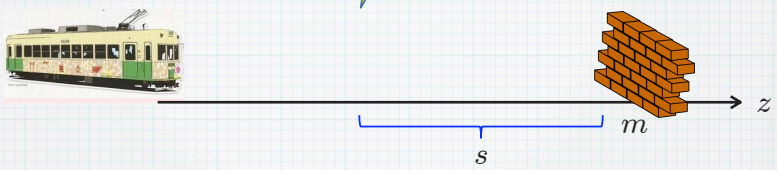
超準解析による
物理情報システムの形式検証

蓮尾 一郎
東京大学 大学院情報理工学系研究科 コンピュータ科学専攻・
理学部 情報科学科
http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro
東京大学 THE UNIVERSITY OF TOKYO

まず最初に．．．
デモを

問題



$$\dot{z} = v$$
$$\dot{v} = \begin{cases} a_0 & \text{if } m - z > s \\ -b & \text{if } m - z < s \end{cases}$$

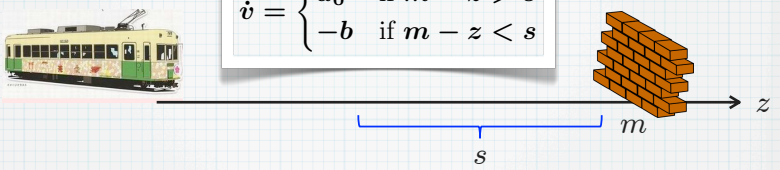
* 等加速度運動
* 安全距離 s で スイッチング

Q. カベにあたらずに止まるための十分条件は？

s: big enough
b: big enough
a_0: small enough
...

問題

$$\dot{z} = v$$
$$\dot{v} = \begin{cases} a_0 & \text{if } m - z > s \\ -b & \text{if } m - z < s \end{cases}$$



Q. カベにあたらずに止まるための十分条件は？

* 全自動で条件生成するツール
* 「正しさの証明」付き
* 入力は．．．

```
while (v > 0) {
  if m - z < s
    then a := -b
    else a := a0;
  t := 0;
  while (t < eps && v > 0) {
    z := z + v * dt;
    v := v + a * dt;
    t := t + dt }}

{z < m}
```

* プログラム？
* 微分方程式はどこ？

Hoare^{dt} Analyzer デモ

超準解析による 物理情報システムの形式検証

蓮尾 一郎

東京大学 大学院情報理工学系研究科 コンピュータ科学専攻・
理学部 情報科学科

http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro

東京大学 THE UNIVERSITY OF TOKYO

* 微小量（infinitesimal）を持つ，解析学の形式化
* Ultrafilter によるモデル
  ➜ 数理論理学（モデル理論）の成果！

超準解析による 物理情報システムの形式検証

形式検証の新たなターゲット

* 連続ダイナミクス（物理系）
* 離散ダイナミクス（デジタル制御）

数学（数理論理学）による品質保証

超準解析による物理情報システムの形式検証

アウトライン

* 形式検証とは？
  * Hoare 論理を例に
* ハイブリッド・システム
  * 離散＋連続
  * 物理情報システムの一側面
* 超準解析による移転
  * 離散的検証手法を，文字通りそのまま ハイブリッド・システムに適用

For You to Take Home

* Mathematical logic at work!

共同研究者

* 末永 幸平
  京都大学白眉センター
* 片岡 俊基，関根 大剛，木戸 肩吾
  東京大学コンピュータ科学専攻修士課程

論文

* [ICALP'11] K. Suenaga and I. Hasuo. Programming with infinitesimals: A while-language for hybrid system modeling. In L. Aceto, M. Henzinger and J. Sgall, editors, ICALP (2), vol. 6756 of Lect. Notes Comp. Sci., pp. 392–403. Springer, 2011.
* [CAV'12] I. Hasuo and K. Suenaga. Exercises in Nonstandard Static Analysis of hybrid systems. In P. Madhusudan and S.A. Seshia, editors, CAV, vol. 7358 of Lect. Notes Comp. Sci., pp. 462–478. Springer, 2012.
* [POPL'13] K. Suenaga, H. Sekine and I. Hasuo. Hyperstream processing systems: nonstandard modeling of continuous-time signals. In R. Giacobazzi and R. Cousot, editors, POPL, pp. 417–430. ACM, 2013.

論文・スライドなどはウェブページから

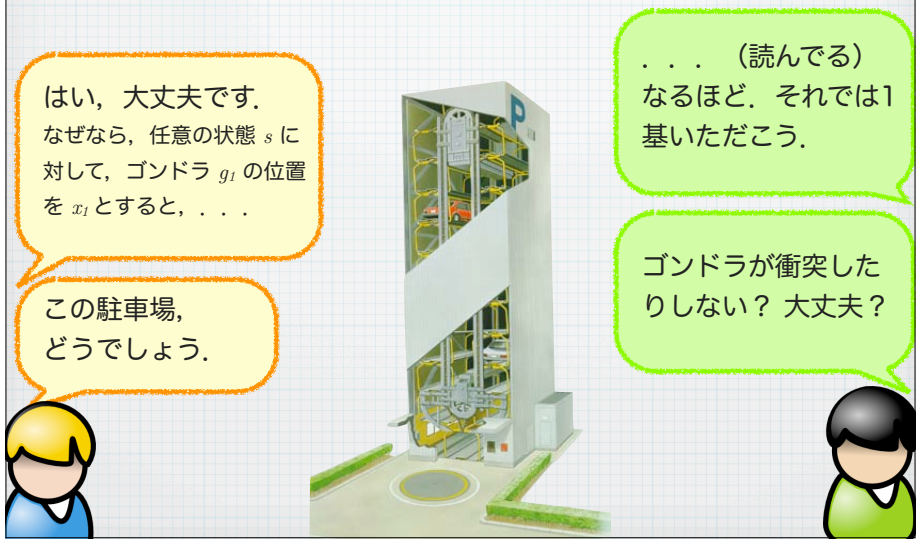
http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro/

1

形式検証とは
(Hoare 論理を例に)

証明で金もうけ

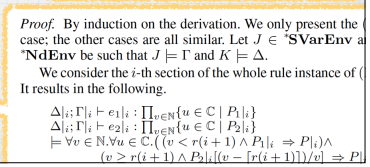
証明で金もうけ



形式検証

* システムが
* 仕様通り動作することを
* 数学的に証明すること．

仕様 (specification):
システムの満たすべき性質．
「ゴンドラがぶつからない」
など．


Proof. By induction on the derivation. We only present the case; the other cases are all similar. Let $J \in {}^{\forall}\mathbf{SVarEnv}$ a ${}^{\forall}\mathbf{NdEnv}$ be such that $J \models \Gamma$ and $K \models \Delta$.

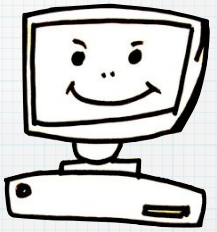
We consider the i-th section of the whole rule instance of (It results in the following.

$$\Delta|_i;\Gamma|_i \vdash e_1|_i : \prod_{v\in\mathbb{N}}\{u \in \mathbb{C} \mid P_1|_i\}$$
$$\Delta|_i;\Gamma|_i \vdash e_2|_i : \prod_{v\in\mathbb{N}}\{u \in \mathbb{C} \mid P_2|_i\}$$
$$\models \forall v \in \mathbb{N}.\forall u \in \mathbb{C}.((v < r(i+1) \wedge P_1|_i \Rightarrow P|_i) \wedge (v \geq r(i+1) \wedge P_2|_i[(v - \lceil r(i+1)\rceil)/v] \Rightarrow P|$$

形式検証
Formal Verification

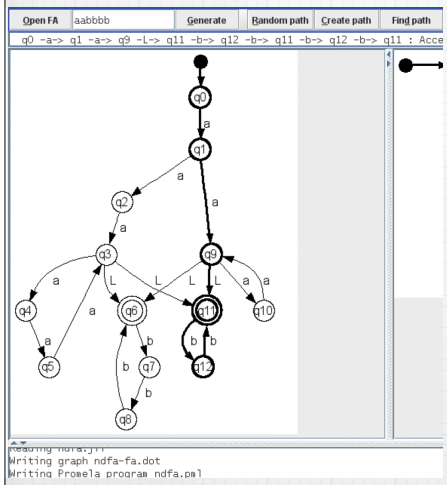
* システム検証 (system verification), 形式手法 (formal methods), ...
* もはや研究者の toy ではない
  * デバイスドライバの自動検証 (Microsoft)
  * 航空機制御ソフトウェアの検証 (Airbus)
  * ...

