Abstract Hoare Logic

Paulo Oliva
(joint work with U. Martin and E. A. Mathiesen)

Queen Mary, University of London, UK
(pbo@dcs.qmul.ac.uk)

TMC, Network Algebras, and Applications

Wrocław, 15 July 2007

Aim

Hoare logic for continuous systems

Outline

1. Introduction
   - Hoare logic
   - TMC and system categories
2. Abstract Hoare logic
   - Verification functor
   - Abstract logical rules
3. Instantiations
   - While programs: partial correctness
   - Pointer programs: partial correctness
   - While programs: complexity and termination
   - Stream circuits
   - Continuous systems

Outline

1. Introduction
   - Hoare logic
   - TMC and system categories
2. Abstract Hoare logic
   - Verification functor
   - Abstract logical rules
3. Instantiations
   - While programs: partial correctness
   - Pointer programs: partial correctness
   - While programs: complexity and termination
   - Stream circuits
   - Continuous systems

Hoare Logic

Hoare triples: $\{P\}\ f\ \{Q\}$

- **Partial correctness**

  If input satisfies P then output (if terminates) satisfies Q

- **Partial correctness** (pointer programs)

  If input satisfies P then program does not abort and output (if terminates) satisfies Q

- **Backward reasoning**

  For output to satisfy Q it is sufficient that input satisfies P

- **Total correctness**

  If input satisfies P then f terminates and output satisfies Q

Hoare Logic

- **Higher order programs**
- **Parallel programs**
- ...

Hoare Logic

- **Higher order programs**
- **Parallel programs**
- ...
- **Continuous systems**

Motivation

- Develop Hoare logic for **continuous systems**

Motivation

- Develop Hoare logic for **continuous systems**
- ... or show that such thing does not exist

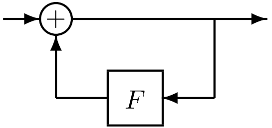
Motivation

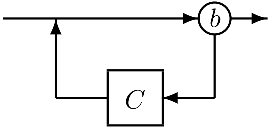
- Develop Hoare logic for **continuous systems**
- ... or show that such thing does not exist
- Proceeded by trying to understand "structure" of Hoare logics

Related Work

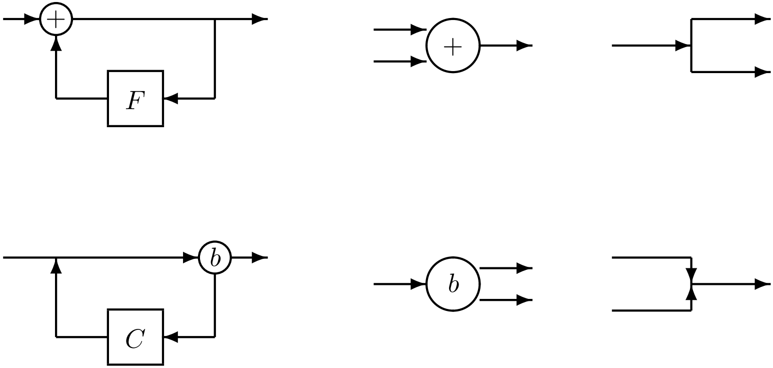
- Dijkstra's predicate transformer
- Kozen's KAT (Kleene Algebras with Test)
- Abramsky's specification categories
- Bloom and Esik's iteration theory
- ...

Network vs Flowcharts



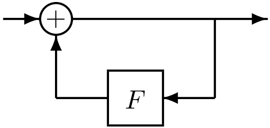


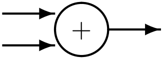
Network vs Flowcharts

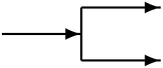


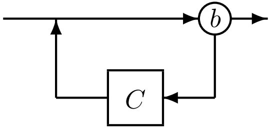
Network vs Flowcharts

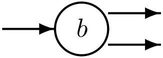
$$C^\infty \to (C^\infty \times C^\infty)$$

