Modal properties of recursive programs
Work in progress

Paul Blain Levy

University of Birmingham

September 24, 2008

Summary

- A language
- Roscoe's Seeing Beyond Divergence model
- A model of lower bisimilarity
- What's really happening: modal logic

A basic language

Syntax

Let $\mathcal{A}$ be a countable alphabet.

$$M ::= \ \ \texttt{print}\, c.\, M \ \mid\ \texttt{x} \ \mid\ \texttt{rec}\, \texttt{x}.\, M \ \mid\ \texttt{choose}_{n \in \mathbb{N}}\, M_n \qquad c \in \mathcal{A}$$

Many other things can be added.

A basic language

Syntax

Let $\mathcal{A}$ be a countable alphabet.

$$M ::= \quad \texttt{print}\ c.\ M \mid \texttt{x} \mid \texttt{rec}\ \texttt{x}.\ M \mid \texttt{choose}_{n \in \mathbb{N}}\ M_n \qquad c \in \mathcal{A}$$

Many other things can be added.

Small-step semantics

$$\begin{array}{rcll} \texttt{print}\ c.\ M & \overset{c}{\rightsquigarrow} & M & \\ \texttt{rec}\ \texttt{x}.\ M & \rightsquigarrow & M[\texttt{rec}\ \texttt{x}.\ M/\texttt{x}] & \\ \texttt{choose}_{n \in \mathbb{N}}\ M_n & \rightsquigarrow & M_{\hat{n}} & \hat{n} \in \mathbb{N} \end{array}$$

A basic language

Syntax

Let $\mathcal{A}$ be a countable alphabet.

$$M ::= \quad \texttt{print}\ c.\ M \mid \texttt{x} \mid \texttt{rec}\ \texttt{x}.\ M \mid \texttt{choose}_{n\in\mathbb{N}}\ M_n \qquad c \in \mathcal{A}$$

Many other things can be added.

Small-step semantics

$$\begin{array}{rcll} \texttt{print}\ c.\ M & \overset{c}{\leadsto} & M & \\ \texttt{rec x.}\ M & \leadsto & M[\texttt{rec x.}\ M/\texttt{x}] & \\ \texttt{choose}_{n\in\mathbb{N}}\ M_n & \leadsto & M_{\hat{n}} & \hat{n} \in \mathbb{N} \end{array}$$

A program either

- prints a finite string, then diverges
- or prints an infinite string.

Medium step semantics

Convergence $M \overset{c}{\searrow} N$ defined inductively

$$\frac{}{\texttt{print}\ c.\ M \overset{c}{\searrow} M} \qquad \frac{M[\texttt{rec x.}\ M/\texttt{x}] \overset{c}{\searrow} N}{\texttt{rec x.}\ M \overset{c}{\searrow} N}$$

$$\frac{M_{\hat{n}} \overset{c}{\searrow} N}{\texttt{choose}_{n\in\mathbb{N}}\ M_n \overset{c}{\searrow} N}\ \hat{n} \in \mathbb{N}$$

Divergence $M \Uparrow$ defined coinductively

$$\frac{M[\texttt{rec x.}\ M/\texttt{x}] \Uparrow}{\texttt{rec x.}\ M \Uparrow} \qquad \frac{M_{\hat{n}} \Uparrow}{\texttt{choose}_{n\in\mathbb{N}}\ M_n \Uparrow}\ \hat{n} \in \mathbb{N}$$

Medium step semantics

Convergence $M \overset{c}{\searrow} N$ defined inductively

$$\frac{}{\mathtt{print}\ c.\ M \overset{c}{\searrow} M} \qquad \frac{M[\mathtt{rec\ x.}\ M/\mathtt{x}] \overset{c}{\searrow} N}{\mathtt{rec\ x.}\ M \overset{c}{\searrow} N}$$

$$\frac{M_{\hat{n}} \overset{c}{\searrow} N}{\mathtt{choose}_{n \in \mathbb{N}}\ M_n \overset{c}{\searrow} N}\ \hat{n} \in \mathbb{N}$$

Divergence $M \Uparrow$ defined coinductively

$$\frac{M[\mathtt{rec\ x.}\ M/\mathtt{x}] \Uparrow}{\mathtt{rec\ x.}\ M \Uparrow} \qquad \frac{M_{\hat{n}} \Uparrow}{\mathtt{choose}_{n \in \mathbb{N}}\ M_n \Uparrow}\ \hat{n} \in \mathbb{N}$$

We have

- $M \overset{c}{\searrow} N$ iff $M \rightsquigarrow^* \overset{c}{\rightsquigarrow} N$
- $M \Uparrow$ iff $M \rightsquigarrow^\omega$

Well-pointed semantics

We are interested in denotational models where a term denotes a function from environments.

