Games for countable nondeterminism

GaLoP 2017, Uppsala

John Gowers

April 23, 2017

Related Work I

R. Harmer and G. McCusker.

A fully abstract game semantics for finite nondeterminism.

In *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pages 422–430, 1999.

Russell S. Harmer.

Games and full abstraction for nondeterministic languages.

Technical report, 1999.

J. Laird.

Sequential algorithms for unbounded nondeterminism.

Electronic Notes in Theoretical Computer Science, 319:271 – 287, 2015.

Related Work II

Paul Blain Levy.

Infinite trace equivalence.

Annals of Pure and Applied Logic, 151(2):170 – 198, 2008.

A. W. ROSCOE.

Unbounded non-determinism in CSP.

Journal of Logic and Computation, 3(2):131, 1993.

Takeshi Tsukada and C. H. Luke Ong.

Nondeterminism in game semantics via sheaves.

In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, LICS '15, pages 220–231, Washington, DC, USA, 2015. IEEE Computer Society.

Nondeterministic Strategies

Nondeterministic Strategies

If A is a game with a set P_A of positions, a strategy σ for A is a non-empty prefix-closed subset of P_A^P satisfying:

Determinism If $sa, sb \in \sigma$, then $a = b$

Totality If $s \in \sigma$ and $sa \in P_A$, then $sab \in \sigma$ for some b

Nondeterministic Strategies

If A is a game with a set P_A of positions, a strategy σ for A is a non-empty prefix-closed subset of P_A^P satisfying:

Determinism If $sa, sb \in \sigma$, then $a = b$

Totality If $s \in \sigma$ and $sa \in P_A$, then $sab \in \sigma$ for some b

Relaxing *totality* gives us *partial strategies*.

Nondeterministic Strategies

If A is a game with a set P_A of positions, a strategy σ for A is a non-empty prefix-closed subset of P_A^P satisfying:

Determinism If $sa, sb \in \sigma$, then $a = b$

Totality If $s \in \sigma$ and $sa \in P_A$, then $sab \in \sigma$ for some b

Relaxing *totality* gives us *partial strategies*.

Relaxing *determinism* gives us *nondeterministic strategies*.

Nondeterministic Strategies

If A is a game with a set P_A of positions, a strategy σ for A is a non-empty prefix-closed subset of P_A^P satisfying:

Determinism If $sa, sb \in \sigma$, then $a = b$

Totality If $s \in \sigma$ and $sa \in P_A$, then $sab \in \sigma$ for some b

Relaxing *totality* gives us *partial strategies*.

Relaxing *determinism* gives us *nondeterministic strategies*.

Problem: how can we model divergence in nondeterministic strategies?

The Harmer-McCusker model

The Harmer-McCusker model

Harmer and McCusker model nondeterminism by equipping each (nondeterministic) strategy with an explicit set of divergent positions.

The Harmer-McCusker model

Harmer and McCusker model nondeterminism by equipping each (nondeterministic) strategy with an explicit set of divergent positions.

Divergence in a composition of strategies may arise either from a divergence in one of the strategies we are composing, or through *livelock* (infinite chattering).

The Harmer-McCusker model

Harmer and McCusker model nondeterminism by equipping each (nondeterministic) strategy with an explicit set of divergent positions.

Divergence in a composition of strategies may arise either from a divergence in one of the strategies we are composing, or through *livelock* (infinite chattering).

This method only works for finite nondeterminism: if we naively relax the finite branching condition, then composition is not associative.