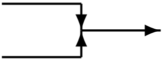








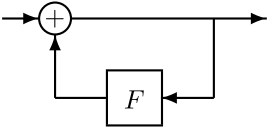


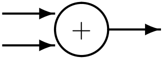


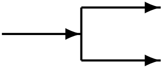
$$H \to (H \uplus H)$$

Network vs Flowcharts

$C^\infty \to (C^\infty \times C^\infty)$

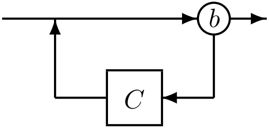


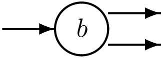


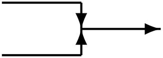


$(C^\infty \times C^\infty) \to C^\infty$

$(H \uplus H) \to H$







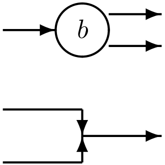
$H \to (H \uplus H)$

Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

Each flowchart corresponds to a network

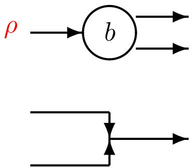


Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

Each flowchart corresponds to a network

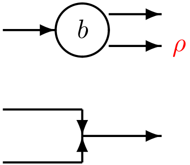


Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

Each flowchart corresponds to a network

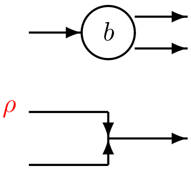


Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

Each flowchart corresponds to a network

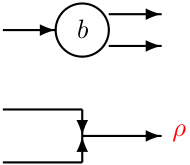


Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^{H} \times 2^{J}$$

Each flowchart corresponds to a network



Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

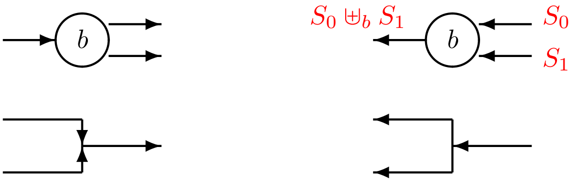
Each flowchart corresponds to a network

Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^{H} \times 2^{J}$$

Each flowchart corresponds to a network

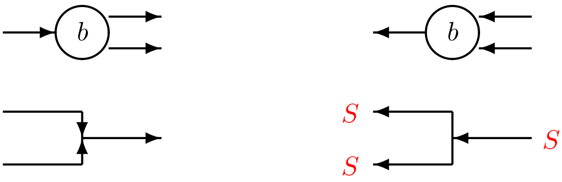


Bainbridge Duality

Exploit the duality between sum and product

$$2^{H \uplus J} \simeq 2^H \times 2^J$$

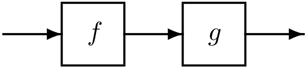
Each flowchart corresponds to a network



Monoidal Categories

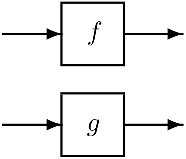
- **Sequential composition**: categorical composition
  $f : X \to Y$, $g : Y \to Z$ then $g \circ f : X \to Z$

$g \circ f$



- **Parallel composition**: Monoidal operation
  $f : X \to Y$, $g : Z \to W$ then $f \otimes g : (X \otimes Z) \to (Y \otimes W)$

$f \otimes g$



Traced Monoidal Categories

- **Iteration**: Trace operation
  If $f : (X \otimes Z) \to (Y \otimes Z)$ then $\mathsf{Tr}(f) : X \to Y$

Traced Monoidal Categories

- **Iteration**: Trace operation
  If $f : (X \otimes Z) \to (Y \otimes Z)$ then $\mathsf{Tr}(f) : X \to Y$

- **Examples**
  - Disjoint union
    $\mathsf{Tr}(f) \equiv \{\langle x, y\rangle : \exists z_0, \ldots, z_n(\langle x, z_0\rangle \in f \wedge \ldots \wedge \langle z_n, y\rangle \in f)\}$
  - Cartesian products
    $\mathsf{Tr}(f) \equiv \{\langle x, y\rangle : \exists z(\langle\langle x, z\rangle, \langle y, z\rangle\rangle \in f)\}$

System Category

Let $\mathsf{cl}(M)$ denote the closure of the set of morphisms M under sequential and monoidal composition, and trace.

Definition (System category)

A *system category* $\mathcal{S}$ is a traced monoidal category with a distinguished set of morphisms $\mathcal{S}_b \subseteq \mathcal{S}_m$, so-called *basic systems*, such that $\mathsf{cl}(\mathcal{S}_b) = \mathcal{S}_m$.

System Category

Let cl(M) denote the closure of the set of morphisms M under sequential and monoidal composition, and trace.

Definition (System category)

A *system category* $\mathcal{S}$ is a traced monoidal category with a distinguished set of morphisms $\mathcal{S}_b \subseteq \mathcal{S}_m$, so-called *basic systems*, such that cl$(\mathcal{S}_b) = \mathcal{S}_m$.

Flowcharts	**Stream circuits**
Boolean Test $(\Sigma \to \Sigma \uplus \Sigma)$	Sum $(\Sigma \times \Sigma \to \Sigma)$
Joining of Wires $(\Sigma \uplus \Sigma \to \Sigma)$	Splitting of Wires $(\Sigma \to \Sigma \times \Sigma)$
Assignment $(\Sigma \to \Sigma)$	Scalar Multiplication $(\Sigma \to \Sigma)$
	Register $(\Sigma \to \Sigma)$

Outline

Hoare Logic

- *Pre/Post-conditions*:
  Describe properties of input/output

Hoare Logic

- *Pre/Post-conditions*:
  Describe properties of input/output
- *Ordering on information*:
  Rule of consequence

Hoare Logic

- *Pre/Post-conditions*:
  Describe properties of input/output
- *Ordering on information*:
  Rule of consequence
- *Partial correctness assertions*:
  Predicate transformers

Hoare Logic

- *Pre/Post-conditions*:
  Describe properties of input/output
- *Ordering on information*:
  Rule of consequence
- *Partial correctness assertions*:
  Predicate transformers
- *Others*:
  Strongest post condition, loop invariant, ...

