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)

London Theory Day, 24 April 2006

Outline

Outline

Overview

- **What**:

  Abstraction of modular reasoning about 'while programs'

Overview

- **What**:

  Abstraction of modular reasoning about 'while programs'

- **How**:

  Using system theory, tmc, and fixed-point theory

Overview

- **What**:

  Abstraction of modular reasoning about 'while programs'

- **How**:

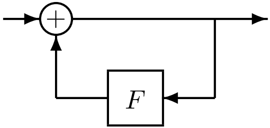
  Using system theory, tmc, and fixed-point theory

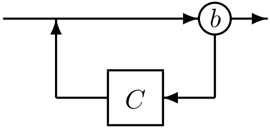
- **Why**:

  Develop Hoare-logic for dynamical systems

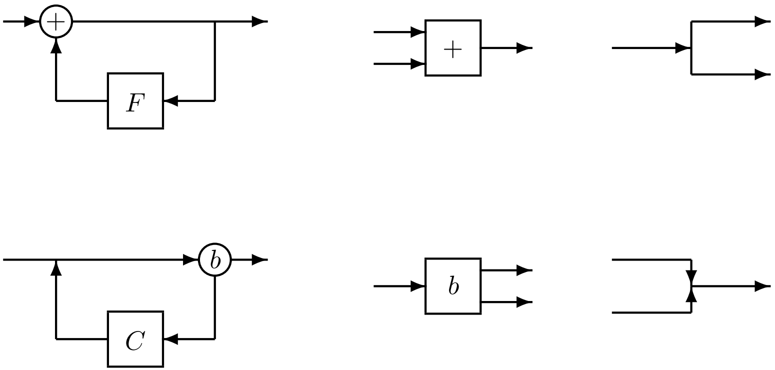
Outline

Network vs Flowcharts



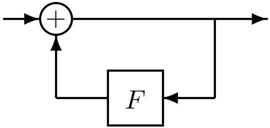


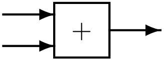
Network vs Flowcharts

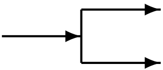


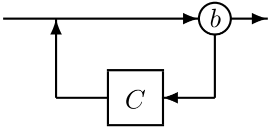
Network vs Flowcharts

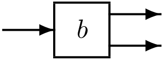
$$C^\infty \to (C^\infty \times C^\infty)$$

