Logic and Hybrid Systems

Manasvi Saxena

Formal Systems Lab, UIUC

Hybrid Systems

- Dynamical Systems exhibiting both discrete (jump) and continuous (flow) behaviors.
- Serve as models of physical systems, from thermostats to trains.
- Continuous dynamics specified using Differential Equations.

Differential Dynamic Logic (dL)

- Main focus - Differential Dynamic Logic for Hybrid Systems (Andre Platzer).

Differential Dynamic Logic (dL)

- Main focus - Differential Dynamic Logic for Hybrid Systems (Andre Platzer).
- Practical deductive verification of hybrid systems.

Differential Dynamic Logic (dL)

- Main focus - Differential Dynamic Logic for Hybrid Systems (Andre Platzer).
- Practical deductive verification of hybrid systems.
- Introduces Hybrid Program - program notation for hybrid systems.

Differential Dynamic Logic (dL)

- Main focus - Differential Dynamic Logic for Hybrid Systems (Andre Platzer).
- Practical deductive verification of hybrid systems.
- Introduces Hybrid Program - program notation for hybrid systems.
- Dynamic Logic for Hybrid Programs, a generalization of Dynamic Logic.

Differential Dynamic Logic (dL)

- Main focus - Differential Dynamic Logic for Hybrid Systems (Andre Platzer).
- Practical deductive verification of hybrid systems.
- Introduces Hybrid Program - program notation for hybrid systems.
- Dynamic Logic for Hybrid Programs, a generalization of Dynamic Logic.
- Suited for automation.

Hybrid Automata

- Commonly used to model Hybrid Systems, via Graphs.
- Nodes specify continuous dynamics. Edges describe discrete transitions.
- Intuitive, but not suitable for deductive verification.

Hybrid Automata

- Commonly used to model Hybrid Systems, via Graphs.
- Nodes specify continuous dynamics. Edges describe discrete transitions.
- Intuitive, but not suitable for deductive verification.

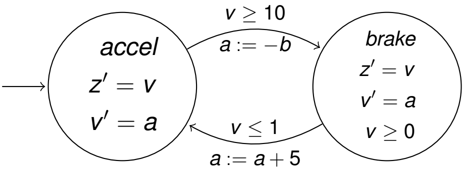


Figure: Hybrid Automata (simplified) of a Train Control System

- **First Order Logic** - No builtin means for referring to state transitions.
- **Temporal Logics** - Modal operators allow referring to state transitions. But valid formulas only express generic facts.

- **First Order Logic** - No builtin means for referring to state transitions.
- **Temporal Logics** - Modal operators allow referring to state transitions. But valid formulas only express generic facts.
- **Dynamic Logic (DL)** - Combines operational system models with operators for reasoning.
  - Provides parameterized modal operators, $[\alpha]$, $\langle\alpha\rangle$ that refer to states reachable by system α.
  - $[\alpha]\phi$ expresses all states reachable by α satisfy ϕ, allowing reasoning about discrete systems.
  - Say $(b > 0) \rightarrow [a := -b](a < 0)$ expresses a discrete transition. We can prove $(b > 0) \vdash (a < 0)[b/a]$ using DL's calculus.
  - No built in notion for describing or reasoning about continuous dynamics.

- Generalize DL so operational models α can be used in modal formulas like $[\alpha]\phi$. dL refers to generalized models as “Hybrid Programs”.
- A compositional calculus for verification. Decompose $[\alpha]\phi$ into an equivalent formula $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$.
- Prove subsystems and subproperties $[\alpha_i]\phi_i$ independently and combine results conjuntively.
- Complete relative to handling of differential equations.

dL formulas built over

- V, set of real-valued logical variables and signature Σ containing functions, predicate symbols over reals, like $0, 1, +, \geq$.
- Signature Σ containing functions and predicates, like $0, 1 \geq$. Σ also contains *System State Variables*. Unlike rigid symbols, like $1, 2$, their interpretation can change from state to state.
- Set $\mathrm{Trm}(\Sigma, V)$ of *terms* defined as classical FOL polynomial (or rational) expressions over V with additional skolem terms $s(X_1, \ldots, X_n)$, where $X_1, \ldots, X_n \in V$.

