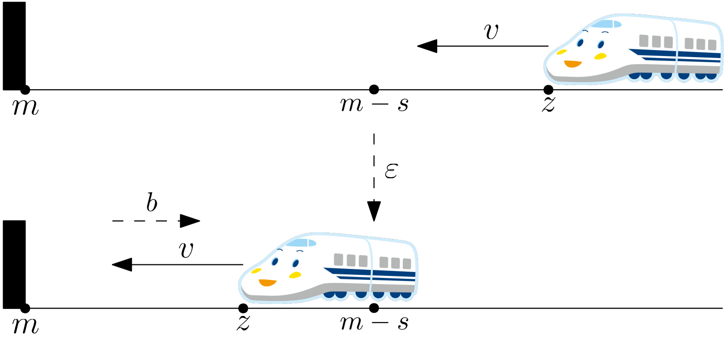
A Nonstandard Temporal Logic for Continuous-Time Verification

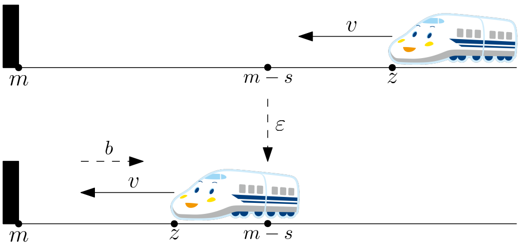
CSCAT 2025

Kei Imada and Justin Hsu
Cornell University, Computer Science

Nozomi 31 (tetsumaru.jp)


v
m
$m - s$
z
ε
b
v
m
z
$m - s$

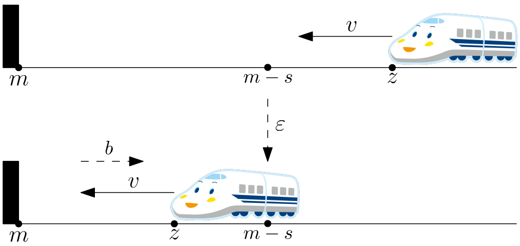
Train control system [**Po8**]



(Liveness) Will the train eventually stop?

(Safety) Will the train always stay behind the barrier?

Hybrid systems



Discrete dynamics — safety distance checks

Continuous evolutions — train position and velocity

Hybrid automata [**ACHH93**]

Discrete dynamics	finite automata
Continuous evolutions	differential equations

Differential dynamic logic [**P07; P08; P12; P17**]

Discrete dynamics	Kleene algebra with tests
Continuous evolutions	differential equations

Hybrid automata [**ACHH93**]

Discrete dynamics	finite automata
Continuous evolutions	**differential equations**

Differential dynamic logic [**P07; P08; P12; P17**]

Discrete dynamics	Kleene algebra with tests
Continuous evolutions	**differential equations**

Discrete approaches

Programs

Automata

Kleene algebra with tests

Hoare logic

Separation logic

Relational logic

Linear temporal logic

$\vdots$

Continuous approaches

Differential equations

Propositional logic

Barrier certificates

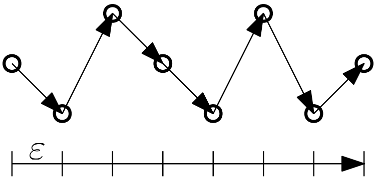
Lyapunov functions

$\vdots$

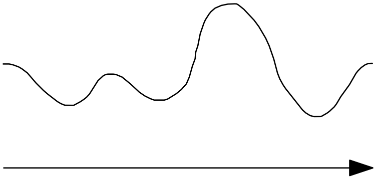
Discrete approaches		Continuous approaches
Programs		Differential equations
Automata		Propositional logic
Kleene algebra with tests	$\xrightarrow{?}$	Barrier certificates
Hoare logic		Lyapunov functions
Separation logic		$\vdots$
Relational logic		
Linear temporal logic		
$\vdots$		