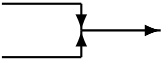








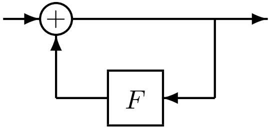


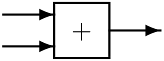


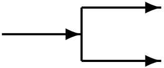
$$H \to (H \uplus H)$$

Network vs Flowcharts

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

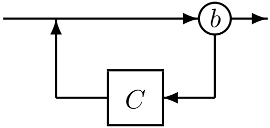


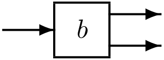


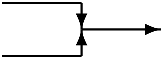


$$(C^\infty \times C^\infty) \to C^\infty$$

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







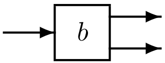
$$H \to (H \uplus H)$$

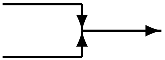
Bainbridge Duality

Exploit the duality between sum and product

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

Each flowchart corresponds to a network



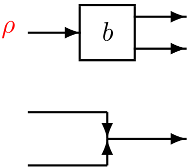


Bainbridge Duality

Exploit the duality between sum and product

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

Each flowchart corresponds to a network

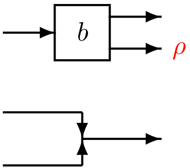


Bainbridge Duality

Exploit the duality between sum and product

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

Each flowchart corresponds to a network

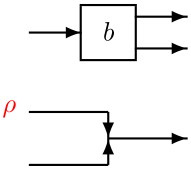


Bainbridge Duality

Exploit the duality between sum and product

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

Each flowchart corresponds to a network

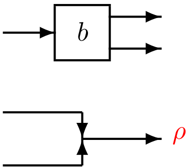


Bainbridge Duality

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

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

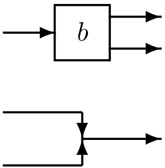
Each flowchart corresponds to a network

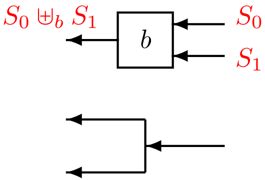
Bainbridge Duality

Exploit the duality between sum and product

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

Each flowchart corresponds to a network



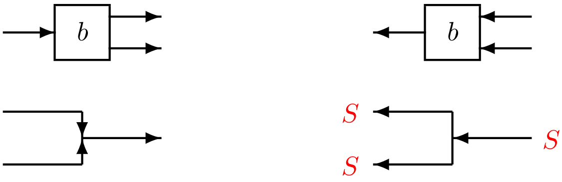


Bainbridge Duality

Exploit the duality between sum and product

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

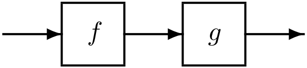
Each flowchart corresponds to a network



Monoidal Categories

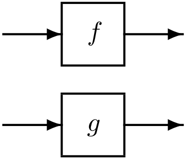
- **Sequential composition**: categorical composition
  $f : X \to Y$, $g : Y \to Z$ then $g \circ f : X \to Z$

$g \circ f$



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

$f \otimes g$

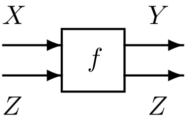


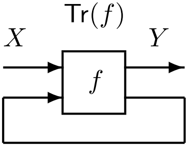
Traced Monoidal Categories

- **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 $\mathrm{cl}(M)$ 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 $\mathrm{cl}(\mathcal{S}_b) = \mathcal{S}_m$.

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, ...

Verification Category

Let Pos denote the category of posets and monotone mappings

Definition (Verification category)

A subcategory $\mathcal{V}$ of Pos is called a *verification category* if for any element $P \in X$ and morphism $f : (X \times Z) \to (Y \times Z)$ the set of pre-fixed points, i.e.

$$\{Q \ : \ \exists R \,.\, f\langle P, Q\rangle \sqsubseteq \langle R, Q\rangle\}$$

has a least element. We will denote such least element by $\mu_{f,P}$.

Verification Category

Let Pos denote the category of posets and monotone mappings

Definition (Verification category)

A subcategory $\mathcal{V}$ of Pos is called a *verification category* if for any element $P \in X$ and morphism $f : (X \times Z) \to (Y \times Z)$ the set of pre-fixed points, i.e.

$$\{Q \; : \; \exists R \,.\, f\langle P, Q\rangle \sqsubseteq \langle R, Q\rangle\}$$

has a least element. We will denote such least element by $\mu_{f,P}$.

By monotonicity of f, $\mu_{f,P}$ is also the least fixed point.

Verification Category: Intuition

Usual Hoare Logic	**Verification Categories**
Pre/Post-conditions	Points of posets
Logical implication	Partial order
Rule of consequence	Monotonicity
Strongest loop invariant	Least pre-fixed point

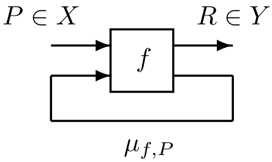
Verification Category and TMC

Lemma (A)

Any verification category $\mathcal{V}$ *gives rise to a traced monoidal category with trace defined as*

$$\mathsf{Tr}(f)(P) :\equiv R$$

for any morphism $f : (X \times Z) \to (Y \times Z)$, *where* R *is the unique element of* Y *such that* $f\langle P, \mu_{f,P}\rangle = \langle R, \mu_{f,P}\rangle$.