システム検証



* つまらなくて，間違いやすい証明
  * ➜ 計算機による支援，できれば自動化

モデル検査

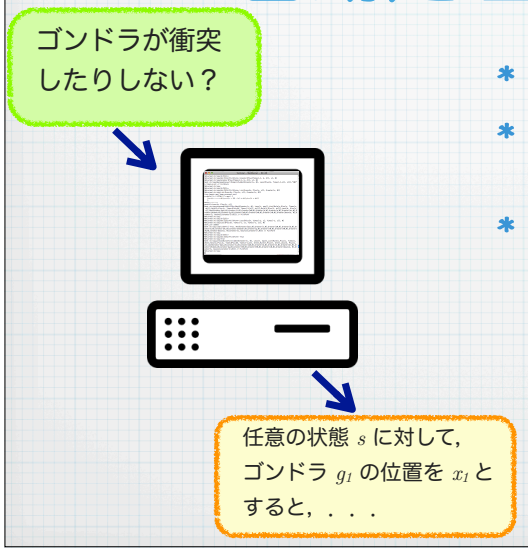


M. Ben-Ari, ACM Inroads, 2010

* 全状態を総当りでチェック
* SPIN, PRISM, Uppaal, MCRL2, ...

  \+ 全自動
  − 状態数爆発
  − 「∀サイズ」の検証

自動定理証明



* 証明を全自動で書く！
* 応用分野をはっきり決めて，そこに特化
* KeY, KeYmaera, Hoare^{dt} Analyzer, ...

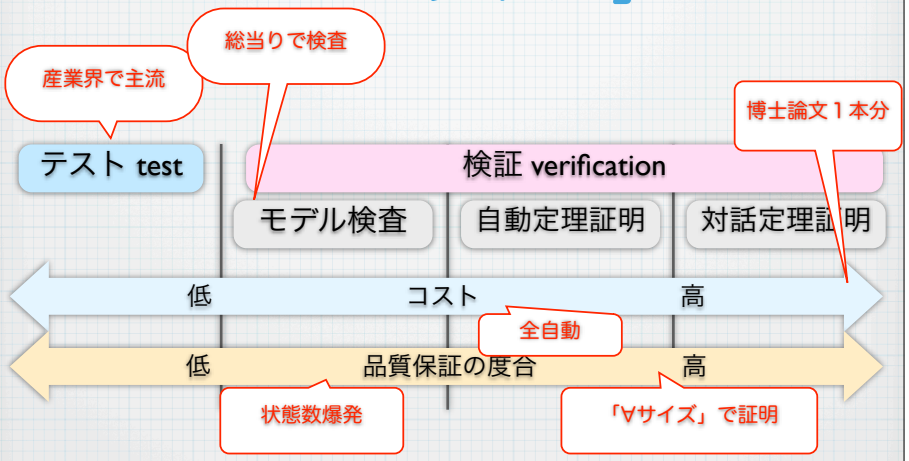
  \+ 全自動
  \+ 「∀サイズ」の検証
  − スケーラビリティ

対話型定理証明

* 人間が証明をコンピュータで書く.
* Coq, Agda, PVS, ACL2, ...

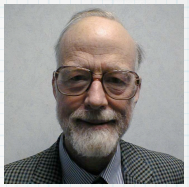
\+ 「∀サイズ」の検証
\+ フレキシビリティ
− 人的コスト（莫大！）

システム品質保証の「スペクトル」



Hoare 論理による形式検証

Hoare 論理



Sir Antony Hoare
(1934.1.11-)
Microsoft Research, Cambridge

* [Hoare, 1969]
* 「システム」 = （命令形）while プログラム
  「仕様」= precondition & postcondition
* Hoare triple を導いていく体系

{A} P {B}

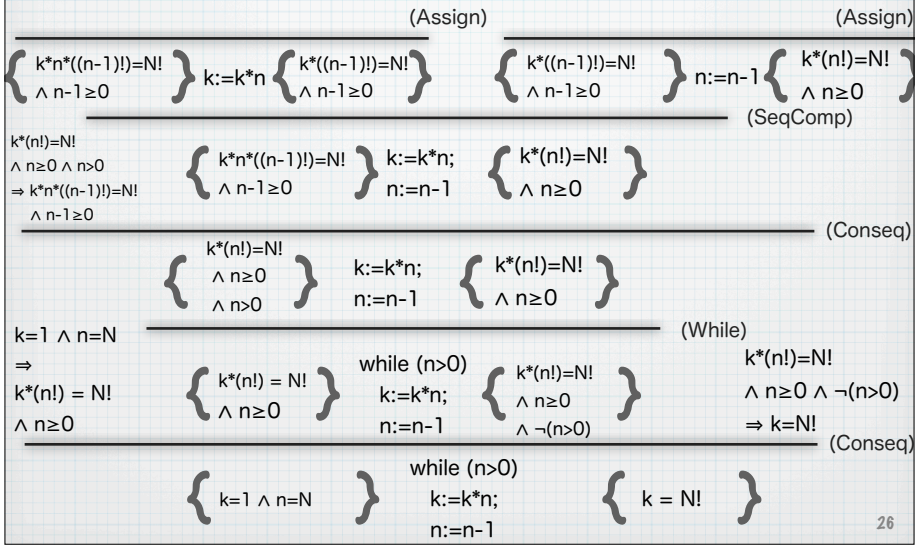
例：{ n=2 } n:=n+1 { n=3 }

実行前に成り立つ性質 “precondition”
プログラム
実行後に成り立つ性質 “postcondition”

（真である）
Hoare triple の例

$\{ k=1 \wedge n=N \}$ while (n>0) k:=k*n; n:=n-1 $\{ k = N! \}$

Hoare 論理による導出の例



Hoare 論理の「材料」

* プログラム意味論
  * プログラム ＝ メモリ状態の変換
* メモリ状態の性質を記述するための assertion language
* Hoare triple を導くための 導出規則 （ルール）, soundness

プログラム意味論

「意味論」とは？

- プログラムの「意味」は何か
  - 正確な答え：
    「実行した際の MacBook 内部の電圧変化」
  - ➜ 細かすぎて「使えない」

```
n := N;
k := 1;
while (n > 0) {
  k := k*n;
  n := n-1;
}
```

- ここではメモリ状態を用いる
  - プログラミング言語による
    （命令型言語だからメモリ状態を使う．たとえば関数型言語ならば関数として意味をつける）

メモリ状態

- 変数と値の対応の表のこと．

$$\begin{bmatrix} \mathrm{x} & \mapsto & 2 \\ \mathrm{y} & \mapsto & 13 \\ & \vdots & \end{bmatrix}$$

- 数学的には： 関数

$$\sigma : \mathbf{Var} \longrightarrow \mathbb{Z}$$

プログラムの意味

- メモリ状態の変換として
  - つまり，関数 $\mathbf{MSt} \longrightarrow \mathbf{MSt} \cup \{\bot\}$

$$[\![\mathrm{x} := a]\!] : \quad \sigma \quad \longmapsto \quad \sigma\left[\mathrm{x} \mapsto [\![a]\!]\sigma\right]$$