Well-pointed semantics

We are interested in denotational models where a term denotes a function from environments.

To interpret recursion, we need an appropriate way of finding a fixpoint of an endofunction.

For example, least fixpoint or greatest fixpoint.

Well-pointed semantics

We are interested in denotational models where a term denotes a function from environments.

To interpret recursion, we need an appropriate way of finding a fixpoint of an endofunction.

For example, least fixpoint or greatest fixpoint.

Roscoe's Seeing Beyond Divergence model uses a reflected fixpoint.

Finite traces, divergences, and infinite traces

A closed command M has

- a set $T(M) \subseteq \mathcal{A}^*$ of finite traces

  M can print `hello`

Finite traces, divergences, and infinite traces

A closed command M has

- a set $T(M) \subseteq \mathcal{A}^*$ of finite traces

  M can print `hello`

- a set $D(M) \subseteq \mathcal{A}^*$ of divergences

  M can print `hello` then diverge

Finite traces, divergences, and infinite traces

A closed command M has

- a set $T(M) \subseteq \mathcal{A}^*$ of finite traces

  M can print `hello`

- a set $D(M) \subseteq \mathcal{A}^*$ of divergences

  M can print `hello` then diverge

- a set $I(M) \subseteq \mathcal{A}^\omega$ of infinite traces

  M can print `helloworldworldworld` ...

Seeing Beyond Divergence (1)

Definition of $[M]_{\mathcal{N}}$

- the set of finite traces of M, together with extensions of divergences
- the set of extensions of divergences of M
- the set of infinite traces of M, together with extensions of divergences

This semantics is divergence strict.

Seeing Beyond Divergence (1)

Definition of $[M]_{\mathcal{N}}$

- the set of finite traces of M, together with extensions of divergences
- the set of extensions of divergences of M
- the set of infinite traces of M, together with extensions of divergences

This semantics is divergence strict.

To model recursion, we take the greatest fixpoint. (Reverse ordering is the upper powerdomain.)

Seeing Beyond Divergence (2)

Definition of $[M]_{\mathsf{SBD}}$

- the set of finite traces of M
- the set of divergences of M
- the set of infinite traces, together with limits of divergences (called "ω-divergences")

Seeing Beyond Divergence (2)

Definition of $[M]_{\text{SBD}}$

- the set of finite traces of M
- the set of divergences of M
- the set of infinite traces, together with limits of divergences (called "ω-divergences")

To model recursion:

- first compute the greatest fixpoint wrt $[\,]_{\mathcal{N}}$, giving a "diamond": a complete lattice of possible solutions that are $[\,]_{\mathcal{N}}$ equivalent
- then compute the least fixpoint wrt $[\,]_{\text{SBD}}$ within that complete lattice.

This is called the reflected fixpoint.

Seeing Beyond Divergence: further points

Continuity

For any recursion, the second phase of fixpoint calculation converges in ω steps.

Seeing Beyond Divergence: further points

Continuity

For any recursion, the second phase of fixpoint calculation converges in ω steps.

This is because every term denotes a function that is continuous on each diamond.

Seeing Beyond Divergence: further points

Continuity

For any recursion, the second phase of fixpoint calculation converges in ω steps.

This is because every term denotes a function that is continuous on each diamond.

Lexicographic Ordering

The reflected fixpoint is the least prefixed point wrt the lexicographic ordering:

- reverse inclusion for $[]_{\mathcal{N}}$
- then inclusion for $[]_{\mathsf{SBD}}$

Seeing Beyond Divergence: further points

Continuity

For any recursion, the second phase of fixpoint calculation converges in ω steps.

This is because every term denotes a function that is continuous on each diamond.

Lexicographic Ordering

The reflected fixpoint is the least prefixed point wrt the lexicographic ordering:

- reverse inclusion for $[]_{\mathcal{N}}$
- then inclusion for $[]_{\mathsf{SBD}}$

Terms are not monotone wrt this ordering.