Failure of associativity in the presence of infinite branching

$$
\begin{array}{cccccc}
\mathbf{I} & \boldsymbol{\Sigma}^* & \mathbb{N} & \otimes & \boldsymbol{\Sigma}^* & \boldsymbol{\Sigma} \\
 & & & & & q \\
 & & q & & & \\
 & & 2 & & & \\
 & & & & q & \\
 & q & & & & \\
 & a & & & & \\
 & & & & a & \\
 & & & & q & \\
 & q & & & & \\
 & a & & & & \\
 & & & & a & \\
 & & & & & a
\end{array}
$$

Solution: keep track of infinite sequences of moves

A game A is now given by a tuple $(M_A, \lambda_A, \zeta_A, P_A)$ where

- M_A is a set of moves
- $\lambda_A \colon M_A \to \{O, P\}$ identifies each move as an O-move or a P-move
- $P_A \subseteq \overline{M_A}^*$ is a set of legal positions
- $\zeta_A \colon P_A \to \{O, P\}$ designates each position as an O-position or a P-position

…subject to a few rules

Advantages

Advantages

We can distinguish between computations that diverge and computations that converge, but may take an arbitrarily large number of steps to do so.

Advantages

We can distinguish between computations that diverge and computations that converge, but may take an arbitrarily large number of steps to do so.

We do not have to check for infinite chattering explicitly when defining the divergences of a composition of strategies.

Advantages

We can distinguish between computations that diverge and computations that converge, but may take an arbitrarily large number of steps to do so.

We do not have to check for infinite chattering explicitly when defining the divergences of a composition of strategies.

Composition is associative.

Fair PCF

$$\frac{}{(\lambda x.M)N \longrightarrow M[N/x]} \qquad \frac{}{\mathbf{Y}_T M \longrightarrow M(\mathbf{Y}_T M)}$$

$$\frac{}{\mathtt{If0}\ 0 \longrightarrow \lambda x.\lambda y.x} \qquad \frac{}{\mathtt{If0}\ (\mathtt{suc}\ \mathtt{n}) \longrightarrow \lambda x.\lambda y.(y\mathtt{n})}$$

$$\frac{M \longrightarrow M'}{\mathtt{suc}\ M \longrightarrow \mathtt{suc}\ M'} \qquad \frac{M \longrightarrow M'}{\mathtt{If0}\ M \longrightarrow \mathtt{If0}\ M'}$$

$$\frac{M \longrightarrow M'}{MN \longrightarrow M'N} \qquad \frac{}{? \longrightarrow \mathtt{n}}\ n \in \mathbb{N}$$

Must-testing for fair PCF

If M is a term of Fair PCF of ground type `nat`, write $M \Downarrow$ if M has no infinite evaluation paths $M \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \cdots$.

Must-testing for fair PCF

If M is a term of Fair PCF of ground type $\mathtt{nat}$, write $M \Downarrow$ if M has no infinite evaluation paths $M \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \cdots$.

Examples:

- $\Omega_{\mathtt{nat}} = \mathbf{Y}_{\mathtt{nat}}(\lambda x.x) \not\Downarrow$

Must-testing for fair PCF

If M is a term of Fair PCF of ground type $\mathtt{nat}$, write $M \Downarrow$ if M has no infinite evaluation paths $M \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \cdots$.

Examples:

- $\Omega_{\mathtt{nat}} = \mathbf{Y}_{\mathtt{nat}}(\lambda x.x) \not\Downarrow$
- $\mathtt{If0}\ ?\ \Omega_{\mathtt{nat}}\ 0 \not\Downarrow$

Must-testing for fair PCF

If M is a term of Fair PCF of ground type `nat`, write $M \Downarrow$ if M has no infinite evaluation paths $M \longrightarrow M_1 \longrightarrow M_2 \longrightarrow \cdots$.

Examples:

- $\Omega_{\mathtt{nat}} = \mathbf{Y}_{\mathtt{nat}}(\lambda x.x) \not\Downarrow$
- $\mathtt{If0}\ ?\ \Omega_{\mathtt{nat}}\ 0 \not\Downarrow$
- $\mathbf{Y}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)? \Downarrow$

Denotational Semantics for $\mathbf{Y}_T$

Denotational Semantics for $\mathbf{Y}_T$

Harmer and McCusker define an order $\leq_s$ on strategies for a game A:

$\sigma \leq_s \tau$ if

1. $\sigma \subseteq \tau$
2. Every divergent position in τ is divergent in σ
3. Every position in $\tau \setminus \sigma$ is divergent in σ.