Propagation of Upper Bounds

Theorem (Soundness and completeness)

Let $\mathcal{V}$ be a verification category and $\mathcal{V}_b$ a set of basic morphisms spanning $\mathcal{V}_m$. The following set of propagation of upper bound rules is sound and complete for $\mathcal{V}$ with respect to $\mathcal{V}_b$

$$\frac{f \in \mathcal{V}_b}{f(P) \sqsubseteq f(P)} \text{ (axiom)}$$

$$\frac{P' \sqsubseteq P \quad f(P) \sqsubseteq Q \quad Q \sqsubseteq Q'}{f(P') \sqsubseteq Q'} \text{ (con)}$$

Propagation of Upper Bounds

Theorem (Soundness and completeness)

Let $\mathcal{V}$ be a verification category and $\mathcal{V}_b$ a set of basic morphisms spanning $\mathcal{V}_m$. The following set of propagation of upper bound rules is sound and complete for $\mathcal{V}$ with respect to $\mathcal{V}_b$

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

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

$$\frac{P' \sqsubseteq P \quad f(P) \sqsubseteq Q \quad Q \sqsubseteq Q'}{f(P') \sqsubseteq Q'}\ (\text{con})$$

Propagation of Upper Bounds

Theorem (Soundness and completeness)

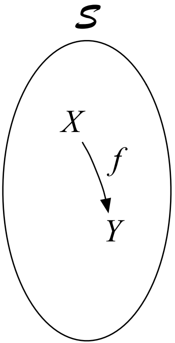
Let $\mathcal{V}$ be a verification category and $\mathcal{V}_b$ a set of basic morphisms spanning $\mathcal{V}_m$. The following set of propagation of upper bound rules is sound and complete for $\mathcal{V}$ with respect to $\mathcal{V}_b$

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

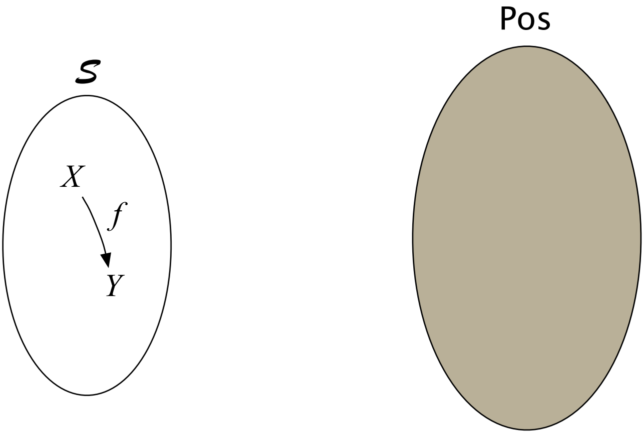
$$\frac{f(P) \sqsubseteq Q \quad g(R) \sqsubseteq S}{(f \times g)\langle P, R\rangle \sqsubseteq \langle Q, S\rangle}\ (\times) \qquad \frac{f\langle P, Q\rangle \sqsubseteq \langle R, Q\rangle}{\mathsf{Tr}_{\mathcal{V}}(f)(P) \sqsubseteq R}\ (\mathsf{Tr}_{\mathcal{V}})$$

$$\frac{P' \sqsubseteq P \quad f(P) \sqsubseteq Q \quad Q \sqsubseteq Q'}{f(P') \sqsubseteq Q'}\ (\text{con})$$