Hybrid Programs

Consider $x_i \in \Sigma$, $\theta_i, \vartheta_i \in \mathrm{Trm}(\Sigma, V)$ for $1 \leq i \leq n$, χ a (Σ, V) FOL-formula, $\alpha, \beta \in \mathrm{HP}(\Sigma, V)$ Set $\mathrm{HP}(\Sigma, V)$, is defined inductively as -

- $(x_1 := \theta_1, \ldots, x_n := \theta_n) \in \mathrm{HP}(\Sigma, V)$
- $(x_1' = \vartheta_i, \ldots, x_n' = \vartheta_n) \,\&\, \chi \in \mathrm{HP}(\Sigma, V)$. $x_i' = \vartheta_i$ is a differential equation where x_i' is the first order time derivative of x_i.
- $(?\chi) \in \mathrm{HP}(\Sigma, V)$.
- $\alpha \cup \beta \in \mathrm{HP}(\Sigma, V)$.
- $\alpha; \beta \in \mathrm{HP}(\Sigma, V)$.
- $\alpha^* \in \mathrm{HP}(\Sigma, V)$.

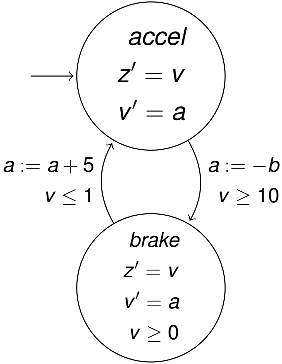


Figure: Hybrid Automata of Simple Train Control System

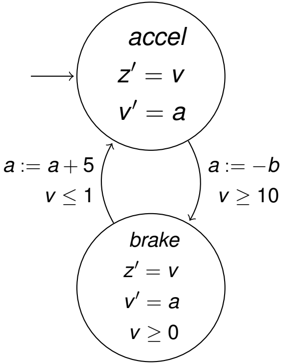


Figure: Hybrid Automata of Simple Train Control System

$$
\begin{aligned}
&q := accel;\\
&((?q = accel; z' = v, v' = a)\\
&\cup (?q = accel \wedge z \geq s; a := -b; q := brake; ?v \geq 0)\\
&\cup (?q = brake; z' = v, v' = a \& v \geq 0)\\
&\cup (?q = brake \wedge \leq 1; a := a + 5; q := accel))*
\end{aligned}
$$

dL Formulas

Some Notation

- Interpretation I assigns functions and relations over Reals to rigid symbols in Σ.
- A state is a map $\nu : \Sigma_{fl} \rightarrow \mathbb{R}$.
- Assignment of logical variables is a map $\eta : V \rightarrow \mathbb{R}$.
- Note the difference between *Logical* and *State Variables*. The evaluation of a state variable “evolves” across states. Logical Variables can be quantified over, but not state variables.
- The models of dL are Kripke Structures, where nodes are hybrid system states.

dL Valuation of dL Terms

Say $val_{I,\eta}(\nu, \cdot)$ is evaluation w.r.t. interpretation I, assignment η and state ν.

dL Valuation of dL Terms

Say $val_{I,\eta}(\nu, \cdot)$ is evaluation w.r.t. interpretation I, assignment η and state ν.

- (State Variables) $val_{I,\eta}(\nu, a) = \nu(a), x \in \Sigma_{fl}$

dL Valuation of dL Terms

Say $val_{I,\eta}(\nu, \cdot)$ is evaluation w.r.t. interpretation I, assignment η and state ν.

- (State Variables) $val_{I,\eta}(\nu, a) = \nu(a), x \in \Sigma_{fl}$
- (Logical Variables) $val_{I,\eta}(\nu, x) = \eta(x), x \in V$

dL Valuation of dL Terms

Say $val_{I,\eta}(\nu, \cdot)$ is evaluation w.r.t. interpretation I, assignment η and state ν.