Challenge: There is no smallest positive real-valued timestep

Discrete-time traces change states each tick



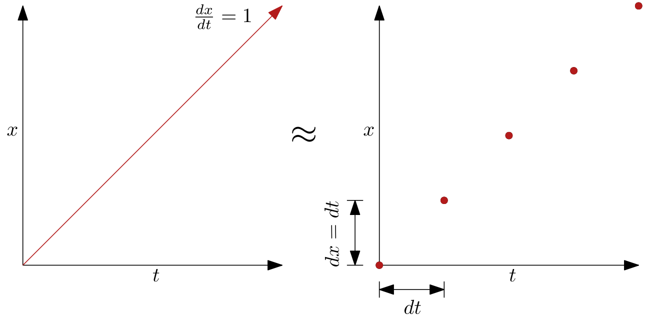
Continuous-time traces change states continuously over $t \in \mathbf{R}^{\geq 0}$



Continuous-time traces are discrete-time traces with infinitesimal timesteps

$\mathbb{R}$ becomes ${}^{\bullet}\mathbb{R}$ with infinitesimals

Transfer definitions and theorems to ${}^{\bullet}\mathbb{R}$



Agenda

What we will be talking about

Trace semantics of WHILE^{dt} and the smoothing operation

Continuous LTL (CLTL) and a preservation theorem

Verify the safety property of the train control system

Agenda

What we will (and won't) be talking about

Trace semantics of $\textsc{While}^{dt}$ and the smoothing operation

Continuous LTL (CLTL) and a preservation theorem

Verify the safety property of the train control system

Coagenda: category theory

Categorical interpretations are welcome!

Kei Imada

Justin Hsu

While$^{\mathtt{dt}}$

A While language with infinitesimals

Suenaga and Hasuo [**SH11**]

Programming with Infinitesimals: A While-Language for Hybrid System Modeling*

Kohei Suenaga[1] and Ichiro Hasuo[2]

[1] JSPS Research Fellow, Kyoto University, Japan
[2] University of Tokyo, Japan

$$\textbf{AExp} \ni a \triangleq x \mid r \mid a_1 \ \mathtt{aop} \ a_2 \mid \mathtt{dt} \mid \infty$$

$$\textbf{BExp} \ni b \triangleq \mathtt{true} \mid \mathtt{false} \cdots$$

$$\textbf{Cmd} \ni c \triangleq x \coloneqq a \mid c_1 ; c_2 \mid \mathtt{while} \ b \ \mathtt{do} \ c \cdots$$

Train control system in While^{dt}

Suenaga and Hasuo [**SH11**], modified for presentation

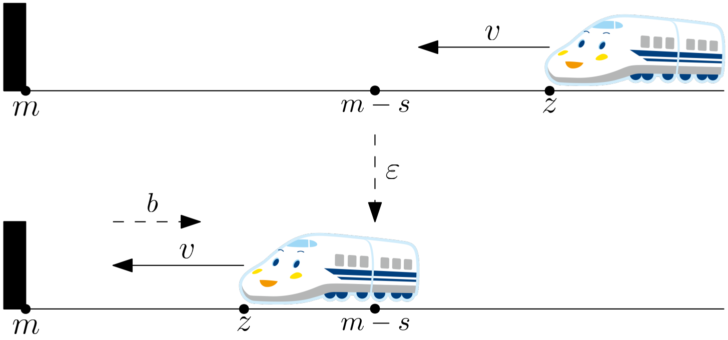
```
while (v > 0) {
  if (decel) {
    v := v - (b * dt)
  };
  z := z + v * dt
  // determine decel
}
```

decel: deceleration flag

v: velocity

b: deceleration

z: position



While$^{\mathrm{dt}}$ Trace Semantics

`dt` is a variable ranging over $\mathbf{R}^+$

$$a \in \textbf{AExp} \qquad b \in \textbf{BExp} \qquad c \in \textbf{Cmd}$$

$$[\![a]\!] : \mathbf{R}^+ \to \Sigma \to \mathbf{R} \qquad [\![b]\!] : \mathbf{R}^+ \to \Sigma \to \{\top, \bot\} \qquad [\![c]\!] : \mathbf{R}^+ \to \Sigma \to \Sigma$$