Denotational Semantics for $\mathbf{Y}_T$

Harmer and McCusker define an order $\leq_s$ on strategies for a game A:

$\sigma \leq_s \tau$ if

1. $\sigma \subseteq \tau$
2. Every divergent position in τ is divergent in σ
3. Every position in $\tau \setminus \sigma$ is divergent in σ.

The order $\leq_s$ is preserved by composition.

Denotational Semantics for $\mathbf{Y}_T$

Harmer and McCusker define an order $\leq_s$ on strategies for a game A:

$\sigma \leq_s \tau$ if

1. $\sigma \subseteq \tau$
2. Every divergent position in τ is divergent in σ
3. Every position in $\tau \setminus \sigma$ is divergent in σ.

The order $\leq_s$ is preserved by composition.

If F is a set of strategies that is directed with respect to $\leq_s$ then F has a least upper bound given by

$$\left(\bigcup_{\sigma \in F} \sigma \, , \, \bigcap_{\sigma \in F} D_\sigma \right)$$

Denotational semantics for $\mathbf{Y}_T$

This means that we can define a strategy $\mathbf{Y}_A$ for $(A \Longrightarrow A) \Longrightarrow A$ to be the least fixed point of the strategy corresponding to the λ-term

$$\lambda F.\lambda f.f(Ff)$$

Denotational semantics for $\mathbf{Y}_T$

This means that we can define a strategy $\mathbf{Y}_A$ for $(A \Longrightarrow A) \Longrightarrow A$ to be the least fixed point of the strategy corresponding to the λ-term

$$\lambda F.\lambda f.f(Ff)$$

However, $\leq_s$ -limits are not preserved by composition (in either direction), so it is hard to prove computational adequacy for this denotation of $\mathbf{Y}_T$.

Introduction to Ordinal-Indexed Recursion

Introduction to Ordinal-Indexed Recursion

In languages with no nondeterminism or finite nondeterminism, we can study $\mathbf{Y}_T$ by studying its finite approximants $\mathbf{Y}_T^n$, where:

$$\mathbf{Y}_T^n M = \underbrace{M \cdots M}_{n} \Omega_T$$

Introduction to Ordinal-Indexed Recursion

In languages with no nondeterminism or finite nondeterminism, we can study $\mathbf{Y}_T$ by studying its finite approximants $\mathbf{Y}_T^n$, where:

$$\mathbf{Y}_T^n M = \underbrace{M \cdots M}_{n} \Omega_T$$

If $Y_T M M_1 \ldots M_n \Downarrow$, then $\mathbf{Y}_T^n M M_1 \ldots M_n \Downarrow$ for some n – since branching is finite, any well-founded evaluation tree is *bounded*.

Introduction to Ordinal-Indexed Recursion

In languages with no nondeterminism or finite nondeterminism, we can study $\mathbf{Y}_T$ by studying its finite approximants $\mathbf{Y}_T^n$, where:

$$\mathbf{Y}_T^n M = \underbrace{M \cdots M}_{n} \Omega_T$$

If $Y_T M M_1 \ldots M_n \Downarrow$, then $\mathbf{Y}_T^n M M_1 \ldots M_n \Downarrow$ for some n – since branching is finite, any well-founded evaluation tree is *bounded*.

However, in the presence of infinite branching, we may have well-founded evaluation trees that are not bounded.

Introduction to Ordinal-Indexed Recursion

In languages with no nondeterminism or finite nondeterminism, we can study $\mathbf{Y}_T$ by studying its finite approximants $\mathbf{Y}_T^n$, where:

$$\mathbf{Y}_T^n M = \underbrace{M \cdots M}_{n} \Omega_T$$

If $Y_T M M_1 \ldots M_n \Downarrow$, then $\mathbf{Y}_T^n M M_1 \ldots M_n \Downarrow$ for some n – since branching is finite, any well-founded evaluation tree is *bounded*.

