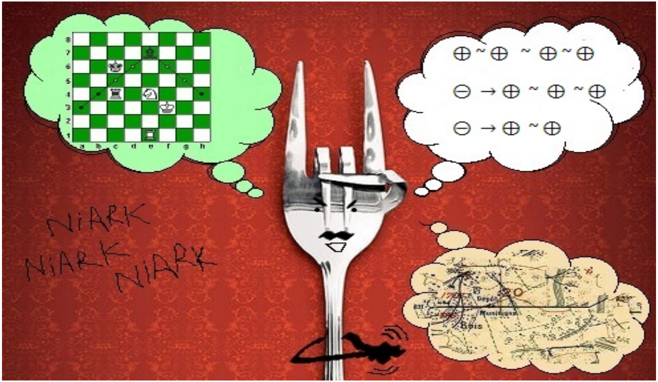
La stratégie de la fourchette



Simon Castellan, 8 janvier 2015

JFLA 2015

fork(II)

11/3/71 SYS FORK (II)

NAME fork -- spawn new process

SYNOPSIS
sys fork / fork = 2.
(new process return)
(old process return)

DESCRIPTION *fork* is the only way new processes are created. The new process's core image is a copy of that of the caller of *fork* the only distinction is the return location and the fact that r0 in the old process contains the process ID of the new process. This process ID is used by *wait*.

FILES

SEE ALSO sys wait, sys exec

DIAGNOSTICS The error bit (c–bit) is set in the old process if a new process could not be created because of lack of swap space.

BUGS See *wait* for a subtle bug in process destruction.

OWNER ken, dmr

Demo

...demo...

Syntax

- Idealized Algol + a monoid structure on each base type.

$$
\begin{array}{rll}
A, B ::= & \mathtt{nat} \mid \mathtt{bool} \mid \mathtt{unit} & \text{(base types)} \\
\mid & \mathtt{var} & \text{(references)} \\
\mid & A \Rightarrow B & \text{(arrow types)} \\
t, u ::= & x \mid \lambda x.\ t \mid t\ u \mid Y & \text{(simply typed } \lambda\text{-calculus + fixpoint)} \\
\mid & c & \text{(PCF constants)} \\
\mid & \mathtt{new}\ r := k\ \mathtt{in}\ t & \text{(reference creation)} \\
\mid & t := u \mid\ !t & \text{(operations on references)} \\
\mid & t; u \mid \text{if } t \text{ then } u \text{ else } v & \text{(destructors)} \\
\mid & t \parallel u \mid \text{exit} & \text{(forks, only on base types)}
\end{array}
$$

Operational semantics - structural congruence

Idea: pure β-reduction + redution at base type for effects. (Inspired from [AM99, GM07])

- **Structural congruence** $\equiv$ to reduce at higher-order types: smallest congruence containing the two equations

$$(\lambda x.t)\ u \equiv t[u/x] \quad Y\ M \equiv M\ (Y\ M)$$

- **Small-step reduction** at base types: $\Gamma \vdash (t, \rho) \to (t', \rho') : X$
  - Γ contains only references
  - t, t' are terms such that $\Gamma \vdash t, t' : X$ (base type)
  - $\rho, \rho' : \mathrm{dom}(\Gamma) \to \mathbb{N}$

Operational semantics - effect reduction

$$\beta\text{-red}\ \frac{t' \equiv t \qquad \Gamma \vdash t, \rho \to u, \rho' \qquad u \equiv u'}{\Gamma \vdash t', \rho \to u', \rho'}$$

$$\text{Deref}\ \frac{\rho(r) = k}{\Gamma \vdash\ !r, \rho \to k, \rho}$$

$$\text{Assign}\ \frac{}{\Gamma \vdash r := k, \rho \to (), (r \mapsto k) \cup \rho \setminus r}$$

$$\text{New1}\ \frac{r \notin \mathsf{fv}(t)}{\Gamma \vdash \mathtt{new}\ r := k\ \mathtt{in}\ t, \rho \to t, \rho \setminus r}$$

$$\text{New2}\ \frac{\Gamma, r : \mathtt{var} \vdash t, \rho \to t', \rho'}{\Gamma \vdash \mathtt{new}\ r := \rho(r)\ \mathtt{in}\ t, \rho \setminus r \to \mathtt{new}\ r := \rho'(r)\ \mathtt{in}\ t', \rho' \setminus r}$$