Pre Order

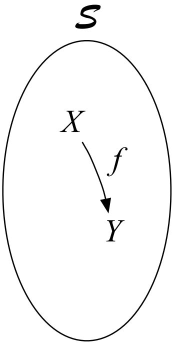
Hoare Logic	**Abstract Hoare Logic**
Pre/Post-conditions	Elements of pre-orders
Logical implication	Partial order
Rule of consequence	Monotonicity
Loop invariants	Fixed points
...	...

Pre Order

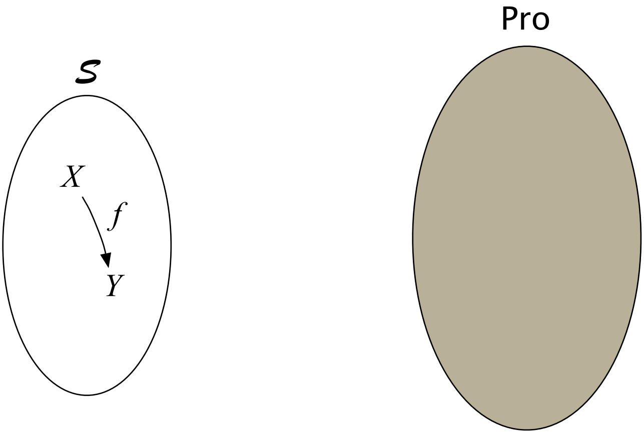
Hoare Logic	**Abstract Hoare Logic**
Pre/Post-conditions	Elements of pre-orders
Logical implication	Partial order
Rule of consequence	Monotonicity
Loop invariants	Fixed points
...	...

Hoare logic derived from embedding of a TMC into category of pre-orders

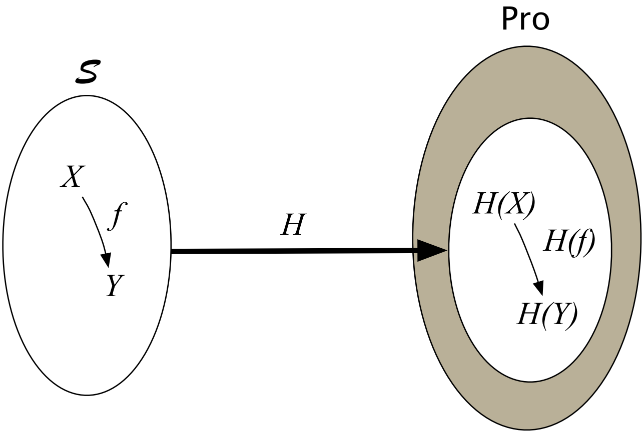
Pre Order



Pre Order



Pre Order



Verification Functor

Definition (Verification functor)

A strict monoidal functor $H : \mathcal{S} \to \mathsf{Pro}$ is called a *verification functor* for $\mathcal{S}$ if it satisfies:

(1) trace soundness

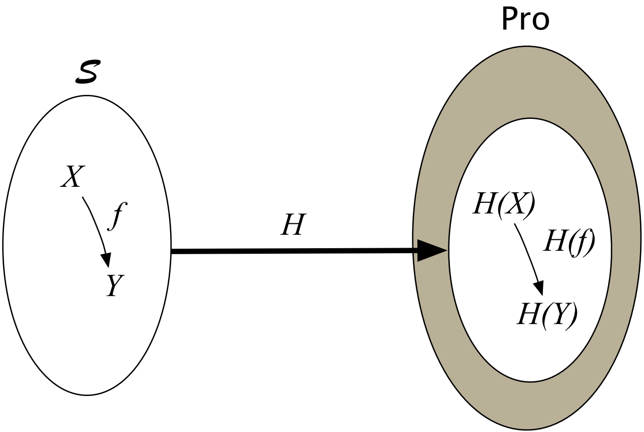
$$\exists Q^{H(Z)}(H(f)\langle P, Q\rangle \sqsubseteq \langle R, Q\rangle) \;\Rightarrow\; H(\mathsf{Tr}(f))(P) \sqsubseteq R$$