$$[\![p_{\mathsf{train}}]\!](0.01) = [\![q_{\mathsf{train}}]\!](*)$$

p_{train}

```
if (decel) {
  v := v - (b * dt)
};
z := z + v * dt
```

q_{train}

```
if (decel) {
  v := v - (b * 0.01)
};
z := z + v * 0.01
```

“dt seconds pass every loop iteration”

```
while (v > 0) {
  if (decel) {
    v := v - (b * dt)
  };
  z := z + v * dt
  // determine decel
}
```

$$\tau_{(w,\sigma_0)} : \mathbf{R}^+ \to \mathbf{N} \to \Sigma$$

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^+$	timestep assigned to `dt`
$\mathbf{N}$	number of loop iterations
Σ	program state

$$\tau_{(w,\sigma_0)} : \mathbf{R}^+ \to \mathbf{N} \to \Sigma \qquad \mathsf{sm}\Big(\tau_{(w,\sigma_0)}\Big) : \mathbf{R}^{\geq 0} \to \Sigma$$

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^+$	timestep assigned to `dt`
$\mathbf{N}$	number of loop iterations
Σ	program state

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^{\geq 0}$	time passed (t)
Σ	program state

$$\tau_{(w,\sigma_0)} : \mathbf{R}^+ \to \mathbf{N} \to \Sigma \qquad \mathsf{sm}\big(\tau_{(w,\sigma_0)}\big) : \mathbf{R}^{\geq 0} \to \Sigma$$

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^+$	timestep assigned to `dt`
$\mathbf{N}$	number of loop iterations
Σ	program state

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^{\geq 0}$	time passed (t)
Σ	program state

$$\tau_{(w,\sigma_0)} : \mathbf{R}^+ \to \mathbf{N} \to \Sigma \qquad \mathrm{sm}\left(\tau_{(w,\sigma_0)}\right) : \mathbf{R}^{\geq 0} \to \Sigma$$

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^+$	timestep assigned to `dt`
$\mathbf{N}$	number of loop iterations
Σ	program state

w	`while` b `do` p
σ_0	initial program state
$\mathbf{R}^{\geq 0}$	time passed (t)
Σ	program state

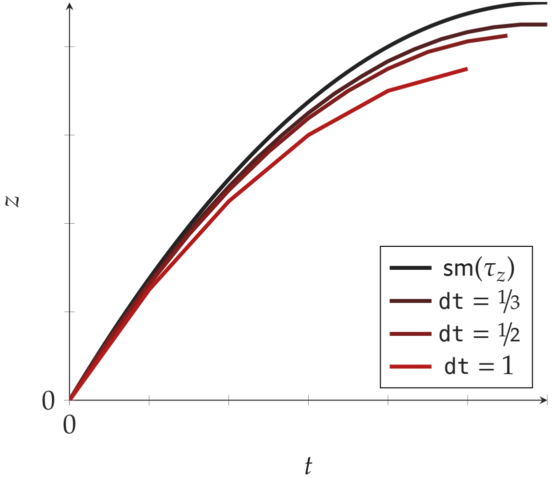
$$\xrightarrow{\text{Nonstandard analysis}}$$

$$
\begin{tikzcd}
{}^\bullet\tau : {}^\bullet\mathbf{R}^+ \to {}^\bullet\mathbf{N} \to {}^\bullet\Sigma \arrow[r, "\varepsilon"] & {}^\bullet\tau(\varepsilon) : {}^\bullet\mathbf{N} \to {}^\bullet\Sigma \arrow[d, "\mathsf{sh}"] \\
\tau : \mathbf{R}^+ \to \mathbf{N} \to \Sigma \arrow[u, "\bullet"] & \mathsf{sm}(\tau) : \mathbf{R}^{\geq 0} \to \Sigma
\end{tikzcd}
$$

$$\begin{tikzcd}
{}^{\bullet}\tau : {}^{\bullet}\mathbf{R}^{+} \to {}^{\bullet}\mathbf{N} \to {}^{\bullet}\Sigma \arrow[r, "\varepsilon"] & {}^{\bullet}\tau(\varepsilon) : {}^{\bullet}\mathbf{N} \to {}^{\bullet}\Sigma \arrow[d, "\mathsf{sh}"] \\
\tau : \mathbf{R}^{+} \to \mathbf{N} \to \Sigma \arrow[u, "\bullet"] \arrow[r, "\mathsf{sm}"] & \mathsf{sm}(\tau) : \mathbf{R}^{\geq 0} \to \Sigma
\end{tikzcd}$$

