

A CPS Encoding of Name-Passing in Higher-order Mobile Embedded Resources

Mikkel Bundgaard, Thomas Hildebrandt, and Jens Chr. Godskesen

`{mikkelbu,hilde,jcg}@itu.dk`

Department of Theoretical Computer Science
IT University of Copenhagen

Overview

The following items will be covered in this talk:

- Higher-Order Mobile Embedded Resources
- π -calculus (with explicit substitutions)
- The encoding
- Some examples
- Results and conclusion

Why Homer

- A calculus for **explicit mobile computing resources** at nested locations with a simple syntax and semantics.
 - **Process passing** the only communication form possible.
- Yet powerful:
 - **Recursion** can be encoded by copying processes (as in the **λ -calculus** and **CHOCS**).
 - π -calculus **name-passing** and **binding** can be encoded (this talk).

A Core Higher-order Calculus

The grammar of **processes expressions**:

$$p, q, r ::= \mathbf{0} \quad | \quad p \parallel q \quad | \quad (n)p \quad | \quad \lambda . p \quad | \quad x$$

where λ ranges over the **prefixes**:

$\bar{n}\langle r \rangle$ send a passive resource r to n

$n(x)$ receive a resource at n and bind it to x

A reduction example

$$\bar{n}\langle q \rangle . p_1 \parallel n(x) . p_2 \searrow p_1 \parallel p_2[q/x]$$

Nested Addresses & Active PP

The grammar of **processes expressions** (as before):

$$p, q, r ::= \mathbf{0} \quad | \quad p \parallel q \quad | \quad (n)p \quad | \quad \lambda . p \quad | \quad x$$

where λ ranges over the **prefixes** (and $\delta \in \mathcal{N}^+$):

$\bar{\delta}\langle r \rangle$ send a passive resource r to δ

$\delta(x)$ receive a resource at δ and bind it to x

$\delta\langle r \rangle$ active (computing) resource r at δ

$\bar{\delta}(x)$ take computing resource from δ and bind it to x

Reduction Examples

Computing resources can be **moved**

$$n\langle q \rangle . p_1 \parallel \bar{n}(x) . p_2 \searrow p_1 \parallel p_2[q/x]$$

but can also **compute**

$$q \searrow q' \text{ implies } n\langle q \rangle . p_1 \searrow n\langle q' \rangle . p_1$$

and **communicate** upwards and downwards

$$n_1\langle n_2\langle q \rangle . q' \parallel q'' \rangle . p_1 \parallel \overline{n_1 n_2}(x) . p_2 \searrow \\ n_1\langle q' \parallel q'' \rangle . p_1 \parallel p_2[q/x]$$

The π -calculus

- We consider a **finite** π -calculus (can easily be extended to the full π -calculus).
- As Zimmer [Zim03] we use an intermediate π -calculus with **explicit substitutions**.
- So instead of

$$\frac{}{n(m) . P \mid \bar{n}\langle m' \rangle . Q \rightarrow_{\pi} \{m'/m\}P \mid Q}$$

- we have (assuming a mapping $\sigma : N \rightarrow N$)

$$\frac{}{\sigma \vdash n(m) . P \mid \bar{n}'\langle m' \rangle . Q \rightarrow_{\pi\sigma} \sigma[m \mapsto \sigma m'] \vdash P \mid Q},$$

if $m \notin \text{dom } \sigma \cup \text{codom } \sigma$ and $\sigma n = \sigma n'$

Observations

- There is an **operational correspondence** between the two π -calculi.

If $P = Q\sigma$ then $P \rightarrow_{\pi} P'$ iff $\sigma \vdash Q \rightarrow_{\pi\sigma} \sigma' \vdash Q'$ such that $P' = Q'\sigma'$.

- The set of free names (in the process and in $\text{codom } \sigma$) are **fixed** under reduction because of the explicit substitutions.

Intuition Behind the Encoding

- We encode a **name** n as a process $\llbracket n \rrbracket$ with two methods: to send and to receive on n .
- We represent **dynamic binding** by indirection (we let $n \langle \llbracket m \rrbracket \rangle$ represent that the formal name n has been substituted by the name m).
- We use local, nested, named locations to “**localise**” communication $(a)(a \langle \llbracket m \rrbracket \rangle \parallel P)$.
- We utilise **CPS** to encode the synchronous communication.

Encoding of Names and Terms

We assume name set $\mathcal{N} \uplus \{n' \mid n \in \mathcal{N}\}$ and encode the **process constructs** as follows

$$\begin{aligned} \llbracket (\nu n)P \rrbracket &= (n)(n')(\llbracket P \rrbracket \parallel n'\langle \llbracket n \rrbracket \rangle) \\ \llbracket P \mid Q \rrbracket &= \llbracket P \rrbracket \parallel \llbracket Q \rrbracket \\ \llbracket \mathbf{0} \rrbracket &= \mathbf{0} \end{aligned}$$

Let $\llbracket \sigma \vdash P \rrbracket$ denote the following

$$(\tilde{m})(\llbracket P \rrbracket \parallel \prod_{n \in \text{dom } \sigma} n'\langle \llbracket \sigma n \rrbracket \rangle) ,$$

where $\tilde{m} = \{n' \mid n \in \text{dom } \sigma \text{ and } n \neq \sigma n\}$.