Lower And Convex Bisimulation

Let $\mathcal{R}$ be a binary relation on closed terms.

It is a lower simulation when $M \mathrel{\mathcal{R}} M'$ and $M \overset{\mathsf{c}}{\searrow} N$ implies $\exists N'$ such that $M' \overset{\mathsf{c}}{\searrow} N'$ and $N \mathrel{\mathcal{R}} N'$.

It is a lower bisimulation when $\mathcal{R}$ and $\mathcal{R}^{\mathsf{op}}$ are lower simulations.

It is a convex bisimulation when moreover $M \mathrel{\mathcal{R}} M'$ implies $M \Uparrow \Leftrightarrow M' \Uparrow$.

The greatest lower bisimulation is called lower bisimilarity.

Lower And Convex Bisimulation

Let $\mathcal{R}$ be a binary relation on closed terms.

It is a lower simulation when $M \mathrel{\mathcal{R}} M'$ and $M \overset{c}{\searrow} N$ implies $\exists N'$ such that $M' \overset{c}{\searrow} N'$ and $N \mathrel{\mathcal{R}} N'$.

It is a lower bisimulation when $\mathcal{R}$ and $\mathcal{R}^{\mathsf{op}}$ are lower simulations.

It is a convex bisimulation when moreover $M \mathrel{\mathcal{R}} M'$ implies $M \Uparrow \Leftrightarrow M' \Uparrow$.

The greatest lower bisimulation is called lower bisimilarity.

Two closed terms M, M' are lower bisimilar

Lower And Convex Bisimulation

Let $\mathcal{R}$ be a binary relation on closed terms.

It is a lower simulation when $M \mathrel{\mathcal{R}} M'$ and $M \overset{c}{\searrow} N$ implies $\exists N'$ such that $M' \overset{c}{\searrow} N'$ and $N \mathrel{\mathcal{R}} N'$.

It is a lower bisimulation when $\mathcal{R}$ and $\mathcal{R}^{\mathrm{op}}$ are lower simulations.

It is a convex bisimulation when moreover $M \mathrel{\mathcal{R}} M'$ implies $M \Uparrow \Leftrightarrow M' \Uparrow$.

The greatest lower bisimulation is called lower bisimilarity.

Two closed terms M, M' are lower bisimilar

- iff they have the same anamorphic image

Lower And Convex Bisimulation

Let $\mathcal{R}$ be a binary relation on closed terms.

It is a lower simulation when $M \; \mathcal{R} \; M'$ and $M \overset{\mathsf{c}}{\searrow} N$ implies $\exists N'$ such that $M' \overset{\mathsf{c}}{\searrow} N'$ and $N \; \mathcal{R} \; N'$.

It is a lower bisimulation when $\mathcal{R}$ and $\mathcal{R}^{\mathsf{op}}$ are lower simulations.

It is a convex bisimulation when moreover $M \; \mathcal{R} \; M'$ implies $M \Uparrow \Leftrightarrow M' \Uparrow$.

The greatest lower bisimulation is called lower bisimilarity.

Two closed terms M, M' are lower bisimilar

- iff they have the same anamorphic image
- iff there is a strategy for the bisimilarity game between them (Opponent moves first, and in each move can play either left or right)

Lower And Convex Bisimulation

Let $\mathcal{R}$ be a binary relation on closed terms.

It is a lower simulation when $M \; \mathcal{R} \; M'$ and $M \overset{c}{\searrow} N$ implies $\exists N'$ such that $M' \overset{c}{\searrow} N'$ and $N \; \mathcal{R} \; N'$.

It is a lower bisimulation when $\mathcal{R}$ and $\mathcal{R}^{\text{op}}$ are lower simulations.

It is a convex bisimulation when moreover $M \; \mathcal{R} \; M'$ implies $M \Uparrow \Leftrightarrow M' \Uparrow$.

The greatest lower bisimulation is called lower bisimilarity.

Two closed terms M, M' are lower bisimilar

- iff they have the same anamorphic image
- iff there is a strategy for the bisimilarity game between them (Opponent moves first, and in each move can play either left or right)
- iff they satisfy the same formulas in Hennessy-Milner logic

$$P ::= \Diamond a.P \mid \bigvee_{j \in J} P_i \mid \neg P$$

Model of bisimilarity: nested simulation

A 2-nested (lower) simulation is a simulation contained in mutual similarity.