プログラム x:=a の「意味」

メモリ状態，たとえば

$$\begin{bmatrix} \mathrm{x} & \mapsto & 2 \\ \mathrm{y} & \mapsto & 13 \\ & \vdots & \end{bmatrix}$$

アップデートされたメモリ状態．x の値を，a を σ のもとで計算した値（たとえば $[\![\mathrm{y}+1]\!]\sigma = 14$ とか）にアップデート

x は変数，a は「数の表現」（たとえば y+1）

プログラムの意味

- メモリ状態の変換として
  - つまり，関数 $\mathbf{MSt} \longrightarrow \mathbf{MSt} \cup \{\bot\}$

$$[\![P_1; P_2]\!] : \quad \sigma \quad \longmapsto \quad [\![P_2]\!]([\![P_1]\!]\sigma)$$

まず P_1 によって変換

次に P_2 によって変換

P_1，P_2 はプログラム

プログラムの意味

* メモリ状態の変換として
  * つまり，関数 $\mathbf{MSt} \longrightarrow \mathbf{MSt} \cup \{\bot\}$

$$[\![\texttt{if}\ b\ \texttt{then}\ P_1\ \texttt{else}\ P_2]\!] : \quad \sigma \longmapsto \begin{cases} [\![P_1]\!]\sigma & \text{if } [\![b]\!]\sigma \text{ is true} \\ [\![P_2]\!]\sigma & \text{if } [\![b]\!]\sigma \text{ is false} \end{cases}$$

P_1，P_2 はプログラム
b は「真偽表現」
（x > 0 とか）

プログラムの意味

「停止しない」

* メモリ状態の変換として
  * つまり，関数 $\mathbf{MSt} \longrightarrow \mathbf{MSt} \cup \{\bot\}$

$$[\![\texttt{while}\ b\ P]\!] : \quad \sigma \longmapsto ??$$

* 状態変換の繰り返し $[\![P]\!]^n\sigma$を考える：

$$\sigma,\ [\![P]\!]\sigma,\ [\![P]\!]([\![P]\!]\sigma),\ [\![P]\!]([\![P]\!]([\![P]\!]\sigma)),\ \ldots$$

  * ある時点で b が偽になれば，つまり $[\![b]\!]([\![P]\!]^n\sigma) = \text{false}$ なら，そうなるような最初の n に対する $[\![P]\!]^n\sigma$ を返す
  * b がずっと真であれば，⊥（未定義，非停止）

プログラムの意味論：まとめ

「停止しない」

* メモリ状態の変換として
  * つまり，関数 $\mathbf{MSt} \longrightarrow \mathbf{MSt} \cup \{\bot\}$

$$[\![\mathtt{x} := a]\!] : \quad \sigma \longmapsto \sigma[\,\mathtt{x} \mapsto [\![a]\!]\sigma\,]$$

$$[\![P_1; P_2]\!] : \quad \sigma \longmapsto [\![P_2]\!]([\![P_1]\!]\sigma)$$

$$[\![\texttt{if}\ b\ \texttt{then}\ P_1\ \texttt{else}\ P_2]\!] : \quad \sigma \longmapsto \begin{cases} [\![P_1]\!]\sigma & \text{if } [\![b]\!]\sigma \text{ is true} \\ [\![P_2]\!]\sigma & \text{if } [\![b]\!]\sigma \text{ is false} \end{cases}$$

$$[\![\texttt{while}\ b\ P]\!] : \quad \sigma \longmapsto \ldots$$

ポイント：
* 数学的に厳密な定義
* 要素還元的（大きなプログラムの意味は，その部品の意味から決まる）

Hoare 論理の「材料」

* プログラム意味論
  * プログラム ＝ メモリ状態の変換
* メモリ状態の性質を記述するための assertion language
* Hoare triple を導くための導出規則（ルール），soundness

Assertion Language

* Assertion: メモリ状態の性質を記述する論理式．例：
  * x = 5 ∧ y ≤ 3
  * ∃z. (x = 2 * z ∧ y = 3 * z)
* 算術のための一階述語論理を用いる．

$$\begin{array}{llll} \textbf{AExp} \ni & a & ::= & x \mid n \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \\ \textbf{Fml} \ni & A & ::= & \texttt{true} \mid \texttt{false} \mid A_1 \wedge A_2 \mid \neg A \mid a_1 < a_2 \mid \\ & & & \forall x \in \mathbb{N}.\, A \quad \text{where } x \in \textbf{Var} \end{array}$$

  * そうすると相対完全性も成り立つ
  * 自動定理証明器の実装では，命題論理に制限．
    （限量子除去 QE を適宜用いる）

Hoare 論理の「材料」

* プログラム意味論
  * プログラム ＝ メモリ状態の変換
* メモリ状態の性質を記述するための
  assertion language
* Hoare triple を導くための
  導出規則 （ルール），soundness

Hoare 論理の導出規則

$$\frac{}{\{A\}\ \texttt{skip}\ \{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{\,A[a/x]\,\}\ x := a\ \{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\ c_1\ \{C\} \quad \{C\}\ c_2\ \{B\}}{\{A\}\ c_1; c_2\ \{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\ c_1\ \{B\} \quad \{A \wedge \neg b\}\ c_2\ \{B\}}{\{A\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\ c\ \{B'\} \quad \models B' \Rightarrow B}{\{A\}\ c\ \{B\}}\ (\textsc{Conseq})$$

Hoare 論理の導出規則

$$\frac{\{A\}\ P_1\ \{C\} \quad \{C\}\ P_2\ \{B\}}{\{A\}\ P_1; P_2\ \{B\}}\ (\text{SeqComp})$$

* 例：

$$\dfrac{\dfrac{}{\{x\text{-}1=2\}\ y:=x\ \{y\text{-}1=2\}}\ (\text{Assign}) \quad \dfrac{}{\{y\text{-}1=2\}\ y:=y\text{-}1\ \{y=2\}}\ (\text{Assign})}{\{x\text{-}1=2\}\ y:=x;\ y:=y\text{-}1\ \{y=2\}}\ (\text{SeqComp})$$

Hoare 論理の導出規則

A はループ不変量！

$$\frac{\{A \wedge b\}\ P_1\ \{A\}}{\{A\}\ \text{while b}\ P_1\ \{A \wedge \neg b\}}\ \text{(While)}$$

ループを脱出した ➜
b は成立しないはず

* 例：

$$\frac{\{x \geq 0 \wedge x > 0\}\ x := x-1\ \{x \geq 0\}}{\{x \geq 0\}\ \text{while } x>0\ (\ x := x-1\ )\ \{x \geq 0 \wedge \neg(x>0)\}}\ \text{(While)}$$

Hoare 論理の健全性

$$\frac{}{\{A\}\,\texttt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{\,A[a/x]\,\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1;c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\texttt{while } b \texttt{ do } c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Thm. (Soundness)

$$\vdash \{A\}\,c\,\{B\} \implies \models \{A\}\,c\,\{B\}\ ,$$

where

$$\models \{A\}\,c\,\{B\} \overset{\text{def}}{\iff} \left[\begin{array}{l}\text{for each memory state } \sigma,\\ \sigma \models A \text{ implies } [\![c]\!](\sigma) \models B.\end{array}\right]$$

プログラム c の「意味」
（メモリ状態の変換として）

Hoare 論理による形式検証（まとめ）

* システム + 仕様 を Hoare triple
  $\{A\}\ P\ \{B\}$ として表現．真？
* Hoare 論理で，
  自動証明（証明検索）

$$\frac{}{\{A\}\,\texttt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{\,A[a/x]\,\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1;c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\texttt{while } b \texttt{ do } c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

