Behavioural Types for Mobile Calculi with Labels

António Ravara

Dep. of Mathematics
Instituto Superior Técnico
Technical University of Lisbon

Joint work with Vasco T. Vasconcelos

Concurrent uniform objects

Static pattern of behaviour:

modelling *objects as records* leads to *interfaces as types*;

types partially specify what services an object provides.

Approach is not informative regarding behaviour;

one cannot:

- pop elements from an empty stack;
- get the balance of an account from an off-line ATM.

Concurrent non-uniform objects

Dynamic pattern of behaviour:

objects may *dynamically change behaviour*, according to their state, and obeying synchronisation constraints.

References:

Actors [Hewitt 73]

OBLOG project [Sernadas *et al.* 91];

Non-uniform service availability [Nierstrazs 95].

Examples:

- One-Place Buffer.

  read-only when full; write-only when empty.

- Automatic Teller Machine.

  first asks pin; then asks option;

  if withdraw asks amount.

Example: uniform one-place buffer

In TyCO (Vasconcelos):

$$
\begin{aligned}
&\textbf{def} \quad \mathsf{Empty}(b) = \\
&\qquad b?\{\, \mathit{write}(u) = \mathsf{Full}[b,u], \\
&\qquad\quad \mathit{read}(r) = \; b!\mathit{read}\langle r\rangle \mid \mathsf{Empty}[b]\,\} \\
&\textbf{and} \quad \mathsf{Full}(b,u) = \\
&\qquad b?\{\, \mathit{write}(v) = b!\mathit{write}\langle v\rangle \mid \mathsf{Full}[b,u], \\
&\qquad\quad \mathit{read}(r) = \; r!\mathit{val}\langle u\rangle \mid \mathsf{Empty}[b]\,\}
\end{aligned}
$$

In π_a^V (Sangiorgi): **def** $\mathsf{Empty}(b) = b?(x)$.
$\mathsf{case}\ x\ \mathsf{of}\ \{\mathit{write}(u) = \mathsf{Full}[b,u], \mathit{read}(r) = b!\mathit{read}\langle r\rangle \mid \mathsf{Empty}[b]\}$

Example: non-uniform buffer

Alternates between read and write:

$$
\begin{aligned}
&\textbf{def} && \mathsf{Empty}(b) = \\
& && \quad b?\{\, \mathit{write}(u) = \mathsf{Full}[b, u]\, \} \\
&\textbf{and} && \mathsf{Full}(b, u) = \\
& && \quad b?\{\, \mathit{read}(r) = \ r!\mathit{val}\langle u\rangle \mid \mathsf{Empty}[b]\, \}
\end{aligned}
$$

Motivation: behaviour-aware

Type of name b:

$\{write : \alpha,\ read : val(\alpha)\}$, for any type α.

Behaviour of name b:

$\mu t.write(\alpha).read(val(\alpha)).t$

Problem: non-uniform version *is not typable!*

$\mathsf{Empty}[b] \mid b!read\langle r\rangle$ is an *error*;

name b must always have the same interface.

Our aim: typing concurrent objects

Notion of communication error:

message-never-understood.

Dynamic types:

behaviours-as-types.

Provably correct type systems:

uniform and non-uniform

unique object ids

no orphan messages

Typed Concurrent Objects

TyCO [V. T. Vasconcelos, 94]

$P ::= \mathbf{0}$	*terminated process*
$\mid\ a?M$	*ephemeral object*
$\mid\ (\textsf{rec}\, X(\vec{x}).a?M)[\vec{v}]$	*persistent object*
$\mid\ a!l\langle\tilde{v}\rangle$	*asynchronous message*
$\mid\ X\langle\tilde{v}\rangle$	*recursive call*
$\mid\ (\nu\, x)\, P$	*variable declaration*
$\mid\ (P \mid Q)$	*parallel composition*
$M ::= \{l_1(\tilde{x}_1) = P_1, \ldots, l_n(\tilde{x}_n) = P_n\}$	*method collection*

Operational Semantics of TyCO

Structural congruence: standard + permutation of methods

(Early) Labeled transition system

$$\mathit{Out} \quad a!l\langle\tilde{v}\rangle \xrightarrow{a!l\langle\tilde{v}\rangle} \mathbf{0}$$

$$\mathit{In} \quad a?\{l_1(\tilde{x}_1) = P_1, \ldots, l_n(\tilde{x}_n) = P_n\} \xrightarrow{a?l_j\langle\tilde{v}\rangle} P_j[\tilde{v}/\tilde{x}]$$

$$\mathit{Rin} \quad (\mathsf{rec}\, X(M).)[a\vec{u}] \xrightarrow{a?l\langle\tilde{v}\rangle}$$
$$(M[\mathsf{rec}\, X(M)./X][a\vec{u}/x\vec{x}]) \bullet l\langle\tilde{v}\rangle$$