Model of bisimilarity: nested simulation

A 2-nested (lower) simulation is a simulation contained in mutual similarity.

Two closed terms M, M' are related by 2-nested similarity

- iff there is a strategy for the 2-nested simulation game (Opponent starts on the left, and can switch once)
- iff $M \vDash P$ implies $M' \vDash P$ whenever P has at most one level of negation.

Model of bisimilarity: nested simulation

A 2-nested (lower) simulation is a simulation contained in mutual similarity.

Two closed terms M, M' are related by 2-nested similarity

- iff there is a strategy for the 2-nested simulation game (Opponent starts on the left, and can switch once)
- iff $M \models P$ implies $M' \models P$ whenever P has at most one level of negation.

A 3-nested lower simulation is a simulation contained in mutual 2-nested similarity. And so through all countable ordinals.

Model of bisimilarity: nested simulation

A 2-nested (lower) simulation is a simulation contained in mutual similarity.

Two closed terms M, M' are related by 2-nested similarity

- iff there is a strategy for the 2-nested simulation game (Opponent starts on the left, and can switch once)
- iff $M \vDash P$ implies $M' \vDash P$ whenever P has at most one level of negation.

A 3-nested lower simulation is a simulation contained in mutual 2-nested similarity. And so through all countable ordinals.

The intersection of n-nested similarity for $n < \omega_1$ is lower bisimilarity.

Model of bisimilarity: nested simulation

A 2-nested (lower) simulation is a simulation contained in mutual similarity.

Two closed terms M, M' are related by 2-nested similarity

- iff there is a strategy for the 2-nested simulation game (Opponent starts on the left, and can switch once)
- iff $M \vDash P$ implies $M' \vDash P$ whenever P has at most one level of negation.

A 3-nested lower simulation is a simulation contained in mutual 2-nested similarity. And so through all countable ordinals.

The intersection of n-nested similarity for $n < \omega_1$ is lower bisimilarity.

Theorem Up to lower bisimilarity, $\mathtt{rec\ x.}\ M$ is the lexicographically least prefixed point for $N \mapsto N[\mathtt{rec\ x.}\ M/\mathtt{x}]$ wrt this sequence of precongruences.

Digression: other models of bisimilarity (1)

Synchronization Trees

Milner and Winskel have studied semantics in which

- a closed term denotes a synchronization tree of possible behaviours.
- an open term denotes a function (actually a functor) from synchronization trees to synchronization trees.

Very intensional: the idempotency law $M \texttt{ or } M = M$ is not validated.

Digression: other models of bisimilarity (1)

Synchronization Trees

Milner and Winskel have studied semantics in which

- a closed term denotes a synchronization tree of possible behaviours.
- an open term denotes a function (actually a functor) from synchronization trees to synchronization trees.

Very intensional: the idempotency law M `or` $M = M$ is not validated.

Lower bisimilarity is studied as a relation on the trees, but this is not part of the denotational semantics.

Digression: other models of bisimilarity (2)

Abramsky's domain equation for bisimulation

Abramsky gave a semantics (for finite nondeterminism) using a domain equation involving the convex powerdomain.

Digression: other models of bisimilarity (2)

Abramsky's domain equation for bisimulation

Abramsky gave a semantics (for finite nondeterminism) using a domain equation involving the convex powerdomain.

For nondivergent terms, denotational equivalence coincides with lower bisimilarity.

Digression: other models of bisimilarity (2)

Abramsky's domain equation for bisimulation

Abramsky gave a semantics (for finite nondeterminism) using a domain equation involving the convex powerdomain.

For nondivergent terms, denotational equivalence coincides with lower bisimilarity.

But in general, terms may have the same denotation without being lower bisimilar.

Digression: other models of bisimilarity (2)

Abramsky's domain equation for bisimulation

Abramsky gave a semantics (for finite nondeterminism) using a domain equation involving the convex powerdomain.

For nondivergent terms, denotational equivalence coincides with lower bisimilarity.

But in general, terms may have the same denotation without being lower bisimilar.

This is inevitable in least fixpoint semantics.

Some general points

We want to understand various denotational models of nondeterministic languages with recursion.

Some general points

We want to understand various denotational models of nondeterministic languages with recursion.