超準解析による物理情報システムの形式検証

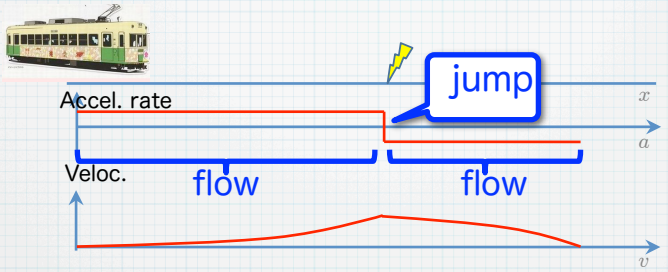
アウトライン

* 形式検証とは？
  * Hoare 論理を例に
* ハイブリッド・システム
  * 離散＋連続
  * 物理情報システムの一側面
* 超準解析による移転
  * 離散的検証手法を，文字通りそのまま
    ハイブリッド・システムに適用

2

ハイブリッド・システム

ハイブリッド・システム

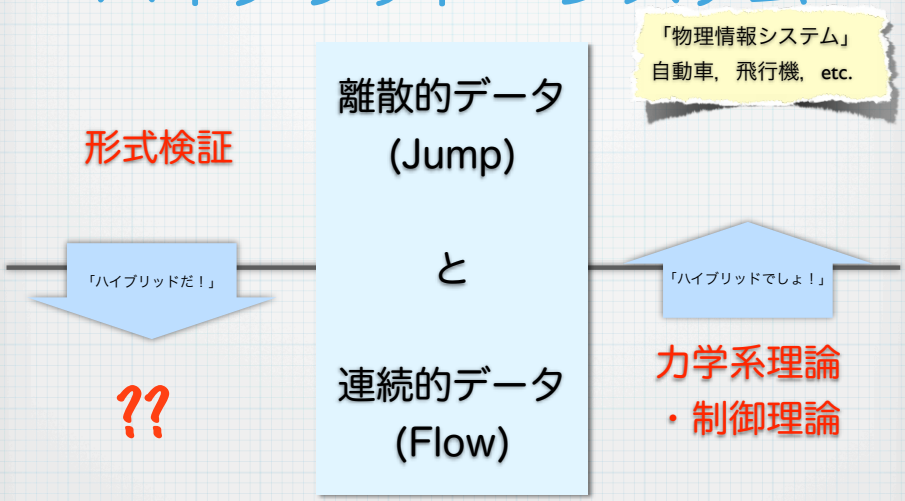


* Flow & jump
  * 物理系におけるデジタル制御
  * 物理情報システムの一側面

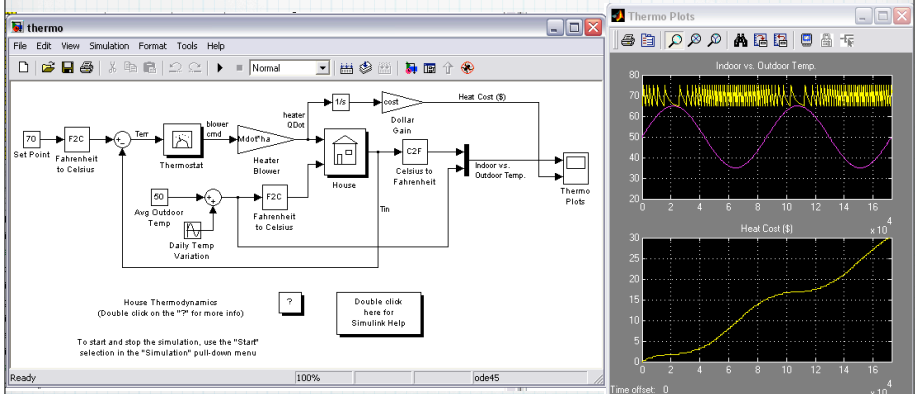
物理情報システム CPS: Cyber Physical System

* The term cyber-physical systems refers to the tight conjoining of and coordination between computational and physical resources.
(NSF Program Solicitation, NSF 08-611)
* 物理情報システム CPS:
  * Real-Time （実時間）
  * Embedded （組み込み）
  * Sensor Network
  * Hybrid System

ハイブリッド・システム



Matlab/Simulink

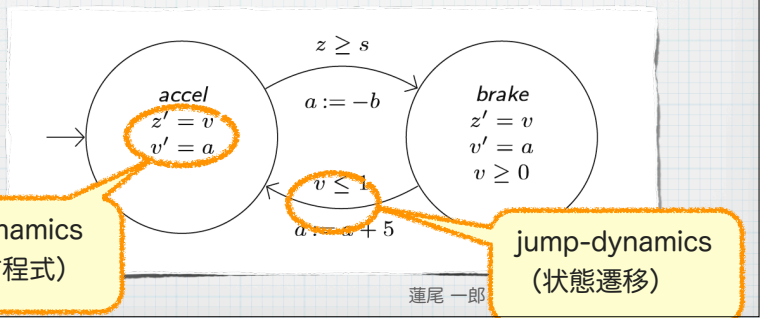


* 産業界でのデファクト・スタンダード
  （オープンソースの代替も多数）
* 用途： モデリング，シミュレーション（テスト），解析

システム検証：
離散から
ハイブリッドへ

離散からハイブリッドへ

* 既存研究： 微分方程式を書いちゃえ
* ハイブリッド・オートマトン [Alur]



離散からハイブリッドへ

* 既存研究： 微分方程式を書いちゃえ
* Differential Dynamic Logic [Platzer]

$$[\dot{x} = 1 \text{ while } x \leq 3]\varphi$$

離散からハイブリッドへ

* 既存研究： 微分方程式を書いちゃえ
  * 問題点： 微分方程式を書きたくない！
* 提案手法：

Flow を Jump に変換

  * 既存の離散的検証手法が
    何でも，文字通り移転できる（原理的には）
  * 数学的にスジよく，超準解析で

Flow を Jump に変換

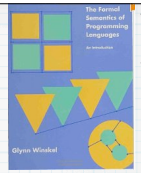
$$t := 0\ ;$$
$$\texttt{while}\ (t \leq 1)\ \texttt{do}\ \{$$
$$\quad t := t + \mathrm{dt}$$
$$\}$$

* Infinitesimal number dt
  * "Infinitely small" :
    $0 < \mathrm{dt} < r$ for **any** positive real r
* 実行後 t = 1?
* 超準解析

  Nonstandard analysis!
  [Robinson '60s]

Theoretical Framework

[Suenaga&H., ICALP'11]



The standard textbook [Winskel]

While^{dt}
Programming lang.

```
while (t<a) do {
  t:=t+1;
  if ...
}
```

Assn^{dt}
First-order assertion lang.