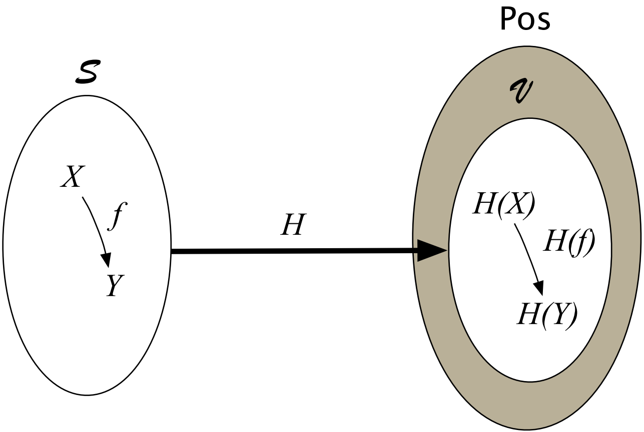
Abstract Hoare Triples



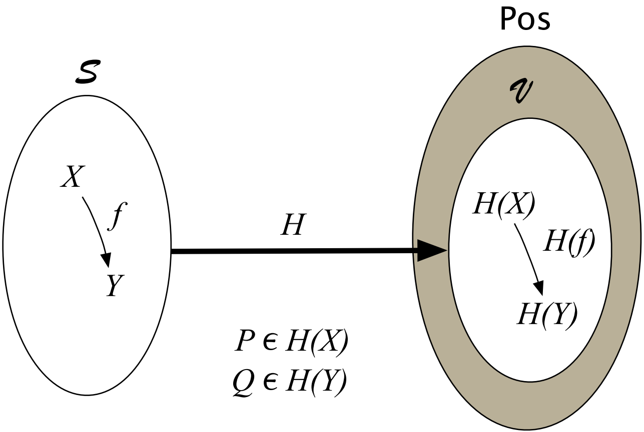
Abstract Hoare Triples



Abstract Hoare Triples



Abstract Hoare Triples



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

Abstract Hoare Triples

Definition (Verification functor)

A monoidal functor $H : \mathcal{S} \to \mathrm{Pos}$ is called a *verification functor* if

- image of H is a verification category
- H preserves traces (trace in image of H defined in Lemma (A))

Abstract Hoare Triples

Definition (Verification functor)

A monoidal functor $H : \mathcal{S} \to \mathsf{Pos}$ is called a *verification functor* if

- image of H is a verification category
- H preserves traces (trace in image of H defined in Lemma (A))

Let

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

Abstract Hoare Triples

Definition (Verification functor)

A monoidal functor $H : \mathcal{S} \to \mathsf{Pos}$ is called a *verification functor* if

- image of H is a verification category
- H preserves traces (trace in image of H defined in Lemma (A))

Let

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

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 $H : \mathcal{S} \to \mathsf{Pos}$:

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

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

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

Outline

Flowcharts

The embedding H is basically the power-set construction, so that

$$H(X \uplus Y) :\equiv H(X) \times H(Y)$$

On morphisms, we define:

- *Forward reasoning*

  $$H(f)(P) :\equiv \{y \in Y \ : \ \exists x \in P\ (f(x) = y)\}$$

- *Backward reasoning*

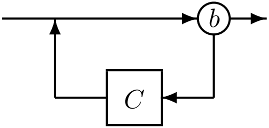
  $$H(f)(Q) :\equiv \{x \in X \ : \ f(x) \in Q\}$$

And if sets are described by formulas:

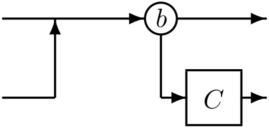
- $H(f)(\Phi) :\equiv \mathsf{SPC}(f, \Phi)$
- $H(f)(\Phi) :\equiv \mathsf{WPC}(f, \Phi)$

While Loop Rule

$\mathsf{while}_b(C)$

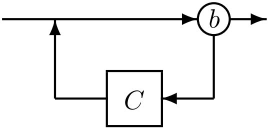


$(1 \uplus C) \circ \mathsf{if}_b \circ \Delta$

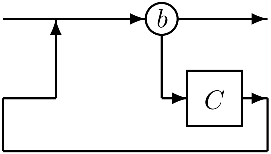


While Loop Rule

$\mathsf{while}_b(C)$

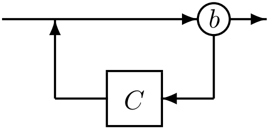


$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$

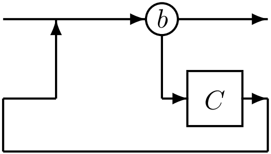


While Loop Rule

$\mathsf{while}_b(C)$



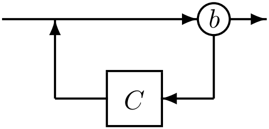
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$