In each of these models, there is a set C of elements and an ordinal sequence of preorders on C.

The intersection of these preorders is discrete.

Some general points

We want to understand various denotational models of nondeterministic languages with recursion.

In each of these models, there is a set C of elements and an ordinal sequence of preorders on C.

The intersection of these preorders is discrete.

Terms are continuous wrt some of these orderings, and ω_1-continuous wrt others.

Some general points

We want to understand various denotational models of nondeterministic languages with recursion.

In each of these models, there is a set C of elements and an ordinal sequence of preorders on C.

The intersection of these preorders is discrete.

Terms are continuous wrt some of these orderings, and ω_1-continuous wrt others.

A recursion is interpreted, in these models, as the least prefixed point wrt the induced lexicographic partial order.

Some general points

We want to understand various denotational models of nondeterministic languages with recursion.

In each of these models, there is a set C of elements and an ordinal sequence of preorders on C.

The intersection of these preorders is discrete.

Terms are continuous wrt some of these orderings, and ω_1-continuous wrt others.

A recursion is interpreted, in these models, as the least prefixed point wrt the induced lexicographic partial order.

Warning Terms are not even monotone wrt this lexicographic partial order.

Modal logic with may and must

Modal logic in the style of Hennessy-Milner:

$$A ::= \neg A \mid \bigvee_{i \in I} A_i \mid \bigwedge_{i \in I} A_i \mid \Diamond a.A \mid \Box_{s \in \mathcal{A}^*} A_s$$

where I is bounded by some suitable cardinal.

Modal logic with may and must

Modal logic in the style of Hennessy-Milner:

$$A ::= \neg A \mid \bigvee_{i \in I} A_i \mid \bigwedge_{i \in I} A_i \mid \Diamond a.A \mid \Box_{s \in \mathcal{A}^*} A_s$$

where I is bounded by some suitable cardinal.

Meaning of $\Diamond$

$\Diamond a.A$ means it is possible that a will be printed and then A will be satisfied.

Modal logic with may and must

Modal logic in the style of Hennessy-Milner:

$$A ::= \neg A \mid \bigvee_{i \in I} A_i \mid \bigwedge_{i \in I} A_i \mid \Diamond a.A \mid \Box_{s \in \mathcal{A}^*} A_s$$

where I is bounded by some suitable cardinal.

Meaning of $\Diamond$

$\Diamond a.A$ means it is possible that a will be printed and then A will be satisfied.

Meaning of $\Box$

$\Box_{s \in \mathcal{A}^*} A_s$ means a time will come when A_s will be satisfied, where s is the string printed between now and then.

Contextual preorders

We say $M \preccurlyeq(A)\ M'$ when

- for every context $\mathcal{C}$, if $\mathcal{C}[M] \vDash A$ then $\mathcal{C}[M'] \vdash A$.

Contextual preorders

We say $M \preccurlyeq(A) M'$ when

- for every context $\mathcal{C}$, if $\mathcal{C}[M] \vDash A$ then $\mathcal{C}[M'] \vdash A$.

This is a preorder, and we can speak of the $\preccurlyeq(A)$ equivalence class of M.

Contextual preorders

We say $M \preccurlyeq(A)\ M'$ when

- for every context $\mathcal{C}$, if $\mathcal{C}[M] \vDash A$ then $\mathcal{C}[M'] \vdash A$.

This is a preorder, and we can speak of the $\preccurlyeq(A)$ equivalence class of M.

More generally, we say $M \preccurlyeq(\{A_i\}_{i \in I})\ M$ when

- for every context $\mathcal{C}$ and $i \in I$, if $\mathcal{C}[M] \vDash A_i$ then $\mathcal{C}[M'] \vDash A_i$.

When does a recursive program satisfy $\Diamond$?

We want to know when $\mathcal{C}[\texttt{rec x.}\ M]$ satisfies $B \stackrel{\text{def}}{=} \Diamond a.A$.

When does a recursive program satisfy $\Diamond$?

We want to know when $\mathcal{C}[\texttt{rec x.}\ M]$ satisfies $B \stackrel{\text{def}}{=} \Diamond a.A$.

Let U be the $\preccurlyeq(A)$ equivalence class of $\texttt{rec x.}\ M$.

Clearly $\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on U, monotone wrt $\preccurlyeq(B)$.