- (State Variables) $val_{I,\eta}(\nu, a) = \nu(a), x \in \Sigma_{fl}$
- (Logical Variables) $val_{I,\eta}(\nu, x) = \eta(x), x \in V$
- (Rigid Symbols)
  $val_{I,\eta}(\nu, f(\theta_1, \ldots, \theta_n)) = f_I(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n))$
  where f is n-ary rigid symbol in Σ

Valuation of dL Formulas

- $val_{I,\eta}(\nu, p(\theta_1, \ldots, \theta_n)) = p_I(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n))$
- $val_{I,\eta}(\nu, \varphi \wedge \psi) = \top$ iff $val_{I,\eta}(\nu, \varphi) = \top \wedge val_{I,\eta}(\nu, \psi) = \top$.
  Similarly for $\rightarrow, \neg, \vee$
- $val_{I,\eta}(\nu, \exists x.\, \varphi) = \top$ iff $val_{I,\eta[x\mapsto d]}(\nu, \varphi) = \top$ for some $d \in \mathbb{R}$

Valuation of dL Formulas

- $val_{I,\eta}(\nu, p(\theta_1, \ldots, \theta_n)) = p_I(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n))$
- $val_{I,\eta}(\nu, \varphi \land \psi) = \top$ iff $val_{I,\eta}(\nu, \varphi) = \top \land val_{I,\eta}(\nu, \psi) = \top$. Similarly for $\rightarrow, \neg, \lor$
- $val_{I,\eta}(\nu, \exists x. \varphi) = \top$ iff $val_{I,\eta[x \mapsto d]}(\nu, \varphi) = \top$ for some $d \in \mathbb{R}$
- $val_{I,\eta}(\nu, [\alpha]\varphi) = \top$ iff $val_{I,\eta}(\omega, \varphi) = \top$ for all states ω with $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$.
- $val_{I,\eta}(\nu, \langle\alpha\rangle\varphi) = \top$ iff $val_{I,\eta}(\omega, \varphi) = \top$ for some state ω with $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$.

Transition Semantics of Hybrid Programs

Evaluation $\rho_{I,\eta}(\alpha)$ of HP α. $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$ means state ω is reachable from ν by operations of α.

- $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1), \ldots, x_n \mapsto val_{I,\eta}(\nu, \theta_n)] = \omega$ and $\forall y \in (\Sigma_{fl} - \{x_1, \ldots, x_n\}).val_{I,\eta}(\nu, y) = val_{I,\eta}(\omega, y)$.

Transition Semantics of Hybrid Programs

Evaluation $\rho_{I,\eta}(\alpha)$ of HP α. $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$ means state ω is reachable from ν by operations of α.

- $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1), \ldots, x_n \mapsto val_{I,\eta}(\nu, \theta_n)] = \omega$ and $\forall y \in (\Sigma_{fl} - \{x_1, \ldots, x_n\}).val_{I,\eta}(\nu, y) = val_{I,\eta}(\omega, y)$.
- $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$

Transition Semantics of Hybrid Programs

Evaluation $\rho_{I,\eta}(\alpha)$ of HP α. $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$ means state ω is reachable from ν by operations of α.

- $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1), \ldots, x_n \mapsto val_{I,\eta}(\nu, \theta_n)] = \omega$ and $\forall y \in (\Sigma_{fl} - \{x_1, \ldots, x_n\}). val_{I,\eta}(\nu, y) = val_{I,\eta}(\omega, y)$.
- $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$
- $\rho_{I,\eta}(?\chi) = \{(\nu, \nu) : val_{I,\eta}(\nu, \chi) = \top\}$

Transition Semantics of Hybrid Programs

Evaluation $\rho_{I,\eta}(\alpha)$ of HP α. $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$ means state ω is reachable from ν by operations of α.