Example: braking train

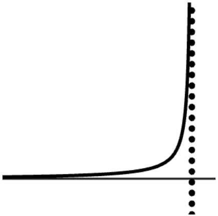
```
while (v > 0) {
  v := v - (b * dt);
  z := z + v * dt
}
```



Restrictions to While^{dt}

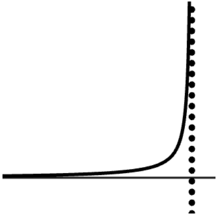
```
while (true) {
  x += x*x*dt
}
```

$$\frac{dx}{dt} = x^2 \text{ with solution } x(t) = \frac{1}{{x_0}^{-1} - t}$$



```
while (true) {
  x += x*x*dt
}
```

$$\frac{dx}{dt} = x^2 \text{ with solution } x(t) = \frac{1}{{x_0}^{-1} - t}$$



Problem: $\mathrm{sm}(\tau) : \mathbf{R}^{\geq 0} \to \Sigma$ no longer total!

Separate variables into **Cont** $\subseteq$ **Var**

$x_i \in$ **Cont** ranges over a proper metric space

$$
\begin{aligned}
\textbf{LExp} \ni f \quad & \triangleq x \ (x \in \textbf{Cont}) \mid r \ (r \in \mathbf{R}) \mid \texttt{dt} \mid r \cdot f \ (r \in \mathbf{R}) \mid R(f_1, \ldots, f_n) \\
\textbf{AExp} \ni a \quad & \triangleq x \mid r \mid a_1 \ \texttt{aop} \ a_2 \mid \cdots \\
\textbf{BExp} \ni b \quad & \triangleq \texttt{true} \mid \texttt{false} \mid \cdots \\
\textbf{Cmd} \ni p \quad & \triangleq x := a \ (x \in \textbf{Var} \setminus \textbf{Cont}) \mid x \mathrel{+}= f \cdot \texttt{dt} \ (x \in \textbf{Cont}) \mid \cdots \\
\textbf{Prog} \ni w \quad & \triangleq \texttt{while} \ b \ \texttt{do} \ p
\end{aligned}
$$

Separate variables into **Cont** $\subseteq$ **Var**

$x_i \in$ **Cont** ranges over a proper metric space

$$
\begin{aligned}
\textbf{LExp} \ni f &\triangleq x \ (x \in \textbf{Cont}) \mid r \ (r \in \mathbf{R}) \mid \mathtt{dt} \mid r \cdot f \ (r \in \mathbf{R}) \mid R(f_1, \dots, f_n) \\
\textbf{AExp} \ni a &\triangleq x \mid r \mid a_1 \ \mathtt{aop} \ a_2 \mid \cdots \\
\textbf{BExp} \ni b &\triangleq \mathtt{true} \mid \mathtt{false} \mid \cdots \\
\textbf{Cmd} \ni p &\triangleq x := a \ (x \in \textbf{Var} \setminus \textbf{Cont}) \mid x \mathrel{+}= f \cdot \mathtt{dt} \ (x \in \textbf{Cont}) \mid \cdots \\
\textbf{Prog} \ni w &\triangleq \mathtt{while} \ b \ \mathtt{do} \ p
\end{aligned}
$$

Theorem Let $w \in$ **Prog**. Then $\mathrm{sm}\left(\tau_{(w,\sigma_0)}\right) : \mathbf{R}^{\geq 0} \to \Sigma_{\textbf{Cont}}$ is total and continuous.