However, in the presence of infinite branching, we may have well-founded evaluation trees that are not bounded.

For example, $\mathbf{Y}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)? \Downarrow$, but $\mathbf{Y}^n_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)? \not\Downarrow$ for all n.

Introduction to Ordinal-Indexed Recursion

We replace $\mathbf{Y}_T$ with a new family of constants $\mathbf{Y}_T^\alpha$, where α ranges over countable ordinals and a formal divergence symbol Ω_T. $\mathbf{Y}_T^\alpha$ has operational semantics given by:

$$\frac{}{\mathbf{Y}_T^\alpha M \longrightarrow M(\mathbf{Y}_T^\beta M)}\ \beta < \alpha \qquad\qquad \frac{}{\mathbf{Y}_T^0 M \longrightarrow \Omega_T}$$

Introduction to Ordinal-Indexed Recursion

We replace $\mathbf{Y}_T$ with a new family of constants $\mathbf{Y}_T^\alpha$, where α ranges over countable ordinals and a formal divergence symbol Ω_T. $\mathbf{Y}_T^\alpha$ has operational semantics given by:

$$\frac{}{\mathbf{Y}_T^\alpha M \longrightarrow M(\mathbf{Y}_T^\beta M)}\ \beta < \alpha \qquad \frac{}{\mathbf{Y}_T^0 M \longrightarrow \Omega_T}$$

If $M \Downarrow$, then the evaluation tree for M is well-founded and hence has complexity given by some countable ordinal α. If we replace $\mathbf{Y}$ with $\mathbf{Y}^\alpha$ in the description of M, then we can recover the original evaluation tree.

Introduction to Ordinal-Indexed Recursion

We replace $\mathbf{Y}_T$ with a new family of constants $\mathbf{Y}_T^\alpha$, where α ranges over countable ordinals and a formal divergence symbol Ω_T. $\mathbf{Y}_T^\alpha$ has operational semantics given by:

$$\frac{}{\mathbf{Y}_T^\alpha M \longrightarrow M(\mathbf{Y}_T^\beta M)}\ \beta < \alpha \qquad\qquad \frac{}{\mathbf{Y}_T^0 M \longrightarrow \Omega_T}$$

If $M \Downarrow$, then the evaluation tree for M is well-founded and hence has complexity given by some countable ordinal α. If we replace $\mathbf{Y}$ with $\mathbf{Y}^\alpha$ in the description of M, then we can recover the original evaluation tree.

However, the new term will have additional branches, some of which will diverge.

Introduction to Ordinal-Indexed Recursion

We replace $\mathbf{Y}_T$ with a new family of constants $\mathbf{Y}_T^\alpha$, where α ranges over countable ordinals and a formal divergence symbol Ω_T. $\mathbf{Y}_T^\alpha$ has operational semantics given by:

$$\frac{}{\mathbf{Y}_T^\alpha M \longrightarrow M(\mathbf{Y}_T^\beta M)}\ \beta < \alpha \qquad\qquad \frac{}{\mathbf{Y}_T^0 M \longrightarrow \Omega_T}$$

If $M \Downarrow$, then the evaluation tree for M is well-founded and hence has complexity given by some countable ordinal α. If we replace $\mathbf{Y}$ with $\mathbf{Y}^\alpha$ in the description of M, then we can recover the original evaluation tree.

However, the new term will have additional branches, some of which will diverge.

We want the choice of ordinal β to be 'friendly', so that it does not affect the behaviour of $\Downarrow$.

Big-Step semantics for $\Downarrow$ in Fair PCF

We can get round this problem using big step semantics.

Big-Step semantics for $\Downarrow$ in Fair PCF

We can get round this problem using big step semantics.

We add the following rule for ? to the usual big-step semantics for $\Downarrow$ in PCF:

$$\frac{\forall n \in \omega \;.\; Mn \Downarrow}{M? \Downarrow}$$

Big-Step semantics for $\Downarrow$ in Fair PCF

We can get round this problem using big step semantics.