(2) trace completeness

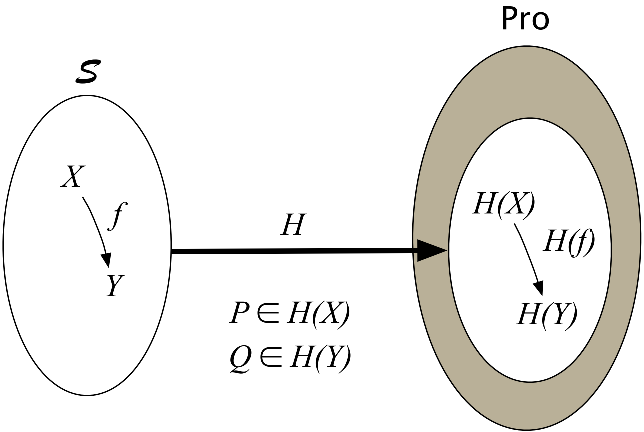
$$H(\mathsf{Tr}(f))(P) \sqsubseteq R \;\Rightarrow\; \exists Q^{H(Z)}(H(f)\langle P, Q\rangle \sqsubseteq \langle R, Q\rangle)$$

for all $f : X \otimes Z \to Y \otimes Z$ in $\mathcal{S}$, and $P \in H(X)$, $R \in H(Y)$.

Abstract Hoare Triples



Abstract Hoare Triples



$\{P\}\, f\, \{Q\}$ $\qquad$ $H(f)(P) \subseteq Q$

Abstract Hoare Triples

Let

- $H : \mathcal{S} \to$ Pro be a verification functor
- $f : X \to Y$ is a morphism (system) in $\mathcal{S}$
- $P \in H(X)$ and $Q \in H(Y)$

Definition (Abstract Hoare triples)

We define abstract Hoare triples as

$$\{P\}\ f\ \{Q\} \ :\equiv\ H(f)(P) \sqsubseteq_{H(Y)} Q$$

Abstract Hoare Logic

Theorem (Soundness and completeness)

The following set of rules is sound and complete for any system category $\mathcal{S}$ and verification functor $H : \mathcal{S} \to \mathsf{Pro}$:

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{H(f)(P)\}}\ \text{(axiom)}$$

$$\frac{P' \sqsubseteq_X P \quad \{P\}\ f\ \{Q\} \quad Q \sqsubseteq_Y Q'}{\{P'\}\ f\ \{Q'\}}\ \text{(csq)}$$

Abstract Hoare Logic

Theorem (Soundness and completeness)

The following set of rules is sound and complete for any system category $\mathcal{S}$ and verification functor $H : \mathcal{S} \to \mathsf{Pro}$:

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{H(f)(P)\}}\ \text{(axiom)} \qquad \frac{\{P\}\ f\ \{Q\} \quad \{Q\}\ g\ \{R\}}{\{P\}\ g \circ f\ \{R\}}\ (\circ)$$

$$\frac{P' \sqsubseteq_X P \quad \{P\}\ f\ \{Q\} \quad Q \sqsubseteq_Y Q'}{\{P'\}\ f\ \{Q'\}}\ \text{(csq)}$$

Abstract Hoare Logic

Theorem (Soundness and completeness)

The following set of rules is sound and complete for any system category $\mathcal{S}$ and verification functor $H : \mathcal{S} \to \mathsf{Pro}$:

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{H(f)(P)\}}\ \text{(axiom)} \qquad \frac{\{P\}\ f\ \{Q\} \quad \{Q\}\ g\ \{R\}}{\{P\}\ g \circ f\ \{R\}}\ (\circ)$$

$$\frac{\{P\}\ f\ \{Q\} \quad \{R\}\ g\ \{S\}}{\{\langle P, R\rangle\}\ f \otimes g\ \{\langle Q, S\rangle\}}\ (\otimes)$$

$$\frac{P' \sqsubseteq_X P \quad \{P\}\ f\ \{Q\} \quad Q \sqsubseteq_Y Q'}{\{P'\}\ f\ \{Q'\}}\ \text{(csq)}$$

Abstract Hoare Logic

Theorem (Soundness and completeness)

The following set of rules is sound and complete for any system category $\mathcal{S}$ and verification functor $H : \mathcal{S} \to \mathsf{Pro}$:

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{H(f)(P)\}}\ (\text{axiom}) \qquad \frac{\{P\}\ f\ \{Q\} \quad \{Q\}\ g\ \{R\}}{\{P\}\ g \circ f\ \{R\}}\ (\circ)$$

$$\frac{\{P\}\ f\ \{Q\} \quad \{R\}\ g\ \{S\}}{\{\langle P, R\rangle\}\ f \otimes g\ \{\langle Q, S\rangle\}}\ (\otimes) \qquad \frac{\{\langle P, Q\rangle\}\ f\ \{\langle R, Q\rangle\}}{\{P\}\ \mathsf{Tr}_{\mathcal{S}}(f)\ \{R\}}\ (\mathsf{Tr}_{\mathcal{S}})$$

$$\frac{P' \sqsubseteq_X P \quad \{P\}\ f\ \{Q\} \quad Q \sqsubseteq_Y Q'}{\{P'\}\ f\ \{Q'\}}\ (\text{csq})$$