$\exists z(x=2*z \wedge y=3*z)$

Hoare^{dt}
Hoare-style program logic

$$\frac{\{A \wedge b\}\, c\, \{A\}}{\{A\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{A \wedge \neg b\}}$$

Rigorous semantics by nonstandard analysis

- **Hoare^{dt}** : sound and relatively complete

Syntax

While^{dt}

While + dt

$$\begin{array}{lll} \text{AExp} \ni a & ::= & x \mid \mathsf{c}_r \mid a_1\ \mathrm{aop}\ a_2 \mid \mathrm{dt} \\ & & \text{where } \mathsf{c}_r \text{ is a const. for } r \in \mathbb{R},\ \mathrm{aop} \in \{+,-,\cdot,\hat{},/\} \\ \text{BExp} \ni b & ::= & \texttt{true} \mid \texttt{false} \mid b_1 \wedge b_2 \mid \neg b \mid a_1 < a_2 \\ \text{Cmd} \ni c & ::= & \texttt{skip} \mid x := a \mid c_1; c_2 \\ & & \mid \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \texttt{while}\ b\ \texttt{do}\ c \end{array}$$

Assn^{dt}

$$A \quad ::= \quad \texttt{true} \mid \texttt{false} \mid A_1 \wedge A_2 \mid \neg A \mid a_1 < a_2 \mid \forall x \in {}^*\mathbb{N}.\, A \mid \forall x \in {}^*\mathbb{R}.\, A$$

Hoare^{dt}

$$\frac{}{\{A\}\,\texttt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{A[a/x]\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1; c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\texttt{while}\ b\ \texttt{do}\ c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Syntax

While^{dt} — **While + dt**

$$\begin{array}{lll} \text{AExp} \ni a & ::= & x \mid \mathtt{c}_r \mid a_1 \ \mathtt{aop}\ a_2 \mid \mathtt{dt} \\ & & \text{where } \mathtt{c}_r \text{ is a const. for } r \in \mathbb{R},\ \mathtt{aop} \in \{+,-,\cdot,\hat{},/\} \\ \text{BExp} \ni b & ::= & \mathtt{true} \mid \mathtt{false} \mid b_1 \wedge b_2 \mid \neg b \mid a_1 < a_2 \\ \text{Cmd} \ni c & ::= & \mathtt{skip} \mid x := a \mid c_1; c_2 \\ & & \mid \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2 \mid \mathtt{while}\ b\ \mathtt{do}\ c \end{array}$$

Assn^{dt} — **Assn, *-transformed**

$$A ::= \mathtt{true} \mid \mathtt{false} \mid A_1 \wedge A_2 \mid \neg A \mid a_1 < a_2 \mid \forall x \in {}^*\mathbb{N}.\, A \mid \forall x \in {}^*\mathbb{R}.\, A$$

Hoare^{dt}

$$\frac{}{\{A\}\,\mathtt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{\,A[a/x]\,\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1; c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\mathtt{while}\ b\ \mathtt{do}\ c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Tokyo)

Syntax

While^{dt} — **While + dt**

$$\begin{array}{lll} \text{AExp} \ni a & ::= & x \mid \mathtt{c}_r \mid a_1 \ \mathtt{aop}\ a_2 \mid \mathtt{dt} \\ & & \text{where } \mathtt{c}_r \text{ is a const. for } r \in \mathbb{R},\ \mathtt{aop} \in \{+,-,\cdot,\hat{},/\} \\ \text{BExp} \ni b & ::= & \mathtt{true} \mid \mathtt{false} \mid b_1 \wedge b_2 \mid \neg b \mid a_1 < a_2 \\ \text{Cmd} \ni c & ::= & \mathtt{skip} \mid x := a \mid c_1; c_2 \\ & & \mid \mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2 \mid \mathtt{while}\ b\ \mathtt{do}\ c \end{array}$$

Thm. HOARE^{dt} rules are *sound* and *relatively complete*.

Hoare^{dt}

$$\frac{}{\{A\}\,\mathtt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{\,A[a/x]\,\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1; c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\,\{B\}}\ (\textsc{If})$$

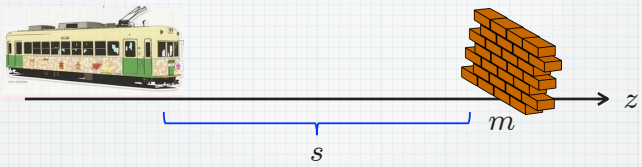
$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\mathtt{while}\ b\ \mathtt{do}\ c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Tokyo)

While^{dt} プログラムの例

$$\begin{array}{l} t := 0\,; \\ \mathtt{while}\ (t \le 1)\ \mathtt{do}\ \{ \\ \quad t := t + \mathtt{dt} \\ \} \end{array}$$

While^{dt} プログラムの例



$$\begin{array}{l} \mathtt{while}\ t < \varepsilon\ \mathtt{do}\ \{ \\ \quad t := t + \mathtt{dt}\,; \\ \quad v := v + a \cdot \mathtt{dt}\,; \\ \quad z := z + v \cdot \mathtt{dt} \\ \} \end{array}$$

$$\begin{array}{l} \mathtt{while}\ v > 0\ \mathtt{do}\ \{ \\ \quad t := 0\,; \\ \quad \mathtt{if}\ m - z < s\ \mathtt{then}\ a := -b\ \mathtt{else}\ a := a_0\,; \\ \quad \mathtt{while}\ t < \varepsilon\ \mathtt{do}\ \{ \\ \qquad t := t + \mathtt{dt}\,; \\ \qquad v := v + a \cdot \mathtt{dt}\,; \\ \qquad z := z + v \cdot \mathtt{dt} \\ \quad \}\} \end{array}$$

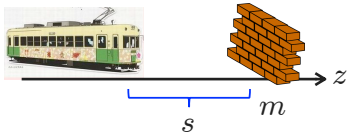
(真な) Hoare^{dt} triple の例

$$\{\, v^2 \le 2b(m-z) \,\} \left[\begin{array}{l} \texttt{while } v > 0 \texttt{ do} \\ \quad z := z + v \cdot \mathrm{dt}\,; \\ \quad v := v - b \cdot \mathrm{dt}\,; \\ \quad t := t + \mathrm{dt} \end{array}\right] \{\, z < m \,\}$$

Hoare^{dt} で
簡単に導出できます
（ループ不変量を使って）

(Tokyo)

提案手法のワークフロー



「壁にぶつかったら
困るじゃないか．．．
ええと，仮定Aのもとで」

1. モデリング

2. 仕様記述

$c :=$

```
while v > 0 do {
    t := 0;
    if m − z < s then a := −b else a := a0;
    while t < ε do {
        t := t + dt;
        v := v + a · dt;
        z := z + v · dt
    }}
```

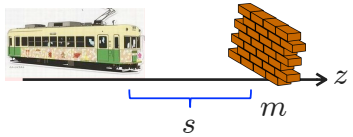
$\{A\}\ c\ \{z < m\}$

3. 形式検証

Hoare 論理で
証明検索

蓮尾 一郎（東

提案手法のワークフロー (本当は)



「壁にぶつかったら
困るじゃないか．．．
事前条件は何かな？」

1. モデリング

2. 仕様記述

$c :=$

```
while v > 0 do {
    t := 0;
    if m − z < s then a := −b else a := a0;
    while t < ε do {
        t := t + dt;
        v := v + a · dt;
        z := z + v · dt
    }}
```

$\{A\}\ c\ \{z < m\}$

3. 形式検証

事前条件 A と証明
を同時に検索

蓮尾 一郎

Hoare^{dt} の健全性

$$\frac{}{\{A\}\,\texttt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{A[a/x]\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1;c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A \wedge b\}\,c_1\,\{B\} \quad \{A \wedge \neg b\}\,c_2\,\{B\}}{\{A\}\,\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A \wedge b\}\,c\,\{A\}}{\{A\}\,\texttt{while } b \texttt{ do } c\,\{A \wedge \neg b\}}\ (\textsc{While}) \qquad \frac{\models A \Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B' \Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Thm. (Soundness)

$$\vdash \{A\}\,c\,\{B\} \implies \models \{A\}\,c\,\{B\}\ ,$$

where

$$\models \{A\}\,c\,\{B\} \overset{\text{def}}{\iff} \left[\begin{array}{l} \text{for each memory state } \sigma, \\ \sigma \models A \text{ implies } [\![c]\!](\sigma) \models B. \end{array}\right]$$

これは？？

プログラム c の「意味」
（メモリ状態の変換として）

While^{dt} プログラムの例

$t := 0$;
while $(t \leq 1)$ do {
  $t := t + \mathrm{dt}$
}

超準解析による物理情報システムの形式検証

アウトライン

* 形式検証とは？
  * Hoare 論理を例に
* ハイブリッド・システム
  * 離散＋連続
  * 物理情報システムの一側面
* 超準解析による移転
  * 離散的検証手法を，文字通りそのまま
    ハイブリッド・システムに適用

3

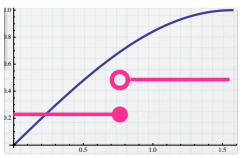
超準解析による移転

Nonstandard Analysis

* Analysis with an **infinitesimal** ∂, e.g.