Encoding Names and Binding

The encoding of a **name** n contains two methods: to send and receive on n , respectively.

$$\llbracket n \rrbracket = s \langle Send_n \rangle \parallel r \langle Receive_n \rangle$$

$$Send_n = v(x) . c(y) . \bar{n} \langle x \rangle . y$$

$$Receive_n = v(x) . c(y) . n(z) . (a) \left(a \langle x \rangle \parallel \bar{a}v \langle z \rangle . \bar{a} \langle z' \rangle . (y \parallel z') \right)$$

$$\llbracket n(m) . P \rrbracket = (a) (\bar{n}' \langle x \rangle . (n' \langle x \rangle \parallel (m') (a \langle x \rangle \parallel \bar{a}r \langle Bind_m \rangle . \bar{a}rc \langle \llbracket P \rrbracket \rangle . \bar{a}r \langle x'' \rangle . \bar{a} \langle z \rangle . x'')))$$

where $Bind_m = v(x) . m' \langle x \rangle$.

An Example – Matching Reductions

A reduction in the $\pi\sigma$ -calculus

$$id_A \vdash \bar{n}\langle m \rangle . P \mid n(f) . Q \rightarrow_{\pi\sigma} id_A[f \mapsto m] \vdash P \mid Q$$

in our encoding, letting r denote $\prod_{n \in A} n' \langle \llbracket n \rrbracket \rangle$

$$\begin{aligned} & \llbracket \bar{n}\langle m \rangle . P \rrbracket \parallel \llbracket n(f) . Q \rrbracket \parallel r \searrow^* \\ & \bar{n}\langle \llbracket m \rrbracket \rangle . \llbracket P \rrbracket \parallel (f')(n(z) . (a)(a\langle Bind_f \rangle \parallel \\ & \quad \bar{a}v\langle z \rangle . \bar{a}(z') . (\llbracket Q \rrbracket \parallel z')))) \parallel r \searrow^* \\ & (f')(\llbracket P \rrbracket \parallel \llbracket Q \rrbracket \parallel r \parallel f'\langle \llbracket m \rrbracket \rangle) = \\ & \llbracket id_A[f \mapsto m] \vdash P \mid Q \rrbracket \end{aligned}$$

Main Results

- We define **barbed bisimilarity** \approx between processes of the $\pi\sigma$ -calculus and Homer, if whenever $(\sigma \vdash P, q) \in \approx$,
 - if $\sigma \vdash P \rightarrow_{\pi\sigma} \sigma' \vdash P'$ then there exists q' such that $q \searrow^* q'$ and $\sigma' \vdash P' \approx q'$
 - if $q \searrow q'$ then there exists P' and σ' such that $\sigma \vdash P \xrightarrow{*}_{\pi\sigma} \sigma' \vdash P'$ and $\sigma' \vdash P' \approx q'$
 - if $\sigma \vdash P \Downarrow n$ then $q \Downarrow n$
 - if $q \Downarrow n$ then $\sigma \vdash P \Downarrow n$
- For all π -calculus processes P and substitutions σ , we have $\sigma \vdash P \approx \llbracket \sigma \vdash P \rrbracket$.

Main Results Cont.

- Our encoding is **fully abstract** with respect to **barbed bisimulation**:

$$\sigma \vdash P \approx_{\pi} \sigma \vdash Q \text{ iff } \llbracket \sigma \vdash P \rrbracket \approx_H \llbracket \sigma \vdash Q \rrbracket.$$

- it is **sound** with respect to **barbed congruence**:

$$\forall \mathcal{C}_H. \mathcal{C}_H(\llbracket \sigma \vdash P \rrbracket) \approx_H \mathcal{C}_H(\llbracket \sigma \vdash Q \rrbracket) \text{ implies} \\ \forall \mathcal{C}_{\pi} \forall \sigma'. \sigma \sigma' \vdash \mathcal{C}_{\pi}(P) \approx_{\pi} \sigma \sigma' \vdash \mathcal{C}_{\pi}(Q),$$

such that $\text{dom } \sigma' \supseteq \text{fn}(\mathcal{C}_{\pi}) \setminus \text{dom } \sigma$ and $\text{dom } \sigma' \cap \text{dom } \sigma = \emptyset$.

Conclusion and Future Work

Conclusions:

- Local, named, nested locations are sufficient to encode **name passing** and dynamic binding.
- *Synchronous communication* can be encoded using a CPS scheme.

Future work:

- Next step: encode a version of Homer **extended** with name-passing in pure Homer.
- Explore Homer calculus in the setting of **Bigraphs**.

References

References

- [Zim03] Pascal Zimmer. On the expressiveness of pure mobile ambients. In Luca Aceto and Björn Victor, editors, *Proceedings of the 7th International Workshop on Expressiveness in Concurrency (EXPRESS'00)*, volume 39 of *Electronic Notes in Theoretical Computer Science*, pages 81–104. Elsevier, 2003.