We add the following rule for ? to the usual big-step semantics for $\Downarrow$ in PCF:

$$\frac{\forall n \in \omega \, . \, Mn \Downarrow}{M? \Downarrow}$$

We then give big-step semantics for the constants $\mathbf{Y}^{\alpha}_{T}$:

$$\frac{\exists \beta < \alpha \, . \, M(\mathbf{Y}^{\beta}_{T} M) M_1 \cdots M_n \Downarrow}{\mathbf{Y}^{\alpha}_{T} M M_1 \cdots M_n \Downarrow}$$

An example

$$\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?$$

An example

$$\begin{aligned} & \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\ \longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \end{aligned}$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\texttt{nat}\to\texttt{nat}}(\lambda f.\lambda x.\texttt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\texttt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\texttt{nat}\to\texttt{nat}}(\lambda f.\lambda x.\texttt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \texttt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\texttt{nat}\to\texttt{nat}}(\lambda f.\lambda x.\texttt{If0}\ x\ 0\ f))
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$\mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ m\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))
\end{aligned}
$$

An example

$$\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}$$

$$\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ m\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ m
\end{aligned}$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ?\\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ?\\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathtt{If0}\ m\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\\
\longrightarrow\ & \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ m\\
\longrightarrow\ & (\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{m+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ m
\end{aligned}
$$

An example

$$
\begin{aligned}
& \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ (n+1)\ 0\ (\mathbf{Y}^{n}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \cdots \longrightarrow \Omega_{\mathtt{nat}}
\end{aligned}
$$

$$
\begin{aligned}
& \mathbf{Y}^{\omega+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ ? \\
\longrightarrow\ & ((\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f))\ ? \\
\longrightarrow\ & \mathtt{If0}\ ?\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathtt{If0}\ m\ 0\ (\mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)) \\
\longrightarrow\ & \mathbf{Y}^{\omega}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ m \\
\longrightarrow\ & (\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\mathbf{Y}^{m+1}_{\mathtt{nat}\to\mathtt{nat}}(\lambda f.\lambda x.\mathtt{If0}\ x\ 0\ f)\ m \\
\longrightarrow\ & \cdots \longrightarrow 0
\end{aligned}
$$

Can we model $\mathbf{Y}_T^\alpha$ within the language?

Can we model $\mathbf{Y}_T^\alpha$ within the language?

Recall that we could model $\mathbf{Y}_T^n M$ within PCF as

$$\underbrace{M \cdots M}_{n} \Omega_T$$

Can we model $\mathbf{Y}_T^\alpha$ within the language?

Recall that we could model $\mathbf{Y}_T^n M$ within PCF as

$$\underbrace{M \cdots M}_{n} \Omega_T$$

We can do this uniformly if we add a recursion constant $\mathtt{iter}_T : \mathtt{nat} \to T \to (T \to T) \to T$ to the language. Then we can model

$$\mathbf{Y}_T^n = \lambda F.\mathtt{iter}_T\, n\, \Omega_T\ F$$

Can we model $\mathbf{Y}_T^{\alpha}$ within the language?

Recall that we could model $\mathbf{Y}_T^{n} M$ within PCF as

$$\underbrace{M \cdots M}_{n} \Omega_T$$

We can do this uniformly if we add a recursion constant $\mathtt{iter}_T : \mathtt{nat} \to T \to (T \to T) \to T$ to the language. Then we can model

$$\mathbf{Y}_T^{n} = \lambda F.\mathtt{iter}_T\, n\, \Omega_T\ F$$

But how would we model $\mathbf{Y}_T^{\omega}$? We need to add friendly choice to the language....