"Infinitely small"
$0 < \partial < r$
$(\forall r \in \mathbb{R}_+)$

f is continuous $\iff$
$\left(|x - x'| \text{ is infinitesimal} \right.$
$\left. \implies |f(x) - f(x')| \text{ is infinitesimal} \right)$



* Done naively ➜ contradiction!

Logical foundation via an ultrafilter
[Robinson,1960]

Hyperreals

= Reals + Infinitesimals + …

Defn.
The set of *hyperreal numbers* is

$$^*\mathbb{R} := \mathbb{R}^{\mathbb{N}}/\sim_{\mathcal{F}} \ni [(a_0, a_1, a_2, \dots)]$$

0th section / 1st section / 2nd section

Ignore

* Operations: sectionwise

$$\begin{array}{rl} & [(a_0, a_1, \dots)] \\ + & [(b_0, b_1, \dots)] \\ = & [(a_0 + b_0, a_1 + b_1, \dots)] \end{array}$$

* Reals are hyperreals

$$\mathbb{R} \hookrightarrow {}^*\mathbb{R},\quad r \mapsto [(r, r, \dots)]$$

Hyperreals

= Reals + Infinitesimals + …

Defn.
The set of *hyperreal numbers* is

$$^*\mathbb{R} := \mathbb{R}^{\mathbb{N}}/\sim_{\mathcal{F}} \ni [(a_0, a_1, a_2, \dots)]$$

* Predicates: sectionwise, "for almost all i"

$$\begin{array}{ll} & [(a_i)_{i\in\mathbb{N}}] < [(b_i)_{i\in\mathbb{N}}] \\ \iff & a_i < b_i \quad \text{"for almost every } i\text{"} \\ \Longleftarrow & \{i \in \mathbb{N} \mid a_i \not< b_i\} \quad \text{is finite} \end{array}$$

"For sufficiently large i"
"Except for finitely many i"

Hyperreals

= Reals + Infinitesimals + …

Defn.
The set of *hyperreal numbers* is

$$^*\mathbb{R} := \mathbb{R}^{\mathbb{N}}/\sim_{\mathcal{F}}$$

$$\begin{array}{ll} & [(a_i)_{i\in\mathbb{N}}] < [(b_i)_{i\in\mathbb{N}}] \\ \iff & a_i < b_i \quad \text{"for almost every } i\text{"} \\ \Longleftarrow & \{i \in \mathbb{N} \mid a_i \not< b_i\} \quad \text{is finite} \end{array}$$

Prop. $\omega^{-1} = [(1, \frac{1}{2}, \frac{1}{3}, \dots)]$ is infinitesimal.

$$\omega^{-1} = (1, \tfrac{1}{2}, \tfrac{1}{3}, \dots, \tfrac{1}{N}, \tfrac{1}{N+1}, \dots)$$

OK! ∧ ✗ ✗ ✗ ✗ ✓ ✓ …

$$\tfrac{1}{N} = (\tfrac{1}{N}, \tfrac{1}{N}, \tfrac{1}{N}, \dots, \tfrac{1}{N}, \tfrac{1}{N}, \dots)$$

Trouble… Resolved

$$0 \;\begin{array}{c}>\\=\\<\\??\end{array}\; [(1, -1, 1, -1, \dots)]$$

* Meaning of "almost every i" extended
  * … so that

For each $S \subset \mathbb{N}$, exactly one of

$$S \quad \text{and} \quad \mathbb{N} \setminus S$$

is "almost all i."

* ➜ Ultrafilter!

Defn.
The set of *hyperreal numbers* is

$$^*\mathbb{R} := \mathbb{R}^{\mathbb{N}}/\sim_{\mathcal{F}}$$

Filters & Ultrafilters

Given $X \subseteq \mathbb{N}$ is "yes, almost all!" or "no!"

Defn.
An *ultrafilter* $\mathcal{F} \subseteq \mathcal{P}(\mathbb{N})$ is such that:

1. For each $X \subseteq \mathbb{N}$, exactly one of
   $$X \quad \text{and} \quad \mathbb{N} \setminus X$$
   is in $\mathcal{F}$.
2. $X, Y \in \mathcal{F} \Longrightarrow X \cap Y \in \mathcal{F}$
3. $X \in \mathcal{F}, X \subseteq Y \Longrightarrow Y \in \mathcal{F}$
4. $\emptyset \notin \mathcal{F}$

Defn.
A *filter* $\mathcal{F} \subseteq \mathcal{P}(\mathbb{N})$ is that which satisfies Cond. 2.–4.

Prop.
$$\mathcal{F}_c := \{S \subseteq \mathbb{N} \mid \mathbb{N} \setminus S \text{ is finite}\}$$
is a filter (the *cofinite/Frechet* filter).

Prop.
Any filter $\mathcal{F}'$ can be extended to an ultrafilter $\mathcal{F} \supseteq \mathcal{F}'$. (By Zorn's lemma)

Cor.
There is an ultrafilter $\mathcal{F}$ such that $\mathcal{F}_c \subseteq \mathcal{F}$.

Fix one such

Hyperreals

Defn.
The set of *hyperreal numbers* is
$$^*\mathbb{R} := \mathbb{R}^{\mathbb{N}} / \sim_{\mathcal{F}}.$$

$$(a_0, a_1, \dots) \sim_{\mathcal{F}} (b_0, b_1, \dots) \overset{\text{def}}{\iff} \{i \in \mathbb{N} \mid a_i = b_i\} \in \mathcal{F}$$

* Predicates: pointwise, "for almost every i"

$$\begin{aligned} &[(a_i)_{i\in\mathbb{N}}] < [(b_i)_{i\in\mathbb{N}}] \\ \iff\ & a_i < b_i \quad \text{for "almost every } i\text{"} \\ \iff\ & \{i \in \mathbb{N} \mid a_i < b_i\} \in \mathcal{F} \end{aligned}$$

* Consequences: ω is infinite; ω^{-1} is infinitesimal;
  *R is an ordered field;
  [(0,1,0,1,...)] is either 0 or 1; ...

The Sectionwise Paradigm

Thm. (Łos)
For any first-order formula $\varphi(x)$ and a hyperreal $\mathbf{a} = [(a_i)_{i\in\mathbb{N}}]$,
$$^*\mathbb{R} \models \varphi(\mathbf{a}) \iff \{i \in \mathbb{N} \mid \mathbb{R} \models \varphi(a_i)\} \in \mathcal{F}.$$

$$\mathbf{a} = [(a_0, a_1, a_2, \dots)]$$
0th section, 1st section, 2nd section

a satisfies φ
iff
almost every section does

移転原理
The Transfer Principle

Thm. (Łos)
For any first-order formula $\varphi(x)$ and a hyperreal $\mathbf{a} = [(a_i)_{i\in\mathbb{N}}]$,
$$^*\mathbb{R} \models \varphi(\mathbf{a}) \iff \{i \in \mathbb{N} \mid \mathbb{R} \models \varphi(a_i)\} \in \mathcal{F}.$$

Thm.
For any first-order sentence φ,
$$\mathbb{R} \models \varphi \iff {}^*\mathbb{R} \models \varphi.$$

$\forall x, y.\ (x < y \vee x = y \vee x > y)$
$\forall x.\ (x \neq 0 \implies \exists y.\ (xy = 1))$

$\forall x \forall \mathbb{R}.\ \psi$ —*-transform→ $\forall x \forall {}^*\mathbb{R}.\ \psi$

我々の戦略：「Hoare 論理に対する移転原理」

数理論理学における Filters & Ultrafilters

Defn.
A subset $\boldsymbol{F} \subseteq \boldsymbol{A}$ of a BA is a *(proper) filter* if it is

- upward closed,
- closed under $\top$ and $\wedge$, and
- $\bot \notin \boldsymbol{F}$.