- $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1), \ldots, x_n \mapsto val_{I,\eta}(\nu, \theta_n)] = \omega$ and $\forall y \in (\Sigma_{fl} - \{x_1, \ldots, x_n\}).val_{I,\eta}(\nu, y) = val_{I,\eta}(\omega, y)$.
- $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$
- $\rho_{I,\eta}(?\chi) = \{(\nu, \nu) : val_{I,\eta}(\nu, \chi) = \top\}$
- $\rho_{I,\eta}(\alpha; \beta) = \{(\nu, \omega) : (\nu, z) \in \rho_{I,\eta}(\alpha) \wedge (z, \omega) \in \rho_{I,\eta}(\beta)$ for some state $z\}$

Transition Semantics of Hybrid Programs

Evaluation $\rho_{I,\eta}(\alpha)$ of HP α. $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$ means state ω is reachable from ν by operations of α.

- $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1), \ldots, x_n \mapsto val_{I,\eta}(\nu, \theta_n)] = \omega$ and $\forall y \in (\Sigma_{fl} - \{x_1, \ldots, x_n\}). val_{I,\eta}(\nu, y) = val_{I,\eta}(\omega, y)$.
- $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$
- $\rho_{I,\eta}(?\chi) = \{(\nu, \nu) : val_{I,\eta}(\nu, \chi) = \top\}$
- $\rho_{I,\eta}(\alpha; \beta) = \{(\nu, \omega) : (\nu, z) \in \rho_{I,\eta}(\alpha) \wedge (z, \omega) \in \rho_{I,\eta}(\beta)$ for some state $z\}$
- $(\nu, \omega) \in \rho_{I,\eta}(\alpha^*)$ iff for $n \in \mathbb{N}$, there are states $\nu = \nu_0, \ldots, \nu_n = \omega$ s.t. $(\nu_i, \nu_{i+1}) \in \rho_{I,\eta}(\alpha)$ for $0 \leq i < n$.

Transition Semantics of Hybrid Programs

$(\nu, \omega) \in \rho_{I,\eta}((x'_1 = \vartheta_1 \dots x'_n = \vartheta_n) \& \chi)$ iff there is a flow of some duration $r \geq 0$, from ν to ω respecting the differential equations and evolution domain. Formally, there is a function $f : [0, r] \rightarrow \text{Sta}(\Sigma)$ such that -

- $f(0) = \nu$ and $f(r) = \omega$.
- $\forall \delta : [0, r].val_{I,\eta}(f(\delta), x_i)$ is continuous and $val_{I,\eta}(f(\delta), \chi) = \top$.
- $\forall \epsilon : (0, r).val_{I,\eta}(f(\epsilon), \vartheta_i) = \dot{f}(\epsilon)$.
- For any $z \notin \{x_1, x_2, \dots, x_n\}$, $val_{I,\eta}(f(\zeta), z)$ remains constant for $\zeta \in [0, r]$. In other words, all other state variables remain unchanged.

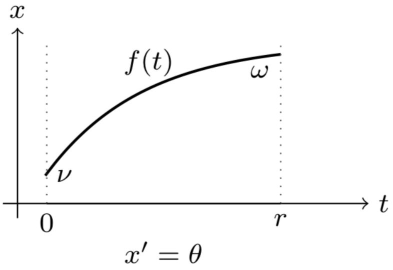


Figure: Unbounded Evolution

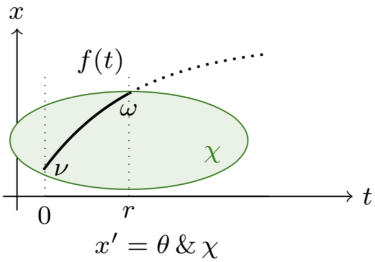


Figure: Evolution bound by χ

Some Intuition behind dL

- Generally, want to reason about Embedded Systems (ES) or Cyber Physical Systems (CPS).

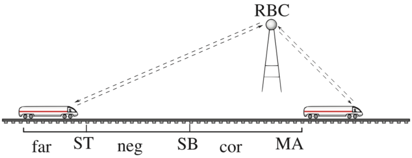
Some Intuition behind dL

- Generally, want to reason about Embedded Systems (ES) or Cyber Physical Systems (CPS).
- Loosely speaking, an ES or CPS is built upon the integration of Computation (Controller) and Physical Systems (Plant). Examples include autopilot, ABS (Anti Lock Braking Systems), fly-by-wire (human interaction).
- Expressed in dL as an HP $\alpha \equiv (\mathit{ctrl}; \mathit{plant})^*$.