Outline

While Programs

- Var: set of program variables
- Store Σ : Var $\rightarrow \mathbb{Z}$
- Atomic programs
  - Assignment $(x := t) : \Sigma \rightarrow \Sigma$
  - Joining $\Delta : \Sigma \uplus \Sigma \rightarrow \Sigma$
  - Boolean test $\text{if}_b : \Sigma \rightarrow \Sigma \uplus \Sigma$

While Programs (partial correctness, forward reasoning)

- Pre order $(\mathcal{P}(\Sigma), \subseteq)$
- $H(f)(P) :\equiv \{y \in Y \;:\; \exists x \in P \; (f(x) = y)\}$
  $H(f)(P) :\equiv \mathsf{SPC}(f, P)$
- $\{P\}\; f \;\{Q\}$ means $H(f)(P) \subseteq Q$, i.e.
  "if P holds before execution then (if program terminates) Q holds afterwards"
- For the basic systems we have:

$$\begin{array}{rcl} \{P\} & x := t & \{\exists x_0 (P[x_0/x] \wedge x = t[x_0/x])\} \\ \{\langle P, Q \rangle\} & \Delta & \{P \vee Q\} \\ \{P\} & \mathsf{if}_b & \{\langle P \wedge b, P \wedge \neg b \rangle\} \end{array}$$

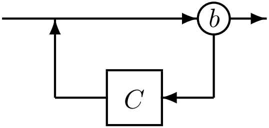
While Programs (partial correctness, backward reasoning)

- Pre order $(\mathcal{P}(\Sigma), \supseteq)$
- $H(f)(P) :\equiv \{x \in X \; : \; f(x) \in Q\}$
  $H(f)(P) :\equiv \mathsf{WPC}(f, P)$
- $\{P\}\; f\; \{Q\}$ means $H(f)(P) \supseteq Q$, i.e.
  "*in order for P to hold after (terminating) execution it is sufficient that Q holds before*"
- For the basic systems we have:

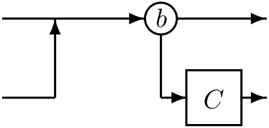
$$\begin{array}{ccc} \{P\} & x := t & \{P[t/x]\} \\ \{P\} & \Delta & \{\langle P, P\rangle\} \\ \{\langle P, Q\rangle\} & \mathsf{if}_b & \{(P \wedge b) \vee (Q \wedge \neg b)\} \end{array}$$

While Loop Rule

$\mathsf{while}_b(C)$

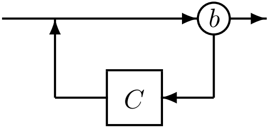


$(1 \uplus C) \circ \mathsf{if}_b \circ \Delta$

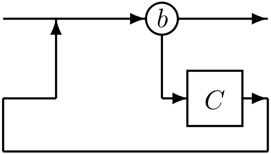


While Loop Rule

$\mathsf{while}_b(C)$

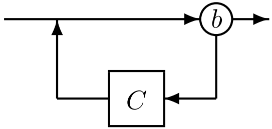


$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$

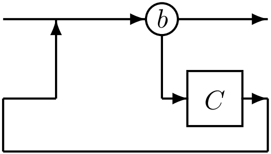


While Loop Rule

$\text{while}_b(C)$



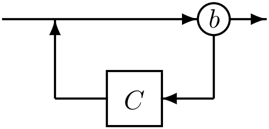
$\text{Tr}((1 \uplus C) \circ \text{if}_b \circ \Delta)$



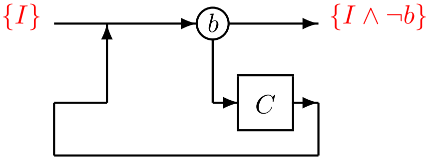
$$
\dfrac{\dfrac{\dfrac{\{I\}\ \text{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\,(\uplus)}{\{I\}\ (1 \uplus C) \circ \text{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}\,(\circ)}{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \text{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}\,(\circ)}{\{I\}\ \text{while}_b(C)\ \{I \wedge \neg b\}}\,(\text{Tr})
$$

While Loop Rule

$\mathsf{while}_b(C)$



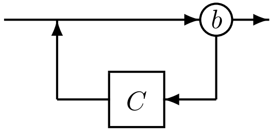
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$



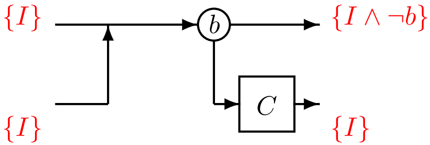
$$\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \quad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\,(\uplus)}{\dfrac{\{I\}\ (1 \uplus C) \circ \mathsf{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \mathsf{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}\,(\circ)}\,(\circ)}{\{I\}\ \mathsf{while}_b(C)\ \{I \wedge \neg b\}}\,(\mathsf{Tr})$$