WHILE^{dt}

```
while (v > 0) {
  if (decel) {
    v := v - (b * dt)
  };
  z := z + v * dt;
  // determine decel
}
```

Continuous WHILE^{dt}

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  // determine decel
}
```

Continuous LTL

(Liveness) The train will eventually stop

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash \mathsf{F}[v = 0]$$

(Safety) The train will always stay behind the barrier

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash \mathsf{G}[z \leq m]$$

(Liveness) The train will eventually stop

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash \mathsf{F}[v = 0]$$

(Safety) The train will always stay behind the barrier

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash \mathsf{G}[z \leq m]$$

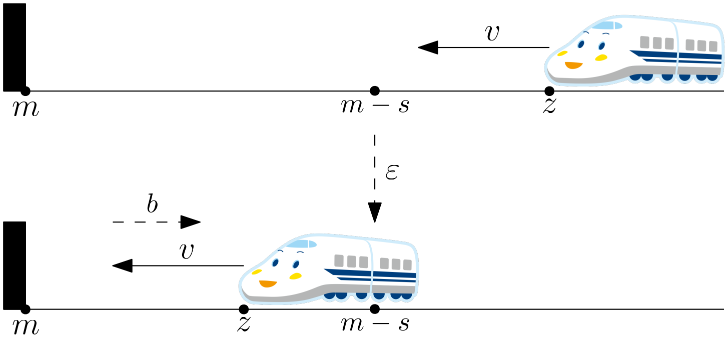
Using discrete methods on τ_{train}

$$\tau \vDash_d \Phi \text{ implies } \mathrm{sm}(\tau) \vDash_c \Phi$$

$$
\begin{tikzcd}
{}^\bullet\tau\,({}^\bullet\vDash_d)\,{}^\bullet\Phi \arrow[rr, Rightarrow, "\varepsilon"] \arrow[d, Leftrightarrow, "\bullet"'] & & {}^\bullet\tau(\varepsilon)\,({}^\bullet\vDash_d)\,{}^\bullet\Phi \arrow[d, Rightarrow, "\mathsf{sh}"] \\
\tau \vDash_d \Phi & & \mathsf{sm}(\tau) \vDash_c \Phi
\end{tikzcd}
$$

$$
\begin{tikzcd}
{}^{\bullet}\tau\ ({}^{\bullet}\vDash_d)\ {}^{\bullet}\Phi \arrow[r, Rightarrow, "\varepsilon"] \arrow[d, Leftrightarrow, "\bullet"'] & {}^{\bullet}\tau(\varepsilon)\ ({}^{\bullet}\vDash_d)\ {}^{\bullet}\Phi \arrow[d, Rightarrow, "\mathsf{sh}"] \\
\tau \vDash_d \Phi \arrow[r, Rightarrow] & \mathsf{sm}(\tau) \vDash_c \Phi
\end{tikzcd}
$$

Train control system verification




```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash_c \mathsf{F}[v = 0]$$

$$\mathsf{sm}(\tau_{\mathsf{train}}) \vDash_c \mathsf{G}[z \leq m]$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\tau_{\text{train}} \vDash_d \mathsf{F}[v = 0]$$
$$\mathsf{sm}(\tau_{\text{train}}) \vDash_c \mathsf{F}[v = 0]$$

$$\tau_{\text{train}} \vDash_d \mathsf{G}[z \leq m]$$
$$\mathsf{sm}(\tau_{\text{train}}) \vDash_c \mathsf{G}[z \leq m]$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

Decrementing functions to show

$$\tau_{\text{train}} \vDash_d \mathsf{F}[v = 0]$$

$$\mathsf{sm}(\tau_{\text{train}}) \vDash_c \mathsf{F}[v = 0]$$

Loop invariants to show

$$\tau_{\text{train}} \vDash_d \mathsf{G}[z \leq m]$$

$$\mathsf{sm}(\tau_{\text{train}}) \vDash_c \mathsf{G}[z \leq m]$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\dfrac{\vdash \left\{\varphi \wedge b\right\} p \left\{\varphi\right\}}{\texttt{while } b \texttt{ do } p \vdash \mathsf{G}\,\varphi}\ \textsc{Global}$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\varphi_{\text{train}} \triangleq \varphi_t \wedge \varphi_0$$