Explicit friendly choice

Explicit friendly choice

We add a new constant to our language:

$$¿ : \texttt{nat}$$

Explicit friendly choice

We add a new constant to our language:

$$¿ : \texttt{nat}$$

We give big-step semantics for $\Downarrow$ corresponding to $¿$:

$$\frac{\exists n \in \omega \ . \ Mn \Downarrow}{M¿ \Downarrow}$$

Then we may model $\mathbf{Y}^{\omega}_{T}$ as:

$$\mathbf{Y}^{\omega}_{T} = \lambda F.\texttt{iter}_T ¿ \ \Omega_T \ F$$

Explicit friendly choice

We add a new constant to our language:

$$¿ : \texttt{nat}$$

We give big-step semantics for $\Downarrow$ corresponding to ¿:

$$\frac{\exists n \in \omega \ . \ Mn \Downarrow}{M¿ \Downarrow}$$

Then we may model $\mathbf{Y}_T^{\omega}$ as:

$$\mathbf{Y}_T^{\omega} = \lambda F.\texttt{iter}_T ¿ \ \Omega_T \ F$$

Going further, we may model $\mathbf{Y}_T^{\omega+1}$ as $\lambda F.F(\texttt{iter}_T ¿ \ \Omega_T \ F)$, and so on.

Explicit friendly choice

We add a new constant to our language:

$$¿ : \mathtt{nat}$$

We give big-step semantics for $\Downarrow$ corresponding to ¿:

$$\frac{\exists n \in \omega \ . \ Mn \Downarrow}{M¿ \Downarrow}$$

Then we may model $\mathbf{Y}^{\omega}_{T}$ as:

$$\mathbf{Y}^{\omega}_{T} = \lambda F.\mathtt{iter}_{T} ¿ \ \Omega_{T} \ F$$

Going further, we may model $\mathbf{Y}^{\omega+1}_{T}$ as $\lambda F.F(\mathtt{iter}_{T} ¿ \ \Omega_{T} \ F)$, and so on.

How far can we go?

Defining recursion principles using friendly choice

Defining recursion principles using friendly choice

Suppose that we can define an ordinal α in our language as a Church encoding:

$$\alpha_T \colon T \to (T \to T) \to ((\mathtt{nat} \to T) \to T) \to T$$

Defining recursion principles using friendly choice

Suppose that we can define an ordinal α in our language as a Church encoding:

$$\alpha_T \colon T \to (T \to T) \to ((\texttt{nat} \to T) \to T) \to T$$

Then we can simulate $\mathbf{Y}^{\alpha}_{T}$ within our language:

$$\mathbf{Y}^{\alpha}_{T} \equiv \lambda F.\alpha_T\ \Omega_T\ F\ (\lambda f.f\text{¿})$$

Defining recursion principles using friendly choice

Suppose that we can define an ordinal α in our language as a Church encoding:

$$\alpha_T \colon T \to (T \to T) \to ((\mathtt{nat} \to T) \to T) \to T$$

Then we can simulate $\mathbf{Y}_T^\alpha$ within our language:

$$\mathbf{Y}_T^\alpha \equiv \lambda F.\alpha_T\ \Omega_T\ F\ (\lambda f.f\ ¿)$$

Using $\mathtt{iter}_T$, it is possible to define Church encodings α_T for all ordinals $\alpha < \epsilon_0$, so we can simulate $\mathbf{Y}_T^\alpha$ for all $\alpha < \epsilon_0$ in PFC - $\mathbf{Y}$ + $\mathtt{iter}$ + ¿.

Friendly vs. unfriendly choice: a 2-player game within the language

Friendly vs. unfriendly choice: a 2-player game within the language

Suppose we have a constant

$$\texttt{If<} : \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat}$$

in our language with the obvious meaning.

Friendly vs. unfriendly choice: a 2-player game within the language

Suppose we have a constant

$$\texttt{If<} : \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat}$$

in our language with the obvious meaning.

Consider the terms

$$\texttt{If<}\ ?\ ¿\ 0\ \Omega_{\texttt{nat}}$$
$$\texttt{If<}\ ¿\ ?\ \Omega_{\texttt{nat}}\ 0$$

Friendly vs. unfriendly choice: a 2-player game within the language

Suppose we have a constant

$$\mathtt{If}< \; : \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat}$$

in our language with the obvious meaning.