Dynamic Notion of Communication Error

Message-never-understood.

$$\mathsf{Empty}[b] \mid b!think\langle a\rangle$$

Blocked objects.

$$(\nu\, b)\,(\mathsf{Empty}[b] \mid b!read\langle r\rangle)$$

$$(\nu\, b)\,(\mathsf{Empty}[b] \mid b!write\langle u\rangle \mid b!write\langle v\rangle)$$

Environment must *"respect"* a process:

$P = \mathsf{Empty}[b] \mid c? * \{think(r) = r!think\langle r\rangle\}$

is not an error, although

$P \xrightarrow{c?think\langle b\rangle} \mathsf{Empty}[b] \mid b!think\langle b\rangle \mid c? * \{think(r) = r!think\langle r\rangle\}$

Error-Free Processes

Redex: *pair object-message*

Bad $a l \tilde{v}$-redex:

$$(\nu\,\tilde{x})\,\Pi_{i\in I}\,a?M_i \mid \Pi_{j\in J}\,(\textbf{rec}\,X(\vec{x}).a?M_j)[\vec{v}_j] \mid a!l\langle\tilde{v}\rangle \mid Q$$

where l is not a method in $M_i \cup M_j$ or $\tilde{v}$ is of the wrong size, and there is no input action m with subject a such that $Q \stackrel{m}{\Longrightarrow}$.

Good processes do not contain *"persistent bad redexes"*.

$$\forall_{Q,\widetilde{m}}\, P \stackrel{\widetilde{m}}{\Longrightarrow} Q \text{ where } Q \text{ has a bad } al\tilde{v}\text{-redex implies}$$

$$\exists_{R,\widetilde{n}}\, Q \stackrel{\widetilde{n}}{\Longrightarrow} R \text{ where } R \text{ does not have a bad } al\tilde{v}\text{-redex.}$$

Algebra of Behavioural Types

(ABT — [Ravara *et al.*, 99])

1. *Labelled guarded sums* and a distinguished *external action* (unblocking, υ);

   type a with $\alpha = l.\upsilon.m + n.p$ when
   $a?\{l = b?l' = a?m,\ n = a?p\}$;

2. *Novel* notion of equivalence;

3. *Complete axiomatisation* of equivalence (that does not eliminate parallel composition).

Algebra of Behavioral Types

ABT — [Ravara *et al.*, 99]

$$\begin{array}{rll} \alpha ::= & \displaystyle\sum_{i\in I} l_i(\widetilde{\alpha}_i).\alpha_i & \textit{labeled sum} \\ \mid & \displaystyle\sum_{i\in I} \upsilon.\alpha_i & \textit{blocked sum} \\ \mid & (\alpha \parallel \beta) & \textit{parallel composition} \\ \mid & t & \textit{type variable} \\ \mid & \mu t.\alpha & \textit{recursive type} \end{array}$$

Operational Semantics of ABT

Labeled transition system

Act $\quad \sum_{i\in I} \pi_i.\alpha_i \xrightarrow{\pi_j} \alpha_j \quad (j \in I)$

Rpar $\quad \dfrac{\alpha \xrightarrow{\pi} \alpha'}{\alpha \parallel \beta \xrightarrow{\pi} \alpha' \parallel \beta}$

Lpar $\quad \dfrac{\alpha \xrightarrow{\pi} \alpha'}{\beta \parallel \alpha \xrightarrow{\pi} \beta \parallel \alpha'}$

Rec $\quad \dfrac{\alpha[\mu t.\alpha/t] \xrightarrow{\pi} \alpha'}{\mu t.\alpha \xrightarrow{\pi} \alpha'}$

Equivalence Notion of ABT

Detects blocks:

$$v.l \not\approx l \text{ and } l.v.m \not\approx l.m$$

Does not count blocks:

$$l.v.v.m \approx l.v.m$$

A bisimulation, *strong on labels, weak on unblockings.*

Static Type System for Non-uniformity

1. Syntax directed type checking system, but not compositional;
2. Two co-inductive relations on types to check that a well-typed process:
   (a) is *not blocked*;
   (b) does not contain *message never understood*.

Results

Uniqueness of Typings.

If Γ is a well-typing for P and $\Gamma \approx \Delta$,
then Δ is a well-typing for P.

Subject Reduction.

Let Γ be a well-typing for P.
If $P \to Q$, then Q is well-typed.

Correctness of the Type System.

Well-typed processes are error-free.

Further Work

On TyCO:

1. *axiomatisation* of a notion of equivalence;
2. *algorithm* for typing inference;
3. *abstract machine* for non-uniform TyCO.

On ABT:

1. *decision procedures* for $\approx$;
2. *subtyping*: semantic and syntactic notions;
3. *modal logics* and liveness properties.