Operational semantics - effect reduction

$$E ::= \quad \textbf{(base type evaluation context)}$$
$$[] \mid \mathsf{succ}\ E \mid \ \ldots \ \mid \mathsf{if}\ E\ \mathsf{then}\ t\ \mathsf{else}\ u \mid x := E \mid x; E$$

$$\text{Duplication}\ \frac{\text{E is an evaluation context}}{\Gamma \vdash E[t \parallel u], \rho \to E[t] \parallel E[u], \rho}$$

$$\text{Erasement}\ \frac{\text{E is an evaluation context}}{\Gamma \vdash E[\mathsf{exit}], \rho \to \mathsf{exit}, \rho}$$

$$\text{Preemption}\ \frac{\Gamma \vdash t, \rho \to t', \rho'}{\Gamma \vdash (t \parallel u), \rho \to (t' \parallel u), \rho'}$$

$$\text{Preemption'}\ \frac{\Gamma \vdash u, \rho \to u', \rho'}{\Gamma \vdash (t \parallel u), \rho \to (t \parallel u'), \rho'}$$

Syntax $\to$ Semantics

- Fork calculus = operational modelisation of the `fork` syscall
- Features concurrency and non-determinism

$$\text{Concurrency} + \text{Non-determinism}$$

$$\text{“true and false”} + \text{“true or false”}$$

$$\text{partial orders} + \text{binary conflict}$$

$\to$ Event structures.

Syntax $\to$ Semantics

- Fork calculus = operational modelisation of the `fork` syscall
- Features concurrency and non-determinism

$$\begin{array}{c} \text{Concurrency} + \text{Non-determinism} \\ \text{“true {\color{red}and} false”} + \text{“true {\color{red}or} false”} \\ \text{partial orders} + \text{binary conflict} \end{array}$$

  $\to$ Event structures.
- Now: a very high-level view of the game model based on event structures.

On game semantics

Denotational semantics (at first of typed λ-calculus + extensions) where:

- **types $\rightarrow$ games**
- **programs $\rightarrow$ strategies**
- **computation $\rightarrow$ interaction of strategies**

Hyland-Ond game semantics: supports references [AM99] and concurency through interleavings [GM07].
Our model: truly concurrent extension of a **truly concurrent** semantics to this language based on Winskel's concurrent games.

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

q^-

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$\begin{tikzcd} & q^{-} \arrow[dl] \\ q^{+} & \end{tikzcd}$$

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to {\color{red}\alpha}) \to \alpha) \to \alpha$$

$$\begin{tikzcd} & & q^- \arrow[dl] \\ & q^+ \arrow[dl] & \\ {\color{red} q^-} & & \end{tikzcd}$$

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$\begin{tikzcd}
 & & & q^- \arrow[dl] \\
 & & q^+ \arrow[dl] & \\
 & q^- \arrow[dl] & & \\
q^+ & & &
\end{tikzcd}$$

Usual HO game semantics

Rules for a type? Arenas.