Consider the terms

$$\mathtt{If}< \; ? \; ¿ \; 0 \; \Omega_{\mathtt{nat}}$$
$$\mathtt{If}< \; ¿ \; ? \; \Omega_{\mathtt{nat}} \; 0$$

Because of evaluation order, the first term must converge, but the second may diverge.

Friendly vs. unfriendly choice: a 2-player game within the language

Suppose we have a constant

$$\texttt{If}< : \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat} \to \texttt{nat}$$

in our language with the obvious meaning.

Consider the terms

$$\texttt{If}< \; ? \; ¿ \; 0 \; \Omega_{\texttt{nat}}$$
$$\texttt{If}< \; ¿ \; ? \; \Omega_{\texttt{nat}} \; 0$$

Because of evaluation order, the first term must converge, but the second may diverge.

With more complicated predicates, we can encode more complicated (bounded) Gale-Stewart games in the language.

Game Semantics for Friendly Choice

Game Semantics for Friendly Choice

We need the ability to make all choices at once, but forget about the bad ones (unless all our choices were bad).

Game Semantics for Friendly Choice

We need the ability to make all choices at once, but forget about the bad ones (unless all our choices were bad).

If A is a game, then a *flexible strategy* for A is a pair $\Sigma = (\sigma_\Sigma, F_\Sigma)$, where σ_Σ is a strategy for A and $F_\Sigma \subseteq \sigma_\Sigma$ is a set of positions in σ_Σ that are forced. All other positions in σ are optional.

Game Semantics for Friendly Choice

We need the ability to make all choices at once, but forget about the bad ones (unless all our choices were bad).

If A is a game, then a *flexible strategy* for A is a pair $\Sigma = (\sigma_\Sigma, F_\Sigma)$, where σ_Σ is a strategy for A and $F_\Sigma \subseteq \sigma_\Sigma$ is a set of positions in σ_Σ that are forced. All other positions in σ are optional.

Then a *switching* for Σ is a strategy σ' for A such that $\sigma' \subseteq \sigma_\Sigma$ and such that if $s \in \sigma'$ and $sab \in F_\Sigma$ then $sab \in \sigma'$. In this case, we write $\sigma' \lhd \Sigma$.

Game Semantics for Friendly Choice

We need the ability to make all choices at once, but forget about the bad ones (unless all our choices were bad).

If A is a game, then a *flexible strategy* for A is a pair $\Sigma = (\sigma_\Sigma, F_\Sigma)$, where σ_Σ is a strategy for A and $F_\Sigma \subseteq \sigma_\Sigma$ is a set of positions in σ_Σ that are forced. All other positions in σ are optional.

Then a *switching* for Σ is a strategy σ' for A such that $\sigma' \subseteq \sigma_\Sigma$ and such that if $s \in \sigma'$ and $sab \in F_\Sigma$ then $sab \in \sigma'$. In this case, we write $\sigma' \lhd \Sigma$.

Given flexible strategies $\Sigma \colon A \to B$ and $T : B \to C$, we may form the composite:

$$T \circ \Sigma = \bigsqcup_{\substack{\sigma' \lhd \Sigma \\ \tau' \lhd T}} \tau' \circ \sigma'$$

Game semantics for friendly choice

Game semantics for friendly choice

We embed the original game semantics into the flexible game semantics by sending $\sigma \mapsto |\sigma| := (\sigma, \sigma)$.

Game semantics for friendly choice

We embed the original game semantics into the flexible game semantics by sending $\sigma \mapsto |\sigma| := (\sigma, \sigma)$.

So ? inherits its original semantics as $(?, ?)$.

Game semantics for friendly choice

We embed the original game semantics into the flexible game semantics by sending $\sigma \mapsto |\sigma| := (\sigma, \sigma)$.

So ? inherits its original semantics as $(?, ?)$.