When does a recursive program satisfy $\Diamond$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Diamond a.A$.

Let U be the $\preccurlyeq(A)$ equivalence class of $\texttt{rec x. } M$.

Clearly $\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on U, monotone wrt $\preccurlyeq(B)$.

Theorem

Suppose U has a $\preccurlyeq(B)$ least element of U, call it N.

When does a recursive program satisfy $\Diamond$?

We want to know when $\mathcal{C}[\texttt{rec x.}\ M]$ satisfies $B \stackrel{\text{def}}{=} \Diamond a.A$.

Let U be the $\preccurlyeq(A)$ equivalence class of $\texttt{rec x.}\ M$.

Clearly $\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on U, monotone wrt $\preccurlyeq(B)$.

Theorem

Suppose U has a $\preccurlyeq(B)$ least element of U, call it N.

Then $\mathcal{C}[\texttt{rec x.}\ M] \vDash \Diamond a.A$ iff there exists $n \in \mathbb{N}$ such that $\mathcal{C}[\theta_M^n N] \vDash \Diamond a.A$.

When does a recursive program satisfy $\Diamond$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Diamond a.A$.

Let U be the $\preccurlyeq(A)$ equivalence class of $\texttt{rec x. } M$.

Clearly $\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on U, monotone wrt $\preccurlyeq(B)$.

Theorem

Suppose U has a $\preccurlyeq(B)$ least element of U, call it N.

Then $\mathcal{C}[\texttt{rec x. } M] \vDash \Diamond a.A$ iff there exists $n \in \mathbb{N}$ such that $\mathcal{C}[\theta_M^n N] \vDash \Diamond a.A$.

The special case that $A = \text{True}$ gives lower powerdomain semantics.

When does a recursive program satisfy $\Box$?

We want to know when $\mathcal{C}[\texttt{rec x.}\ M]$ satisfies $B \stackrel{\mathrm{def}}{=} \Box_{s \in \mathcal{A}^*} A_s$.

When does a recursive program satisfy $\Box$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Box_{s \in \mathcal{A}^*} A_s$.

Let U_s be the $\preccurlyeq(A_s)$ equivalence class of $\texttt{rec x.}M$.

$\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on $\bigcap_{s \in \mathcal{A}^*} U_s$ monotone wrt $\preccurlyeq(B)$.

When does a recursive program satisfy $\Box$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Box_{s \in \mathcal{A}^*} A_s$.

Let U_s be the $\preccurlyeq(A_s)$ equivalence class of $\texttt{rec x.}M$.

$\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on $\bigcap_{s \in \mathcal{A}^*} U_s$ monotone wrt $\preccurlyeq(B)$.

Conjecture

Define a sequence $(N_\alpha)_{\alpha < \omega_1}$ contained in U, increasing wrt $\preccurlyeq(B)$

$$
\begin{aligned}
N_0 &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ least element of } \textstyle\bigcap_{s \in \mathcal{A}^*} U_s\text{, assuming it exists} \\
N_{\beta+1} &\stackrel{\text{def}}{=} \theta_M N_\beta \\
N_\gamma &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ supremum of } (N_\alpha)_{\alpha<\gamma}\text{, assuming it exists}
\end{aligned}
$$

When does a recursive program satisfy $\Box$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Box_{s \in \mathcal{A}^*} A_s$.

Let U_s be the $\preccurlyeq(A_s)$ equivalence class of $\texttt{rec x.}M$.

$\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on $\bigcap_{s \in \mathcal{A}^*} U_s$ monotone wrt $\preccurlyeq(B)$.

Conjecture

Define a sequence $(N_\alpha)_{\alpha < \omega_1}$ contained in U, increasing wrt $\preccurlyeq(B)$

$$
\begin{aligned}
N_0 &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ least element of } \textstyle\bigcap_{s \in \mathcal{A}^*} U_s, \text{ assuming it exists} \\
N_{\beta+1} &\stackrel{\text{def}}{=} \theta_M N_\beta \\
N_\gamma &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ supremum of } (N_\alpha)_{\alpha < \gamma}, \text{ assuming it exists}
\end{aligned}
$$

Then $\mathcal{C}[\texttt{rec x. } M] \vDash B$ iff there exists $\alpha < \omega_1$ such that $\mathcal{C}[N_\alpha] \vDash B$.

