partition E into input and output actions, resp.
  Actions in $L \triangleq H \cup O$ are called locally controlled; $A \triangleq E \cup H$.

such that ...

... the following axioms are satisfied:

E1 (Input action enabling)
For all $\mathbf{x} \in Q$ and all $a \in I$ there exists $\mathbf{x}' \in Q$ such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$.

E2 (Input trajectory enabling)
For all $\mathbf{x} \in Q$ and all $\upsilon \in \mathit{trajs}(U)$, there exists $\tau \in \mathcal{T}$ such that $\tau.\mathit{fstate} = \mathbf{x}$, $\tau \downarrow U \leq \upsilon$, and either
1. $\tau \downarrow U = \upsilon$, or
2. τ is closed and some $l \in L$ is enabled in $\tau.\mathit{lstate}$.

A pre-HIOA is a structure as above, except that it need not to satisfy **E1** and **E2**.

Example

Chassis and Caterpillers of LEGO car cannot be viewed as HIOAs

However, their composition is a HIOA

Sensors and RCX are also HIOAs

Composition

Pre-HIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if $\mathcal{H}_1$ and $\mathcal{H}_2$ are compatible and $Y_1 \cap Y_2 = O_1 \cap O_2 = \emptyset$.

If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible pre-HIOAs then their composition $\mathcal{A}_1 \| \mathcal{A}_2$ is the tuple $\mathcal{A} = (\mathcal{H}, U, Y, I, O)$ where

- $\mathcal{H} = \mathcal{H}_1 \| \mathcal{H}_2$,
- $Y = Y_1 \cup Y_2$,
- $U = (U_1 \cup U_2) - Y$,
- $O = O_1 \cup O_2$, and
- $I = (I_1 \cup I_2) - O$.

Problem The composition of two pre-HIOAs is again a pre-HIOA. However, the composition of two HIOAs is not always a HIOA: the resulting structure not always satisfies **E2**!

Example Suppose $\mathcal{A}_1$ has no discrete steps, input variable v_1, output variable v_2, and as trajectories all functions that satisfy

$$v_2(t) = v_1(t) + 1 \qquad \text{for } t > 0$$

Symmetrically, suppose $\mathcal{A}_2$ has no discrete steps, input variable v_2, output variable v_1, and as trajectories all functions that satisfy

$$v_1(t) = v_2(t) + 1 \qquad \text{for } t > 0$$

Then the composed system has only point trajectories and does not satisy **E2**.

Theorem If $\mathcal{A}_1$ and $\mathcal{A}_2$ are pre-HIOAs that satisfy **E1**, then the composition $\mathcal{A}_1 \| \mathcal{A}_2$ also satisfies **E1**.

Theorem Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two compatible HIOAs such that $U_1 \cap Y_2 = \emptyset$. Then $\mathcal{A}_1 \| \mathcal{A}_2$ is a HIOA.

An HIOA is oblivious if it satisfies:

OBL Let $\tau \in \mathcal{T}$ and $\upsilon \in trajs(U)$ such that $dom(\tau) = dom(\upsilon)$. Then there exists $\tau' \in \mathcal{T}$ such that:

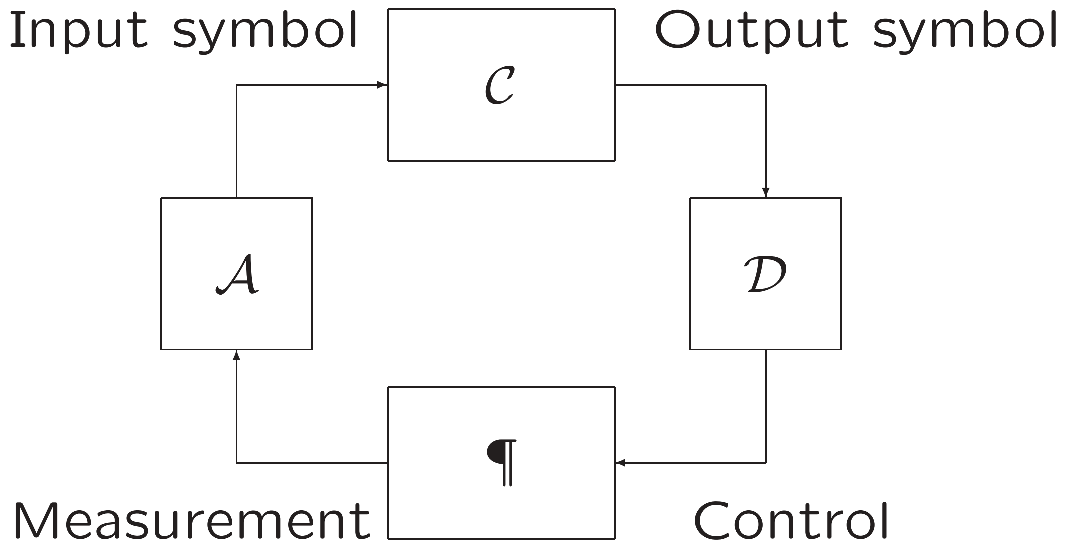
1. $\tau' \downarrow U = \upsilon$.

2. $\tau' \downarrow Y = \tau \downarrow Y$.

3. If τ is closed and some locally controlled action is enabled in $\tau.lstate$ then some locally controlled action is enabled in $\tau'.lstate$.

Theorem Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two compatible HIOAs and suppose that $\mathcal{A}_1$ is oblivious. Then $\mathcal{A}_1 \| \mathcal{A}_2$ is a HIOA.

Example: Hybrid Control System



Zeno

An execution fragment is Zeno if it is time-bounded and is either an infinite sequence, or else a finite sequence ending with a trajectory whose domain is right open.

An execution fragment is locally-Zeno if it is Zeno and contains infinitely many locally controlled actions.

A pre-HIOA is progressive if it has no locally-Zeno execution fragments.

Theorem A progressive HIOA is I/O feasible, i.e. able to follow sequence of input trajectories interleaved with input actions.

Theorem The composition of progressive pre-HIOAs is progressive.

Problem

HIOAs involving only upper bounds on timing of events are typically not progressive. Still, we very much like to use such HIOAs in specifications.

Solution

Introduce notion of receptiveness.

Concept has been studied earlier by e.g. Dill and Abadi & Lamport in terms of two-player games. We can use a simpler definition since our model does not involve general liveness properties.

Receptiveness

A strategy for a pre-HIOA $\mathcal{A}$ is an HIOA $\mathcal{A}'$ that differs from $\mathcal{A}$ only in that $D' \subseteq D$ and $\mathcal{T}' \subseteq \mathcal{T}$.

A pre-HIOA is progressive if it has no locally-Zeno execution fragments.

A pre-HIOA is receptive if it has a progressive strategy.

Theorem Every receptive pre-HIOA is I/O feasible.

Theorem Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two compatible receptive HIOAs with progressive strategies $\mathcal{A}'_1$ and $\mathcal{A}'_2$ such that $\mathcal{A}'_1 \| \mathcal{A}'_2$ is an HIOA. Then $\mathcal{A}_1 \| \mathcal{A}_2$ is a receptive HIOA with progressive strategy $\mathcal{A}'_1 \| \mathcal{A}'_2$.

Conclusions / Future Work

- HIOA model is compositional and supports stepwise refinement.
- Model should be tested further by using it to describe and analyze many more ambitious examples.
- Examples may come from area of embedded systems but for instance also from biology or psychology.
- Probabilities need to be added.
- Need to incorporate additional analysis methods, e.g. Lyapunov stability analysis and robust control methods.
- Much work required to automate these calculations!

Thank you for listening and for your comments!!!