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$\begin{tikzcd}[column sep=small]
 & & & q^- \arrow[dl] \\
 & & q^+ \arrow[dl] & \\
 & q^- \arrow[dl] & & \\
q^+ & & &
\end{tikzcd}$$

Usual HO game semantics

How to inhabit: $\vdash$ $: ((\alpha \to \alpha) \to \alpha) \to \alpha$

$$(((\alpha \to \alpha) \to \alpha) \to \alpha \qquad (((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$q^-$$

$$q^+$$

$$q^-$$

$$q^+$$

Usual HO game semantics

How to inhabit: $\vdash \lambda f.\quad : ((\alpha \to \alpha) \to \alpha) \to \alpha$

$$(((\alpha \to \alpha) \to \alpha) \to \alpha \qquad\qquad (((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$q^- \qquad\qquad q^-_{\lambda f.}$$

$$q^+$$

$$q^-$$

$$q^+$$

Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f \qquad : ((\alpha \to \alpha) \to \alpha) \to \alpha$

$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$q^- \swarrow q^+ \swarrow q^- \swarrow q^+$$

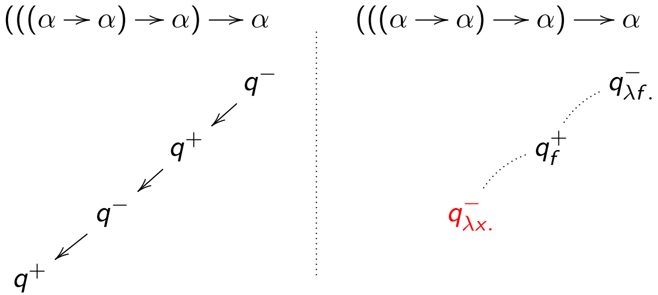
$$(((\alpha \to \alpha) \to \alpha) \to \alpha$$

$$q^-_{\lambda f.}$$

$$q^+_f$$

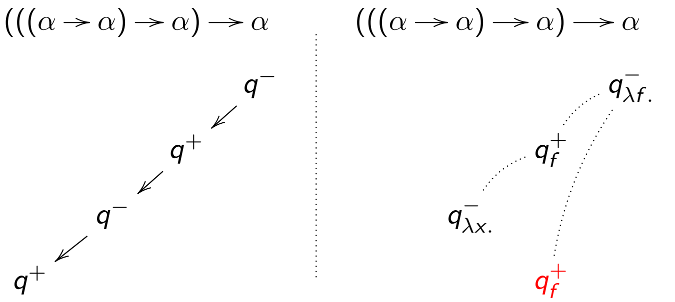
Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f\ (\lambda x.\qquad\qquad)$: $((\alpha \to \alpha) \to \alpha) \to \alpha$



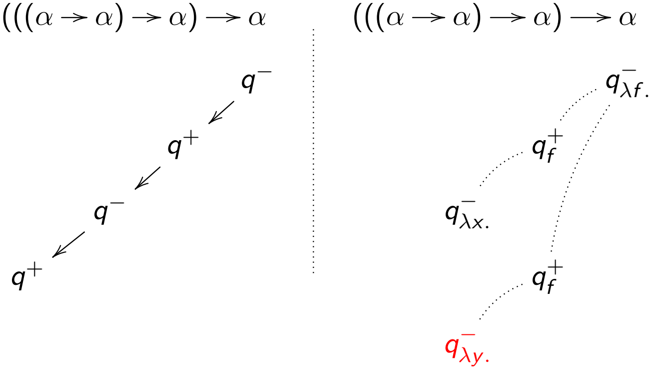
Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f\ (\lambda x.f \qquad) : ((\alpha \to \alpha) \to \alpha) \to \alpha$



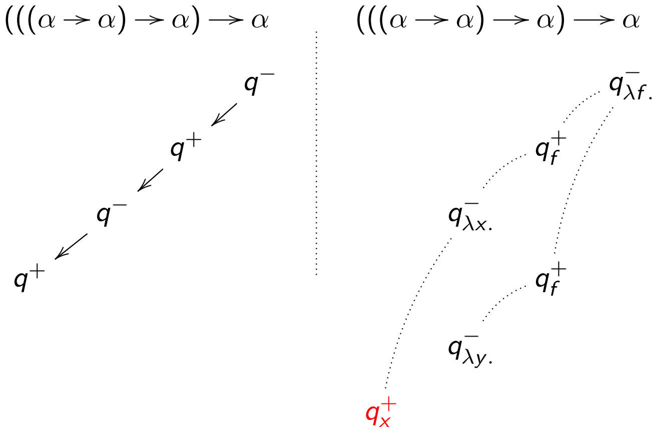
Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f\ (\lambda x.f\ (\lambda y.\ \ ))$: $((\alpha \to \alpha) \to \alpha) \to \alpha$