We give ¿ an explicitly flexible semantics:

$$[\![¿]\!] = (?, \{\epsilon\})$$

Game semantics for friendly choice

We embed the original game semantics into the flexible game semantics by sending $\sigma \mapsto |\sigma| := (\sigma, \sigma)$.

So ? inherits its original semantics as $(?, ?)$.

We give ¿ an explicitly flexible semantics:

$$[\![¿]\!] = (?, \{\epsilon\})$$

So all the choices of number in ¿ are optional.

Game semantics for friendly choice

We embed the original game semantics into the flexible game semantics by sending $\sigma \mapsto |\sigma| := (\sigma, \sigma)$.

So ? inherits its original semantics as $(?, ?)$.

We give ¿ an explicitly flexible semantics:

$$[\![¿]\!] = (?, \{\epsilon\})$$

So all the choices of number in ¿ are optional. We get a consistency result:

Proposition

If M is a term of PCF - $\mathbf{Y}$ + ? + ¿ of type `nat` *such that $M \Downarrow$, then there are no divergent positions in $[\![M]\!]$.*

Stable ordering for flexible strategies

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

We say that $\Sigma \leq_s T$ if $\sigma_\Sigma \leq_s \sigma_T$ and $\sigma_\Sigma \cap F_T \subseteq F_\Sigma$.

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

We say that $\Sigma \leq_s T$ if $\sigma_\Sigma \leq_s \sigma_T$ and $\sigma_\Sigma \cap F_T \subseteq F_\Sigma$.

Then we have least upper bounds for stably-directed sets X of flexible strategies:

$$\sigma_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigsqcup_{\Sigma \in X} \sigma_\Sigma$$

$$F_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigcap_{\Sigma \in X} F_\Sigma$$

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

We say that $\Sigma \leq_s T$ if $\sigma_\Sigma \leq_s \sigma_T$ and $\sigma_\Sigma \cap F_T \subseteq F_\Sigma$.

Then we have least upper bounds for stably-directed sets X of flexible strategies:

$$\sigma_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigsqcup_{\Sigma \in X} \sigma_\Sigma$$
$$F_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigcap_{\Sigma \in X} F_\Sigma$$

$\bigsqcup_{\Sigma \in F} \Sigma$ is a ‘friendly choice between the $\Sigma \in F$’.

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

We say that $\Sigma \leq_s T$ if $\sigma_\Sigma \leq_s \sigma_T$ and $\sigma_\Sigma \cap F_T \subseteq F_\Sigma$.

Then we have least upper bounds for stably-directed sets X of flexible strategies:

$$\sigma_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigsqcup_{\Sigma \in X} \sigma_\Sigma$$

$$F_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigcap_{\Sigma \in X} F_\Sigma$$

$\bigsqcup_{\Sigma \in F} \Sigma$ is a 'friendly choice between the $\Sigma \in F$'.

We can then construct the $\mathbf{Y}^\alpha_T$ as the elements of the Bourbaki-Witt chain.

Stable ordering for flexible strategies

We can extend the stable order $\leq_s$ to flexible strategies.

We say that $\Sigma \leq_s T$ if $\sigma_\Sigma \leq_s \sigma_T$ and $\sigma_\Sigma \cap F_T \subseteq F_\Sigma$.

Then we have least upper bounds for stably-directed sets X of flexible strategies:

$$\sigma_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigsqcup_{\Sigma \in X} \sigma_\Sigma$$

$$F_{\bigsqcup_{\Sigma \in X} \Sigma} = \bigcap_{\Sigma \in X} F_\Sigma$$

$\bigsqcup_{\Sigma \in F} \Sigma$ is a 'friendly choice between the $\Sigma \in F$'.

We can then construct the $\mathbf{Y}_T^\alpha$ as the elements of the Bourbaki-Witt chain.

For suitably large α (conjecturally $\alpha = \epsilon_0$), we get a sound and adequate semantics for $\mathbf{Y}_T$.