* filter ≃ consistent theory
* (filter ↦ ultrafilter)
  = (consistent theory ↦ model)
  = strong completeness!

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012-)

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012-)

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012-)

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012-)

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012-)

THE COAUTHOR

Kohei Suenaga

- PhD (U.Tokyo, 2008) with Naoki Kobayashi
- Program verification, static analysis, type systems
- Industiral experience (IBM Research)
- Assist. Prof. at Kyoto U. (2012.4-)

[Fixed pt. obs., Braga, PT, 2007]

超準解析による While^{dt} の意味論

```
t := 0 ;
while (t ≤ 1) do {
  t := t + dt
}
```

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t < 1)
  t := t + dt;
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t < 1)
  t := t + dt;
```

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (t < (1,1,1,...))
  t := t + (1, 1/2, 1/3, ...) ;
```

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

0th section	1st section	2nd section	
`t := 0; while (t < 1) t := t + 1 ;`	`t := 0; while (t < 1) t := t + 1/2 ;`	`t := 0; while (t < 1) t := t + 1/3 ;`	...
`t = 1`	`t = 1`	`t = 1`	...

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (t < (1,1,1,...))
  t := t + (1, 1/2, 1/3, ...) ;
```

```
t = (1,1,1,...)
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t < 1)
  t := t + dt;
```

```
t = 1
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t <= 1)
  t := t + dt;
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t <= 1)
  t := t + dt;
```

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (t <= (1,1,1,...))
  t := t + (1, 1/2, 1/3, ...) ;
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

0th section	1st section	2nd section	

`t := 0;`
`while (t <= 1)`
`  t := t + 1 ;`
	`t := 0;`
	`while (t <= 1)`
	`  t := t + 1/2 ;`
		`t := 0;`
		`while (t <= 1)`
		`  t := t + 1/3 ;`
			...

`t = 1 + 1`	`t = 1 + 1/2`	`t = 1 + 1/3`	...

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (t <= (1,1,1,...))
  t := t + (1, 1/2, 1/3, ...) ;
```

```
t = (1,1,1,...) + (1, 1/2, 1/3, ...)
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (t <= 1)
  t := t + dt;
```

```
t = 1 + dt
```

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (true)
  t := t + dt;
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (true)
  t := t + dt;
```

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (true)
  t := t + (1, 1/2, 1/3, ...) ;
```

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

0th section	1st section	2nd section	
`t := 0; while (true) t := t + 1 ;`	`t := 0; while (true) t := t + 1/2 ;`	`t := 0; while (true) t := t + 1/3 ;`	...
⊥	⊥	⊥	...

Hasuo (Tokyo)

Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := (0,0,0,...);
while (true)
  t := t + (1, 1/2, 1/3, ...) ;
```

```
t = (⊥,⊥,⊥,...)
```


Denotational Semantics

* Execute sectionwise and bundle up the outcomes!

```
t := 0;
while (true)
  t := t + dt;
```

```
⊥
```


Denotational Semantics

$$\left[\begin{array}{l} t := 0\,; \\ \texttt{while}\ (t \le 1)\ \texttt{do} \\ \quad t := t + \mathtt{dt} \end{array}\right] \overset{i\text{-th section}}{\longmapsto} \left[\begin{array}{l} t := 0\,; \\ \texttt{while}\ (t \le 1)\ \texttt{do} \\ \quad t := t + \frac{1}{i+1} \end{array}\right]$$

$$\begin{array}{lcl} [\![x]\!]\sigma & := & \sigma(x) \\ [\![a_1 \mathbin{\texttt{aop}} a_2]\!]\sigma & := & [\![a_1]\!]\sigma \mathbin{\texttt{aop}} [\![a_2]\!]\sigma \\ [\![\mathtt{dt}]\!]\sigma & := & \omega^{-1} = [\,(1, \frac{1}{2}, \frac{1}{3}, \dots)\,] \end{array}$$

$$\begin{array}{lcl} [\![\texttt{true}]\!]\sigma & := & \mathtt{tt} \\ [\![b_1 \wedge b_2]\!]\sigma & := & [\![b_1]\!]\sigma \wedge [\![b_2]\!]\sigma \\ [\![a_1 < a_2]\!]\sigma & := & [\![a_1]\!]\sigma < [\![a_2]\!]\sigma \end{array} \qquad [\![\mathtt{fal}$$
$$[\![\neg b$$

Def.
The *i-th section* of a WHILE$^{\mathrm{dt}}$ expression e is

$$e|_i \quad :\equiv \quad e\left[\,\frac{1}{i+1} \,/\, \mathrm{dt}\,\right] .$$

$$[\![\texttt{skip}]\!]\sigma := \sigma \qquad [\![x := a]\!]\sigma := \sigma[x \mapsto [\![a]\!]\sigma] \qquad [\![c_1; c_2]\!]\sigma := [\![c_2]\!]([\![c_1]\!]\sigma)$$

$$[\![\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2]\!]\sigma := \begin{cases} [\![c_1]\!]\sigma & \text{if } [\![b]\!]\sigma = \mathtt{tt} \\ [\![c_2]\!]\sigma & \text{if } [\![b]\!]\sigma = \mathtt{ff} \end{cases}$$

$$[\![\texttt{while } b \texttt{ do } c]\!]\sigma := \Big([\![\,(\texttt{while } b \texttt{ do } c)|_i\,]\!](\sigma|_i) \Big)_{i \in \mathbb{N}}$$

Sectionwise definition

"Sectionwise Lemmas"

Sectionwise Execution Lemma.
For any expr. e and $i \in \mathbb{N}$,

$$[\![e]\!]\sigma = \big[\,\big([\![e|_i]\!](\sigma|_i)\big)_{i \in \mathbb{N}}\,\big] .$$

while 文だけでなく，すべて sectionwise

Sectionwise Satisfaction Lemma.
For any hyperstate σ and an ASSN$^{\mathrm{dt}}$ formula φ:

$$\sigma \models \varphi \iff \sigma|_i \models \varphi|_i \quad \text{for almost every } i.$$

"Łos' Theorem"

"Sectionwise Lemmas"

Lem. (Sectionwise validity of Hoare triples)

$$\models \{A\}c\{B\} \iff \models \{A|_i\}\, c|_i\, \{B|_i\} \quad \text{for almost every } i.$$

プログラム検証手法の
移転のためのインターフェイス

Hasuo (Tokyo)

Hoare^{dt} の健全性

$$\frac{}{\{A\}\,\mathtt{skip}\,\{A\}}\ (\textsc{Skip}) \qquad \frac{}{\{A[a/x]\}\,x := a\,\{A\}}\ (\textsc{Assign})$$

$$\frac{\{A\}\,c_1\,\{C\} \quad \{C\}\,c_2\,\{B\}}{\{A\}\,c_1;c_2\,\{B\}}\ (\textsc{Seq}) \qquad \frac{\{A\wedge b\}\,c_1\,\{B\} \quad \{A\wedge\neg b\}\,c_2\,\{B\}}{\{A\}\,\mathtt{if}\ b\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\,\{B\}}\ (\textsc{If})$$

$$\frac{\{A\wedge b\}\,c\,\{A\}}{\{A\}\,\mathtt{while}\ b\ \mathtt{do}\ c\,\{A\wedge\neg b\}}\ (\textsc{While}) \qquad \frac{\models A\Rightarrow A' \quad \{A'\}\,c\,\{B'\} \quad \models B'\Rightarrow B}{\{A\}\,c\,\{B\}}\ (\textsc{Conseq})$$

Thm. (Soundness)

$$\vdash \{A\}\,c\,\{B\} \implies \models \{A\}\,c\,\{B\}\ ,$$

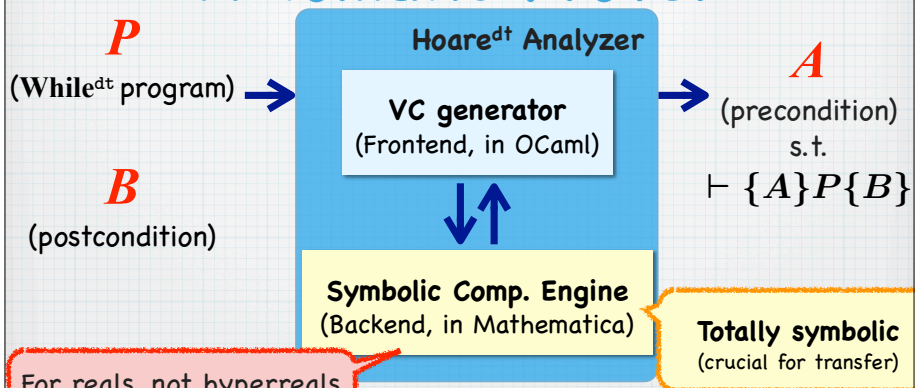
where

$$\models \{A\}\,c\,\{B\} \overset{\text{def}}{\iff} \left[\begin{array}{l}\text{for each memory state } \sigma,\\ \sigma \models A \text{ implies } [\![c]\!](\sigma) \models B.\end{array}\right]$$

Sectionwise lemmas から
直ちに導ける

プログラム c の「意味」
（メモリ状態の変換として）

Prototype Automatic Prover



For reals, not hyperreals
➜ justified by the
transfer principle

Core AMD Opteron 2.3GHz CPU, 32GB memory.
x x86 (64-bit)

th one manual insertion of invariants

Hasuo (Tokyo)

Q. Does the choice of dt matter?

* A.
Yes, for some "pathological" programs