Some Intuition behind dL

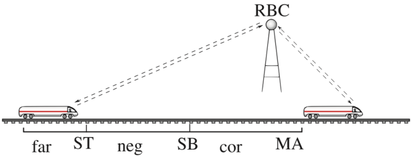
- Generally, want to reason about Embedded Systems (ES) or Cyber Physical Systems (CPS).
- Loosely speaking, an ES or CPS is built upon the integration of Computation (Controller) and Physical Systems (Plant). Examples include autopilot, ABS (Anti Lock Braking Systems), fly-by-wire (human interaction).
- Expressed in dL as an HP $\alpha \equiv (\mathit{ctrl}; \mathit{plant})^*$.
- Formulas of interest are of the form $\varphi \rightarrow [\alpha]\psi$.
- Formula above says that all states satisfying φ will always transition to states satisfying ψ, where α dictates the transition relation.

European Train Control System (ETCS)



- Discards traditional fixed segments of track with mutual exclusion. Instead uses decentralized Radio Block Controllers (RBCs)
- Trains dynamically issues Movement Authorization (MAs).
- At the end of MA, agent requests MA extension (negotation). If request is denied, train breaks before exiting old MA zone.

European Train Control System (ETCS)



- Train controller is responsible for staying in MA.
- Controller has to determine point SB (Start Braking). Before SB, train can move freely (maximizing throughput).
- Beyond SB (correction phase) start braking such that train remains in MA if RBC refuses extension.

European Train Control System (ETCS)

Say we have a train -

- with MA granted upto some track position m
- at position z, heading with initial speed v towards m
- point SB as safety distance s relative to m, i.e. $m - s = SB$

European Train Control System (ETCS)

Say we have a train -

- with MA granted upto some track position *m*
- at position z, heading with initial speed *v* towards *m*
- point SB as safety distance *s* relative to *m*, i.e. $m - s = SB$

The following formula expresses that the train remains in its MA under precondition ψ.

$$\begin{aligned} \psi &\rightarrow [(ctrl; drive)^*]\, z \leq m \\ \text{where } ctrl &\equiv (?m - z \leq s; a := -b) \cup (?m - z \geq s; a := A) \\ drive &\equiv \tau := 0; (z' = v, v' = a, \tau' = 1 \,\&\, v \geq 0 \wedge \tau \leq \varepsilon) \ . \end{aligned}$$

European Train Control System (ETCS)

Say we have a train -

- with MA granted upto some track position m
- at position z, heading with initial speed v towards m
- point SB as safety distance s relative to m, i.e. $m - s = SB$

European Train Control System (ETCS)

Say we have a train -

- with MA granted upto some track position *m*
- at position z, heading with initial speed *v* towards *m*
- point SB as safety distance *s* relative to *m*, i.e. $m - s = SB$

The following formula expresses that the train remains in its MA under precondition ψ.

$$\begin{aligned} \psi \;&\rightarrow\; [(ctrl; drive)^*]\, z \leq m \\ \text{where } ctrl \;&\equiv\; (?m - z \leq s; a := -b) \cup (?m - z \geq s; a := A) \\ drive \;&\equiv\; \tau := 0; (z' = v, v' = a, \tau' = 1 \,\&\, v \geq 0 \wedge \tau \leq \varepsilon) \enspace . \end{aligned}$$

European Train Control System (ETCS)

$$\begin{aligned}
\psi &\rightarrow [(ctrl; drive)^*]\, z \leq m \\
\text{where } ctrl &\equiv (?m - z \leq s; a := -b) \cup (?m - z \geq s; a := A) \\
drive &\equiv \tau := 0; (z' = v, v' = a, \tau' = 1 \,\&\, v \geq 0 \wedge \tau \leq \varepsilon) \ .
\end{aligned}$$

ECTS Safety Verification

In order to verify safety, we need

1. Use the calculus to analyze conditions of safety violation. Will provide some intuition about the safety precondition.
2. Use the calculus to verify safety, allowing exploration of the calculus' rules.