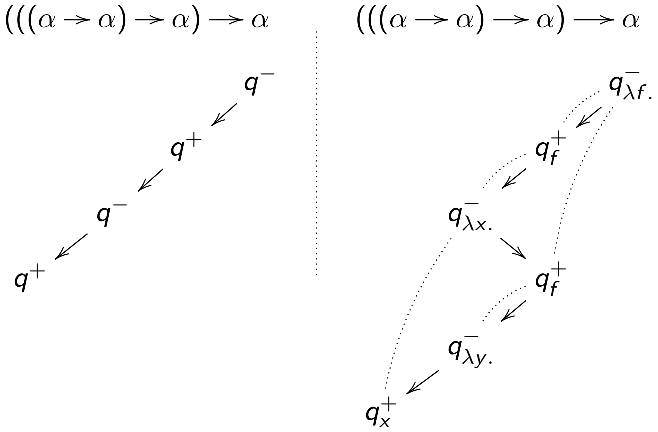
Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f\ (\lambda x.f\ (\lambda y.x)) : ((\alpha \to \alpha) \to \alpha) \to \alpha$



Usual HO game semantics

How to inhabit: $\vdash \lambda f.\ f\ (\lambda x. f\ (\lambda y. x)) : ((\alpha \to \alpha) \to \alpha) \to \alpha$



Moves are chronologically (ie. totally) ordered.

$\to$ To handle concurrency, we need **partial orders** on plays

Concurrency in arenas

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \longrightarrow (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$\begin{tikzcd}
 & & & & & q^- \arrow[dlll] \arrow[dll] \arrow[dl] \\
 & & q^+ \arrow[dll] \arrow[dl] & \mathtt{ff}^+ & \mathtt{tt}^+ & \\
\mathtt{ff}^- & \mathtt{tt}^- & & & &
\end{tikzcd}$$

The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \to (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$q^{-}_{\lambda b.}$$

The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \to (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$\begin{tikzcd} & q^-_{\lambda b.} \arrow[dl] \arrow[dl, dotted, bend right, dash] \\ q^+_b & \end{tikzcd}$$

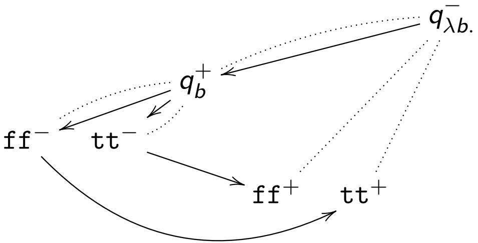
The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \longrightarrow (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$\begin{tikzcd}
 & & & q^-_{\lambda b.} \arrow[dll] \\
 & q^+_b \arrow[dl] \arrow[d] & & \\
\mathtt{ff}^- & \mathtt{tt}^- & &
\end{tikzcd}$$

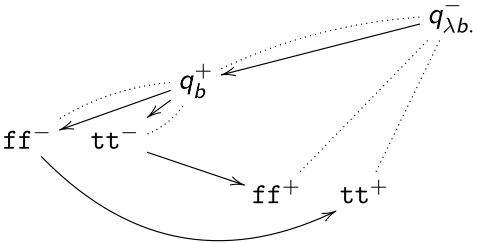
The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \to (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$



The concurrent content of terms

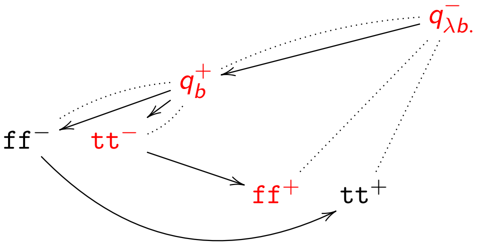
$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \to (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$



$\rightarrow$ evaluation order of the two negative answers is left to the Opponent: this strategy expresses its behaviour against the most concurrent Opponent.