```
t := 0;
while (t ≠ 1)
  t := t + dt;
```

Terminates with dt = (1, 1/2, 1/3, ...)
Doesn't with dt = (1/π, 1/2π, 1/3π, ...)

Q. While^{dt} program って実行できなくない？

* A. その通り，実行できません．
  * ~~プログラミング言語~~ モデリング言語
    * ~~数値的近似~~
      無限小値による，exact なモデリング
  * Static analysis
    ➜ 実行できなくてOK！
    * 「意味論＋健全性」で十分

Static analysis （静的解析）
* 実行することなく解析
* ↔ Dynamic analysis
* 例： プログラム論理による検証

まとめとこれから

* 微小量（infinitesimal）を持つ，
  解析学の形式化
* Ultrafilter によるモデル
  ➜ 数理論理学（モデル理論）の成果！

超準解析による物理情報システムの形式検証

形式検証の新たなターゲット
* 連続ダイナミクス（物理系）
* 離散ダイナミクス
  （デジタル制御）

数学（数理論理学）
による品質保証

Static Analysis

Nonstandard Static Analysis

Nonstandard Analysis

Nonstandard Static Analysis

* プログラム検証のいろいろな枠組みを超準化
  * プログラミング言語（モデリング言語）
  * 形式的意味論
  * プログラム論理

* 離散的検証手法を，
  連続系・ハイブリッドシステムに
  文字通りそのまま移転
  * 微分方程式は不要

論文

今日の理論的内容

* [ICALP'11] K. Suenaga and I. Hasuo. Programming with infinitesimals: A while-language for hybrid system modeling. In L. Aceto, M. Henzinger and J. Sgall, editors, ICALP (2), vol. 6756 of Lect. Notes Comp. Sci., pp. 39[illegible] 2011.

自動検証器

* [CAV'12] I. Hasuo and K. Suenaga. Exercises in Nonstandard Static Analysis of hybrid systems. In P. Madhusudan and S.A. Seshia, editors, CAV, vol. 7358 of Lect. Notes Comp. Sci., pp. 462–478. Springer, 2012.

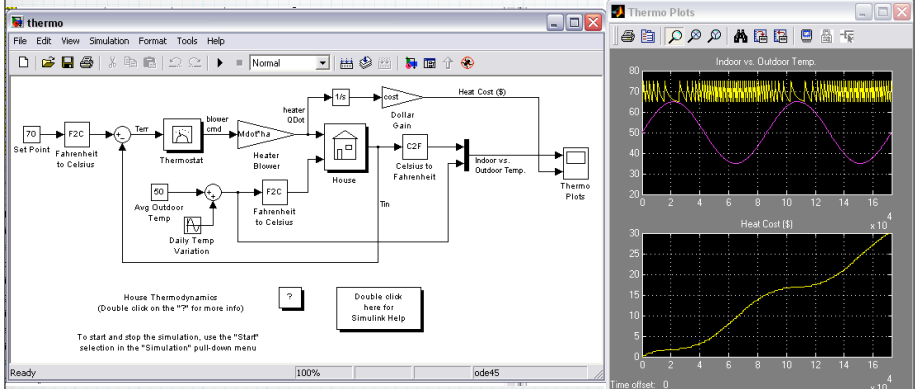
ストリーム処理言語への応用
（Simulink により近い）

* [POPL'13] K. Suenaga, H. Sekine [illegible] processing systems: nonstandard [illegible] time signals. In R. Giacobazzi and [illegible] pp. 417–430. ACM, 2013.

論文・スライドなどはウェブページから

http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro/

Matlab/Simulink



* 産業界でのデファクト・スタンダード
  （オープンソースの代替も多数）
* 用途： モデリング，シミュレーション（テスト），解析

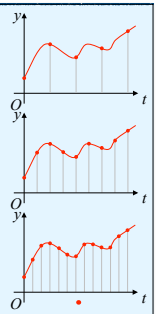
Nonstandard Static Analysis in [POPL'13]

* Advanced 超準解析 [Robinson & Zakon, 1969]：
  算術の言語から集合論の言語へ
  * 2項述語 ∈
  * 意味論：“superstructure” による
* すると，超領域理論 hyperdomain theory が展開可能
  * e.g. 「超領域上の *-連続関数は最小不動点を持つ」

* … 持ちあげるのは簡単．
  Standard な世界へのフィードバックがむずかしい．

Nonstandard St
in [POPL

Hypersampling
による同一視
（mild な連続性
の仮定の下に）

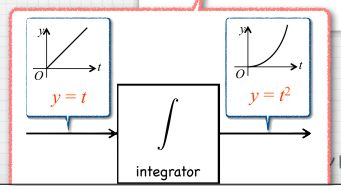


（dt 秒刻み）
超ストリーム処理言語

超準化

（離散時間）
ストリーム処理言語

（連続時間）
シグナル処理



Future Work

* ツールの実効性向上
  * 産業界の現場で重要な問題に適用
* プログラムの「可積分性」を分析
* 構成的に（トポスを使う）
  [Moerdijk, Palmgren, 片岡, ...]
* 精度保証付き数値計算との関係
* 力学系理論・制御理論との協働

For You to Take Home

* Mathematical logic at work!

ご清聴ありがとうございました！
蓮尾 一郎（東大・情報理工・コンピュータ科学）
http://www-mmm.is.s.u-tokyo.ac.jp/~ichiro

* 微小量（infinitesimal）を持つ，
  解析学の形式化
* Ultrafilter によるモデル
  ➜ 数理論理学（モデル理論）の成果！

超準解析による
物理情報システムの形式検証

形式検証の新たなターゲット
* 連続ダイナミクス（物理系）
* 離散ダイナミクス
  （デジタル制御）

数学（数理論理学）
による品質保証