When does a recursive program satisfy $\Box$?

We want to know when $\mathcal{C}[\texttt{rec x. } M]$ satisfies $B \stackrel{\text{def}}{=} \Box_{s \in \mathcal{A}^*} A_s$.

Let U_s be the $\preccurlyeq(A_s)$ equivalence class of $\texttt{rec x.}M$.

$\theta_M : N \mapsto M[N/\texttt{x}]$ is an endofunction on $\bigcap_{s \in \mathcal{A}^*} U_s$ monotone wrt $\preccurlyeq(B)$.

Conjecture

Define a sequence $(N_\alpha)_{\alpha < \omega_1}$ contained in U, increasing wrt $\preccurlyeq(B)$

$$
\begin{aligned}
N_0 &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ least element of } \textstyle\bigcap_{s \in \mathcal{A}^*} U_s, \text{ assuming it exists} \\
N_{\beta+1} &\stackrel{\text{def}}{=} \theta_M N_\beta \\
N_\gamma &\stackrel{\text{def}}{=} \preccurlyeq(B) \text{ supremum of } (N_\alpha)_{\alpha<\gamma}, \text{ assuming it exists}
\end{aligned}
$$

Then $\mathcal{C}[\texttt{rec x. } M] \vDash B$ iff there exists $\alpha < \omega_1$ such that $\mathcal{C}[N_\alpha] \vDash B$.

The special case that $A =$ True gives upper powerdomain semantics.

Seeing Beyond Divergence Revisited

Let t be a finite trace, divergence or infinite trace, and let s be a finite prefix.

We say that M semi-validates $s \sqsubseteq t$ when it is possible for M to prove s and never refute t.

Seeing Beyond Divergence Revisited

Let t be a finite trace, divergence or infinite trace, and let s be a finite prefix.

We say that M semi-validates $s \sqsubseteq t$ when it is possible for M to prove s and never refute t.

Two terms M, M' are SBD equivalent when they semi-validate the same prefixes.

Seeing Beyond Divergence Revisited

Let t be a finite trace, divergence or infinite trace, and let s be a finite prefix.

We say that M semi-validates $s \sqsubseteq t$ when it is possible for M to prove s and never refute t.

Two terms M, M' are SBD equivalent when they semi-validate the same prefixes.

M validates $\texttt{hello} \sqsubseteq \texttt{helloworldworldworld}\ldots$ when

$$M \models \Diamond\texttt{hello}.\ \neg\Box s.s \not\sqsubseteq \texttt{worldworldworld}\ldots$$

We can now see how the two-part calculation of a recursion arises.

Directions

We are starting to understand iterated fixpoint denotations of recursive programs by thinking about the modal properties that they satisfy.

Directions

We are starting to understand iterated fixpoint denotations of recursive programs by thinking about the modal properties that they satisfy.

Studying this syntactically is awkward, since

- we do not know what the $\preccurlyeq(A)$ preorders actually are
- the required suprema might not exist.

Directions

We are starting to understand iterated fixpoint denotations of recursive programs by thinking about the modal properties that they satisfy.

Studying this syntactically is awkward, since

- we do not know what the $\preccurlyeq(A)$ preorders actually are
- the required suprema might not exist.

Perhaps a deductively closed set of modal formulas could serve as a generalized program?

Another lexicographically least prefixed point

To define big-step semantics of a functional language (even one with McCarthy's amb):

- first define convergence ($\Downarrow$) as a least prefixed point
- then define divergence ($\Uparrow$) as a greatest postfixed point.

Another lexicographically least prefixed point

To define big-step semantics of a functional language (even one with McCarthy's amb):

- first define convergence ($\Downarrow$) as a least prefixed point
- then define divergence ($\Uparrow$) as a greatest postfixed point.

Big-step semantics can be seen as describing the denotational semantics of an interpreter, which is a first-order recursive program.

Another lexicographically least prefixed point

To define big-step semantics of a functional language (even one with McCarthy's amb):

- first define convergence ($\Downarrow$) as a least prefixed point
- then define divergence ($\Uparrow$) as a greatest postfixed point.

Big-step semantics can be seen as describing the denotational semantics of an interpreter, which is a first-order recursive program.

The pair $(\Downarrow, \Uparrow)$ is a lexicographically least prefixed point.