The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \rightarrow (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

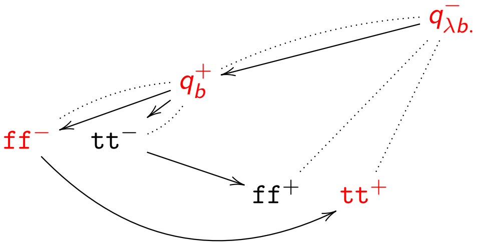


$\rightarrow$ evaluation order of the two negative answers is left to the Opponent: this strategy expresses its behaviour against the most concurrent Opponent.

neg $\mathtt{tt} = \mathtt{ff}$

The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \rightarrow (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

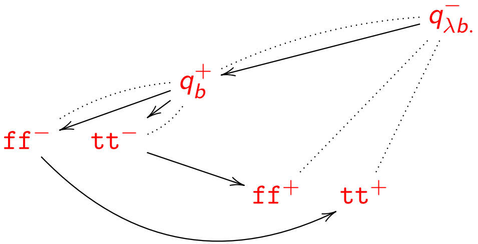


$\rightarrow$ evaluation order of the two negative answers is left to the Opponent: this strategy expresses its behaviour against the most concurrent Opponent.

$$\mathsf{neg}\ \mathtt{tt} = \mathtt{ff} \qquad \mathsf{neg}\ \mathtt{ff} = \mathtt{tt}$$

The concurrent content of terms

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha) \rightarrow (\alpha \longrightarrow \alpha \longrightarrow \alpha)$$



$\rightarrow$ evaluation order of the two negative answers is left to the Opponent: this strategy expresses its behaviour against the most concurrent Opponent.

$$\mathsf{neg}\ \mathtt{tt} = \mathtt{ff} \quad \mathsf{neg}\ \mathtt{ff} = \mathtt{tt} \quad \mathsf{neg}\ (\mathtt{ff} \parallel \mathtt{tt}) = (\mathtt{tt} \parallel \mathtt{ff})$$

Concurrency against non-determinism

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$\begin{tikzcd}
 & & q^- \arrow[dll] \arrow[dl] \\
\mathtt{ff}^+ & \mathtt{tt}^+ &
\end{tikzcd}$$

fork

$$(\alpha \longrightarrow \alpha \longrightarrow \alpha)$$

$$\begin{tikzcd}
 & & q^- \arrow[dll] \arrow[dl] \\
\mathtt{ff}^+ \arrow[r, no head, squiggly] & \mathtt{tt}^+ &
\end{tikzcd}$$

choice

Results & conclusions

Results:

- Denotational semantics for the fork calculus $(t, \rho) \mapsto [\![t; \rho]\!]$
- Soundness results: If $\Gamma \vdash (t, \rho) \to (t', \rho')$ then $[\![t; \rho]\!]$ **simulates** $[\![t'; \rho]\!]$ (in [GM07], the result was full abstraction wrt trace inclusion).
- We retain **a lot** of information on the programs (**causalities**, **non-deterministic branching point**)

Perspectives:

- A showcase of the power of concurrent games: we can actually model complex concurrent programming languages
- Soundness is a first step, aim: full abstract wrt weak bisimulation because strategies retain a lot of information.
- Main problem: diverging branches hidden during composition:

$$[\![\text{if choice then () else } \Omega]\!] \simeq [\![()]\!]$$

Samson Abramsky and Guy McCusker.
Full abstraction for idealized algol with passive expressions.
volume 227, pages 3–42. 1999.

Dan R. Ghica and Andrzej S. Murawski.
Angelic semantics of fine-grained concurrency, 2007.