Preliminaries

- $\phi_{x_1}^{\theta_1}, \ldots, \phi_{x_n}^{\theta_n}$ denotes simultaneous substitution of x_i for θ_i in ϕ.
- Substitution must be "admissible". Given term t and substitution σ, variables of t or $\sigma(t)$ must not occur in the scope of a quantifier or modality binding. α conversion for renaming is assumed and used when needed.
- $\forall^{\alpha}\phi$ denotes the universal closure of ϕ w.r.t. all state variables bound by α. Note quantification over state variables is definable via auxillary logical variables as $\forall X[x := X]\phi$
- The calculus consists of propositional (P), first order (F), global (G), dynamic modality (D) rules.
- The calculus has 32 rules.

Discrete Dynamics

$$\frac{\phi_{x_1}^{\theta_1} \cdots {}_{x_n}^{\theta_n}}{\langle x_1 := \theta_1, .., x_n := \theta_n \rangle \phi}$$

Continuous Dynamics

$$\frac{\forall t{\geq}0 \left((\forall 0{\leq}\tilde{t}{\leq}t \ \langle \mathcal{S}_{\tilde{t}} \rangle \chi) \rightarrow \langle \mathcal{S}_t \rangle \phi \right)}{[x_1' = \theta_1, .., x_n' = \theta_n \ \& \ \chi]\phi}$$

- t and $\tilde{t}$ are fresh logical variables. $\langle S_t \rangle$ is the jump set $\langle x_1 := y_1(t), \ldots x_n := y_n(t) \rangle$, with simultaneous solutions $y_1, \ldots, y_n$ of the respective differential equations with constant symbols x_i as symbolic initial values.

F Rules

$$\frac{\vdash \phi(X)}{\vdash \exists x\, \phi(x)}$$

X is a new logical variable.

$$\frac{\vdash \phi(s(X_1, .., X_n))}{\vdash \forall x\, \phi(x)}$$

$X_1, \ldots, X_n$ are all free logical variables of $\forall x \phi(x)$

F rules are inspired from Tableuax methods for FOL, introducing decision procedures into the proof system itself.

F Rules

$$\frac{\vdash \mathrm{QE}(\forall X\ (\Phi(X) \vdash \Psi(X)))}{\Phi(s(X_1, .. , X_n)) \vdash \Psi(s(X_1, .. , X_n))}$$

X is a new logical variable

$$\frac{\vdash \mathrm{QE}(\exists X\ \bigwedge_i (\Phi_i \vdash \Psi_i))}{\Phi_1 \vdash \Psi_1 \quad \ldots \quad \Phi_n \vdash \Psi_n}$$

Logical variable X only occurs in $\Phi_i \vdash \Psi_i$. The intution is variable X was introduced via skolemization (at some point in the proof tree). The rule reintroduces existentiality over X, allowing for Quantifier Elimination (QE) over the existentially quantified terms.

G Rules

$$\frac{\vdash \forall^{\alpha}(\phi \rightarrow [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi}$$

$$\frac{\vdash \forall^{\alpha}(\phi \rightarrow \psi)}{[\alpha]\phi \vdash [\alpha]\psi}$$

European Train Control System (ETCS)

$$
\begin{aligned}
\psi \;&\rightarrow\; [(\mathit{ctrl}; \mathit{drive})^*]\, z \le m \\
\text{where } \mathit{ctrl} \;&\equiv\; (?m - z \le s; a := -b) \cup (?m - z \ge s; a := A) \\
\mathit{drive} \;&\equiv\; \tau := 0; (z' = v, v' = a, \tau' = 1 \,\&\, v \ge 0 \wedge \tau \le \varepsilon)\ .
\end{aligned}
$$