While Loop Rule

$\mathsf{while}_b(C)$



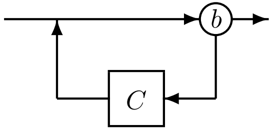
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$



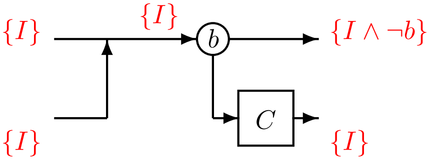
$$\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \quad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)}{\dfrac{\{I\}\ (1 \uplus C) \circ \mathsf{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \mathsf{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}\ (\circ)}\ (\circ)}{\{I\}\ \mathsf{while}_b(C)\ \{I \wedge \neg b\}}\ (\mathsf{Tr})$$

While Loop Rule

$\mathsf{while}_b(C)$



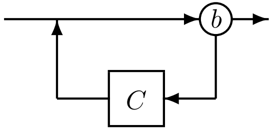
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$



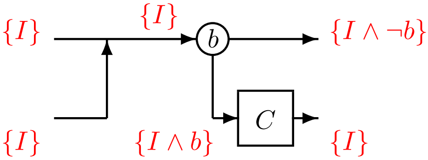
$$\cfrac{\cfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \cfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\,(\uplus)}{\{I\}\ (1 \uplus C) \circ \mathsf{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}\,(\circ)}{\cfrac{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \mathsf{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}{\{I\}\ \mathsf{while}_b(C)\ \{I \wedge \neg b\}}\,(\mathsf{Tr})}\,(\circ)$$

While Loop Rule

$\mathsf{while}_b(C)$



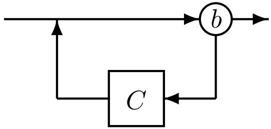
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$



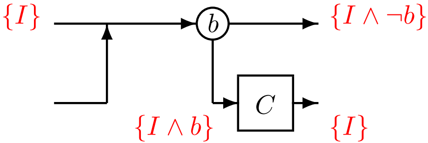
$$\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \quad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)}{\dfrac{\{I\}\ (1 \uplus C) \circ \mathsf{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \mathsf{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}\ (\circ)}\ (\circ)}{\{I\}\ \mathsf{while}_b(C)\ \{I \wedge \neg b\}}\ (\mathsf{Tr})$$

While Loop Rule

$\mathsf{while}_b(C)$



$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$



$$
\cfrac{
  \cfrac{
    \{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\}
    \qquad
    \cfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)
  }{
    \cfrac{\{I\}\ (1 \uplus C) \circ \mathsf{if}_b\ \{\langle I \wedge \neg b, I\rangle\}}{\{\langle I, I\rangle\}\ (1 \uplus C) \circ \mathsf{if}_b \circ \Delta\ \{\langle I \wedge \neg b, I\rangle\}}\ (\circ)
  }\ (\circ)
}{
  \{I\}\ \mathsf{while}_b(C)\ \{I \wedge \neg b\}
}\ (\mathsf{Tr})
$$

Pointer Programs

- Store Σ : Var $\to \mathbb{Z}$
- Heap Π : partial functions $\mathbb{N} \to \mathbb{Z}$ with finite domain
- State : $(\Sigma \times \Pi) \cup \{\text{abort}\}$
- Atomic programs
  - Look up $(x := [t])$
  - Mutation $([t] := s)$
  - Allocation $(x := \text{new}(t))$
  - Deallocation $\text{disp}(t)$

Separation Logic

- Pre order $(\mathcal{P}(\Sigma \times \Pi), \supseteq)$
- $H(f)(P) :\equiv \mathsf{WPC}(f, P)$
- $\{P\}\ f\ \{Q\}$ means $H(f)(P) \supseteq Q$, i.e.
  "if Q holds before execution then f does not abort and if terminates output satisfies P"
- For the basic systems we have:

$$\{P\} \quad x := [t] \quad \{\exists v'((t \mapsto v') * ((t \mapsto v') -\!\!* P[v'/x]))\}$$
$$\{P\} \quad [t] := s \quad \{(t \mapsto -) * ((t \mapsto s) -\!\!* P)\}$$
$$\{P\} \quad x := \mathsf{new}(t) \quad \{\forall i((i \mapsto t) -\!\!* P[i/x])\}$$
$$\{P\} \quad \mathsf{disp}(t) \quad \{(t \mapsto -) * P\}$$

Hoare Logic for Complexity (backward reasoning)

- Pre order $(\Sigma \to \mathbb{N}_\infty, \leq)$
- $H(f)(P)(\rho) :\equiv P(f(\rho)) +$ (time to execute f on ρ)
- $\{P\}\ f\ \{Q\}$ means $\forall \rho (H(f)(P)(\rho) \leq Q(\rho))$
  "*starting with Q credits we can run f and still have P left*"
- For the basic systems we have:

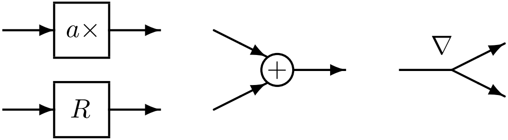
$$
\begin{array}{ccc}
\{P\} & x := t & \{P[t] + 1\} \\
\{P\} & \Delta & \{\langle P+1, P+1\rangle\} \\
\{\langle P, Q\rangle\} & \text{if}_b & \{\max\{P, Q\} + 1\}
\end{array}
$$

Stream Circuits

Smooth functions can be represented as *streams* $\mathbb{R}^\omega$

$$\sigma_y = [y(0), y'(0), y''(0), ...]$$

Stream circuits basic operations:

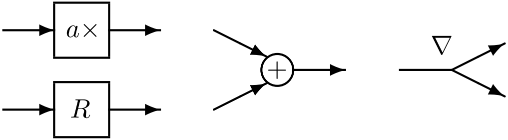


Stream Circuits

Smooth functions can be represented as *streams* $\mathbb{R}^\omega$

$$\sigma_y = [y(0), y'(0), y''(0), ...]$$

Stream circuits basic operations:



$$y' - y = u$$

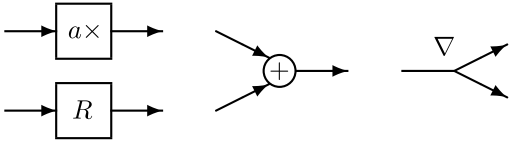
$$y(0) = 0$$

Stream Circuits

Smooth functions can be represented as *streams* $\mathbb{R}^\omega$

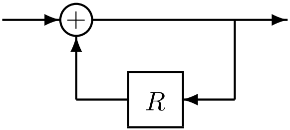
$$\sigma_y = [y(0), y'(0), y''(0), \ldots]$$

Stream circuits basic operations:



$$y' - y = u$$

$$y(0) = 0$$

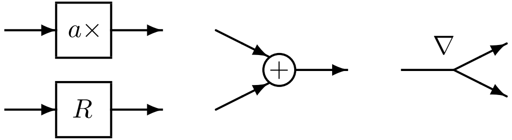


Stream Circuits

Smooth functions can be represented as *streams* $\mathbb{R}^\omega$

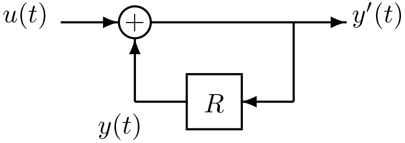
$$\sigma_y = [y(0), y'(0), y''(0), ...]$$

Stream circuits basic operations:



$$y' - y = u$$

$$y(0) = 0$$



Hoare Logic for Stream Circuits

- Pre order $(\mathbb{R}^\omega, =)$
- $H(f)(P) :\equiv f(P)$
- $\{P\}\ f\ \{Q\}$ means $f\langle P, Q\rangle$
  "*input P is related to output Q*"
- For the basic systems we have:

$$\begin{array}{ccc} \{P\} & a\times & \{aP\} \\ \{\langle P, Q\rangle\} & (+) & \{P+Q\} \\ \{P\} & \nabla & \{\langle P, P\rangle\} \\ \{P\} & R & \{0 * P\} \end{array}$$

Hoare Logic for Continuous Systems

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{f(P)\}}\ (\text{axiom}) \qquad \frac{\{P\}\ f\ \{Q\} \quad \{Q\}\ g\ \{R\}}{\{P\}\ g \circ f\ \{R\}}\ (\circ)$$

$$\frac{\{P\}\ f\ \{Q\} \quad \{R\}\ g\ \{S\}}{\{\langle P, R\rangle\}\ f \otimes g\ \{\langle Q, S\rangle\}}\ (\otimes) \qquad \frac{\{\langle P, Q\rangle\}\ f\ \{\langle R, Q\rangle\}}{\{P\}\ \mathsf{Tr}_{\mathcal{S}}(f)\ \{R\}}\ (\mathsf{Tr}_{\mathcal{S}})$$

$$\frac{P' = P \quad \{P\}\ f\ \{Q\} \quad Q = Q'}{\{P'\}\ f\ \{Q'\}}\ (\text{csq})$$

Hoare Logic for Continuous Systems

$$\frac{f \in \mathcal{S}_b}{\{P\}\ f\ \{f(P)\}}\ (\text{axiom}) \qquad \frac{\{P\}\ f\ \{Q\} \quad \{Q\}\ g\ \{R\}}{\{P\}\ g \circ f\ \{R\}}\ (\circ)$$

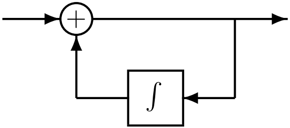
$$\frac{\{P\}\ f\ \{Q\} \quad \{R\}\ g\ \{S\}}{\{\langle P, R\rangle\}\ f \otimes g\ \{\langle Q, S\rangle\}}\ (\otimes) \qquad \frac{\{\langle P, Q\rangle\}\ f\ \{\langle R, Q\rangle\}}{\{P\}\ \mathsf{Tr}_{\mathcal{S}}(f)\ \{R\}}\ (\mathsf{Tr}_{\mathcal{S}})$$

$$\frac{P' = P \quad \{P\}\ f\ \{Q\} \quad Q = Q'}{\{P'\}\ f\ \{Q'\}}\ (\text{csq})$$

Hoare logic translates networks into (differential) equations!