$$\varphi_t \triangleq s > 0 \wedge b > 0 \wedge \mathsf{ep} > 0$$

$$\varphi_0 \triangleq v^2 < 2b(m - z - v \cdot \mathsf{ep})$$

$$\frac{\vdash \left\{\varphi_{\text{train}} \wedge [v > 0]\right\} p \left\{\varphi_{\text{train}}\right\}}{c_{\text{train}} \vdash \mathsf{G}\, \varphi_{\text{train}}} \textsc{Global}$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\varphi_{\text{train}} \triangleq \varphi_t \wedge \varphi_0$$

$$\varphi_t \triangleq s > 0 \wedge b > 0 \wedge \mathsf{ep} > 0$$

$$\varphi_0 \triangleq v^2 < 2b(m - z - v \cdot \mathsf{ep})$$

$$\frac{\vdash \left\{\varphi_{\text{train}} \wedge [v > 0]\right\} p \left\{\varphi_{\text{train}}\right\}}{c_{\text{train}} \vdash \mathsf{G}\, \varphi_{\text{train}}} \ \textsc{Global}$$

φ_{train} implies $z \leq m$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\varphi_{\text{train}} \triangleq \varphi_t \wedge \varphi_0$$

$$\varphi_t \triangleq s > 0 \wedge b > 0 \wedge \mathsf{ep} > 0$$

$$\varphi_0 \triangleq v^2 < 2b(m - z - v \cdot \mathsf{ep})$$

$$\frac{\vdash \left\{\varphi_{\text{train}} \wedge [v > 0]\right\} p \left\{\varphi_{\text{train}}\right\}}{c_{\text{train}} \vdash \mathsf{G}\, \varphi_{\text{train}}} \textsc{Global}$$

φ_{train} implies $z \leq m$

$$c_{\text{train}} \vdash \mathsf{G}[z \leq m]$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\varphi_{\text{train}} \triangleq \varphi_t \wedge \varphi_0$$

$$\varphi_t \triangleq s > 0 \wedge b > 0 \wedge \mathsf{ep} > 0$$

$$\varphi_0 \triangleq v^2 < 2b(m - z - v \cdot \mathsf{ep})$$

$$\frac{\vdash \left\{\varphi_{\text{train}} \wedge [v > 0]\right\} p \left\{\varphi_{\text{train}}\right\}}{c_{\text{train}} \vdash \mathsf{G}\, \varphi_{\text{train}}} \textsc{Global}$$

φ_{train} implies $z \leq m$

$$c_{\text{train}} \vdash \mathsf{G}[z \leq m]$$

$$c_{\text{train}} \vDash \mathsf{G}[z \leq m]$$

```
cont b, v, z;
while (v > 0) {
  if (decel) {
    v += -b * dt
  };
  z += v * dt;
  timer += dt;
  if (timer >= ep) {
    timer := 0;
    if (m - s < z) {
      decel := true
    }
  }
}
```

$$\varphi_{\text{train}} \triangleq \varphi_t \wedge \varphi_0$$

$$\varphi_t \triangleq s > 0 \wedge b > 0 \wedge \mathsf{ep} > 0$$

$$\varphi_0 \triangleq v^2 < 2b(m - z - v \cdot \mathsf{ep})$$

$$\frac{\vdash \left\{\varphi_{\text{train}} \wedge [v > 0]\right\} p \left\{\varphi_{\text{train}}\right\}}{c_{\text{train}} \vdash \mathsf{G}\, \varphi_{\text{train}}} \ \textsc{Global}$$

φ_{train} implies $z \leq m$

$$c_{\text{train}} \vdash \mathsf{G}[z \leq m]$$

$$c_{\text{train}} \vDash \mathsf{G}[z \leq m]$$

The train will always stay behind the barrier!

Today's talk

A programming language Continuous WHILE^{dt}

Continuous-time trace semantics using the smoothing operation

Continuous LTL and a preservation theorem

Loop invariants to verify safety properties of continuous-time traces

Decrementing functions to verify liveness properties

Verification of other classical hybrid systems

An implementation that verifies continuous-time properties

Thank you for listening!

Kei Imada

kei [at] cs [dot] cornell [dot] edu

keikun555.github.io