Deduction modulo for MA violation in braking mode

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{v \ge 0, z < m \vdash v^2 > 2b(m-z)}{\vdash v \ge 0 \wedge z < m \rightarrow v^2 > 2b(m-z)}}{v \ge 0, z < m \vdash T \ge 0 \qquad \text{D9}\dfrac{v \ge 0, z < m \vdash -\frac{b}{2}T^2 + vT + z > m}{v \ge 0, z < m \vdash \langle z := -\frac{b}{2}T^2 + vT + z\rangle z > m}}}{v \ge 0, z < m \vdash T \ge 0 \wedge \langle z := -\frac{b}{2}T^2 + vT + z\rangle z > m}}{v \ge 0, z < m \vdash \exists t{\ge}0\, \langle z := -\frac{b}{2}t^2 + vt + z\rangle z > m}}{\dfrac{v \ge 0, z < m \vdash \langle z' = v, v' = -b\rangle z > m}{\vdash v \ge 0 \wedge z < m \rightarrow \langle z' = v, v' = -b\rangle z > m}}
$$

Deduction modulo for MA violation in braking mode

$$
\dfrac{\dfrac{v \geq 0, z < m \vdash v^2 > 2b(m-z)}{\vdash v \geq 0 \land z < m \to v^2 > 2b(m-z)}}{\dfrac{\dfrac{\dfrac{v \geq 0, z < m \vdash T \geq 0 \qquad {\scriptstyle \mathrm{D9}}\dfrac{v \geq 0, z < m \vdash -\frac{b}{2}T^2 + vT + z > m}{v \geq 0, z < m \vdash \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}}{v \geq 0, z < m \vdash T \geq 0 \land \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}}{v \geq 0, z < m \vdash \exists t{\geq}0\, \langle z := -\frac{b}{2}t^2 + vt + z \rangle z > m}}{\dfrac{v \geq 0, z < m \vdash \langle z' = v, v' = -b \rangle z > m}{\vdash v \geq 0 \land z < m \to \langle z' = v, v' = -b \rangle z > m}}}
$$

$$
\mathrm{QE}\left(\exists T\,((v \geq 0 \land z < m \to T \geq 0) \land (v \geq 0 \land z < m \to -\frac{b}{2}T^2 + vT + z > m))\right)
$$

$$
\equiv\ v \geq 0 \land z < m\ \to\ v^2 > 2b(m-z)\ .
$$

European Train Control System (ETCS) Safety

- Invariant $\phi \equiv v^2 \leq 2b(m - z) \wedge b > 0 \wedge A \geq 0$ can be deduced in the manner of finding the violation condition.
- The safety proof can then be completed by showing
  - Invariant holds initially $\psi \vdash \phi$
  - $\phi \rightarrow [(ctrl; drive)^*]\phi$ holds.
  - $\phi \rightarrow \varphi_{post}$. where $\varphi_{post \equiv (\leq m)}$.

- Verification of a Hybrid System still doesn't guarantee the physical system behaves as intended.
- Hybrid System involve interaction with real world physics, which can never be captured fully by any model.
- Runtime monitoring used to obtain real world compliance.

Runtime Monitoring (ModelPlex)

- ModelPlex (Mitsch et al.) used dL for runtime validation of systems verified in dL.
- Uses dL to obtain monitoring conditions over Reals (which can be checked via SMT solvers)

$$(\nu,\omega) \in \rho(\alpha^*)$$
$$\Updownarrow$$
$$(\nu,\omega) \models \langle\alpha^*\rangle\Upsilon^+$$
$$\Uparrow$$
$$(\nu,\omega) \models F(x,x^+)$$

- Generate sandbox controller with strong safety guarantees.

- dL provides Hybrid Programs - concise and succint notation for Hybrid Systems.

- dL provides Hybrid Programs - concise and succint notation for Hybrid Systems.
- A proof calculus suited for verification of CPS and Embedded Systems.

- dL provides Hybrid Programs - concise and succint notation for Hybrid Systems.
- A proof calculus suited for verification of CPS and Embedded Systems.
- Rich toolchain (Keymaera), ModelPlex built on top of dL.