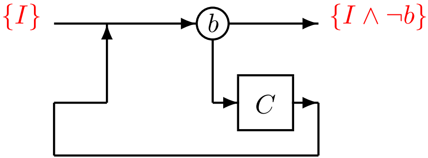


$$
\dfrac{
  \dfrac{
    \{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\}
    \qquad
    \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)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

While Loop Rule

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

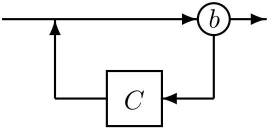


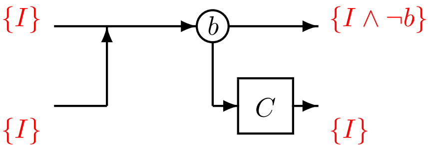


$$
\dfrac{\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\,(\uplus)}{\{I\}\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

While Loop Rule

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

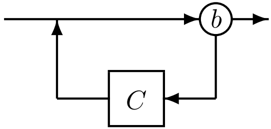




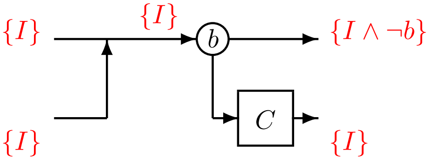
$$
\dfrac{
\dfrac{
\dfrac{
\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\}
\quad
\dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \quad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)
}{
\{I\}\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

While Loop Rule

$\mathsf{while}_b(C)$



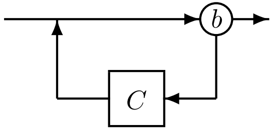
$\mathsf{Tr}((1 \uplus C) \circ \mathsf{if}_b \circ \Delta)$

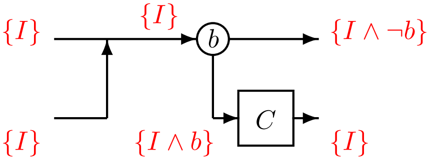


$$
\dfrac{
\dfrac{
\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad
\dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)
}{
\{I\}\ (1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

While Loop Rule

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

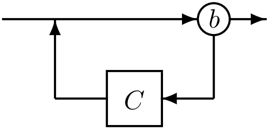




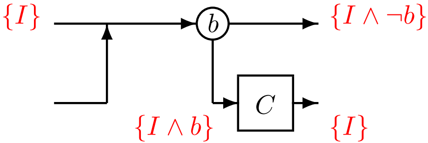
$$
\dfrac{\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \qquad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \qquad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\,(\uplus)}{\{I\}\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

While Loop Rule

$\mathsf{while}_b(C)$



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



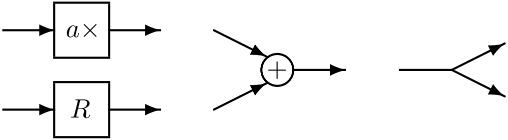
$$
\dfrac{\dfrac{\{I\}\ \mathsf{if}_b\ \{\langle I \wedge \neg b, I \wedge b\rangle\} \quad \dfrac{\{I \wedge \neg b\}\ 1\ \{I \wedge \neg b\} \quad \{I \wedge b\}\ C\ \{I\}}{\{\langle I \wedge \neg b, I \wedge b\rangle\}\ 1 \uplus C\ \{\langle I \wedge \neg b, I\rangle\}}\ (\uplus)}{\dfrac{\{I\}\ (1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

Stream Circuits

Smooth functions can be represented as *streams*

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

Stream circuits basic operations:

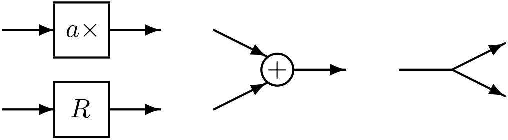


Stream Circuits

Smooth functions can be represented as *streams*

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

Stream circuits basic operations:



$$y' - y = u$$

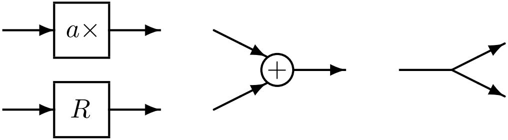
$$y(0) = 0$$

Stream Circuits

Smooth functions can be represented as *streams*

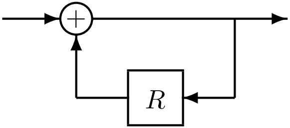
$$\sigma_y = [y(0), y'(0), y''(0), \ldots]$$

Stream circuits basic operations:



$$y' - y = u$$

$$y(0) = 0$$

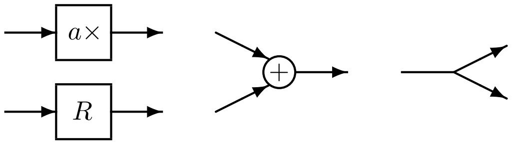


Stream Circuits

Smooth functions can be represented as *streams*

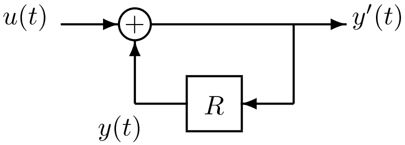
$$\sigma_y = [y(0), y'(0), y''(0), ...]$$

Stream circuits basic operations:



$$y' - y = u$$

$$y(0) = 0$$



Embedding

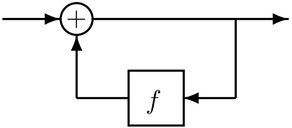
- Stream circuits already have Cartesian product as the monoidal structure, the embedding H into Pos has to respect that

- Our verification embedding of stream circuits is as follows:
  - Define $H(\Sigma)$ as the poset of finite approximations (prefixes) of elements in Σ, with the ordering $s \preceq t$, if t is an extension of s
  - For morphisms (stream circuits) $f : X \to Y$ define $H(f)(t) :\equiv \{f(t * \tau) \ : \ \tau \in X\}$

Embedding

- Stream circuits already have Cartesian product as the monoidal structure, the embedding H into Pos has to respect that
- Our verification embedding of stream circuits is as follows:
  - Define $H(\Sigma)$ as the poset of finite approximations (prefixes) of elements in Σ, with the ordering $s \preceq t$, if t is an extension of s
  - For morphisms (stream circuits) $f : X \to Y$ define $H(f)(t) :\equiv \{f(t * \tau) \;:\; \tau \in X\}$

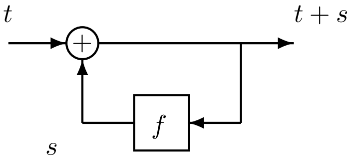
$$\frac{\{t+s\}\ f\ \{s\}}{\{t\}\ \mathsf{FB}(f)\ \{t+s\}}\ (\mathsf{Fd})$$



Embedding

- Stream circuits already have Cartesian product as the monoidal structure, the embedding H into Pos has to respect that
- Our verification embedding of stream circuits is as follows:
  - Define $H(\Sigma)$ as the poset of finite approximations (prefixes) of elements in Σ, with the ordering $s \preceq t$, if t is an extension of s
  - For morphisms (stream circuits) $f : X \to Y$ define $H(f)(t) :\equiv \{f(t * \tau) \ : \ \tau \in X\}$

$$\frac{\{t+s\}\ f\ \{s\}}{\{t\}\ \mathsf{FB}(f)\ \{t+s\}}\ (\mathsf{Fd})$$



Conclusions and Future Work

- Other instantiations (in flowcharts):
  - Pointer programs
  - Total correctness
- Other instantiations (in stream circuits):
  - Boundedness
  - Relative stability
- Related work
  - Dijkstra's predicate transformer
  - Kozen's KAT (Kleene Algebras with Test)
  - Abramsky's specification categories
  - Bloom and Esik on iteration theory
  - Gurevich's existential fixed-point logic