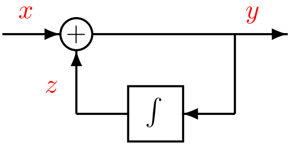
Example

$$\mathsf{Tr}(\langle 1, \textstyle\int \rangle \circ \nabla \circ (+))$$



Example

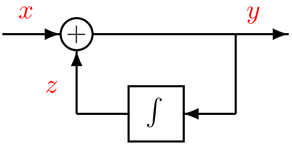
$$\mathsf{Tr}(\langle 1, \textstyle\int \rangle \circ \nabla \circ (+))$$



$$\dfrac{\dfrac{\dfrac{\{x+z\}\ \nabla\ \{\langle x+z, x+z\rangle\} \qquad \dfrac{\{x+z\}\ 1\ \{y\} \qquad \{x+z\}\ \int\ \{z\}}{\{\langle x+z, x+z\rangle\}\ \langle 1, \int\rangle\ \{\langle y, z\rangle\}}}{\{x+z\}\ \langle 1, \int\rangle \circ \nabla\ \{\langle y, z\rangle\}}\ (\circ)}{\{\langle x, z\rangle\}\ \langle 1, \int\rangle \circ \nabla \circ (+)\ \{\langle y, z\rangle\}}\ (\circ)}{\{x\}\ \mathsf{Tr}(\langle 1, \int\rangle \circ \nabla \circ (+))\ \{y\}}\ (\mathsf{Tr})$$

Example

$$\mathsf{Tr}(\langle 1, \textstyle\int \rangle \circ \nabla \circ (+))$$



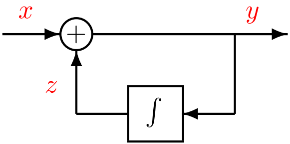
$$y = x + z$$

$$\int (x + z) = z$$

$$\dfrac{\dfrac{\dfrac{\{x+z\}\ \nabla\ \{\langle x+z, x+z\rangle\} \qquad \dfrac{\{x+z\}\ 1\ \{y\} \qquad \{x+z\} \int \{z\}}{\{\langle x+z, x+z\rangle\}\ \langle 1, \int \rangle\ \{\langle y, z\rangle\}}}{\{x+z\}\ \langle 1, \int \rangle \circ \nabla\ \{\langle y, z\rangle\}}\ (\circ)}{\{\langle x, z\rangle\}\ \langle 1, \int \rangle \circ \nabla \circ (+)\ \{\langle y, z\rangle\}}\ (\circ)}{\{x\}\ \mathsf{Tr}(\langle 1, \int \rangle \circ \nabla \circ (+))\ \{y\}}\ (\mathsf{Tr})$$

Example

$$\mathsf{Tr}(\langle 1, \textstyle\int \rangle \circ \nabla \circ (+))$$



$$y = x + z$$

$$\int (x + z) = z$$

$$\Rightarrow \quad \int y = y - x$$

$$\dfrac{\dfrac{\dfrac{\{x+z\}\ \nabla\ \{\langle x+z, x+z\rangle\} \quad \dfrac{\{x+z\}\ 1\ \{y\} \quad \{x+z\} \displaystyle\int \{z\}}{\{\langle x+z, x+z\rangle\}\ \langle 1, \displaystyle\int \rangle\ \{\langle y, z\rangle\}}}{\{x+z\}\ \langle 1, \displaystyle\int \rangle \circ \nabla\ \{\langle y, z\rangle\}}\ (\circ)}{\{\langle x, z\rangle\}\ \langle 1, \displaystyle\int \rangle \circ \nabla \circ (+)\ \{\langle y, z\rangle\}}\ (\circ)}{\{x\}\ \mathsf{Tr}(\langle 1, \displaystyle\int \rangle \circ \nabla \circ (+))\ \{y\}}\ (\mathsf{Tr})$$

Summary

- Abstraction of Hoare logic
  - Flowcharts programs
  - Pointer programs
  - ...
- *Future work*:
  - Total correctness
  - Higher order programs
  - Concurrency
- Continuous systems
  - Finding loop invariants = solving differential equations
- *Future work*:
  - Use modularity to partially solve diff equations
  - Reason with black boxes
  - Non linear systems