Fair termination for resource-aware active objects

Francesco Dagnino*, Paola Giannini‡, Violet Ka I Pun†, Ulises Torrella†

June 12, 2025, TYPES 2025

* DIBRIS, Università di Genova, Italy
‡ DiSSTE, Università del Piemonte Orientale, Italy
† Western Norway University of Applied Sciences, Norway

Resource Aware Active Objects

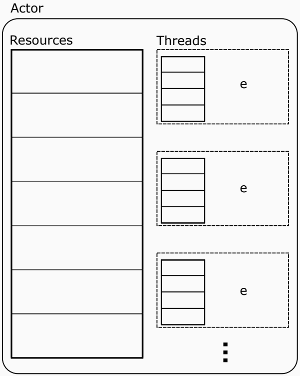
Resource aware active objects

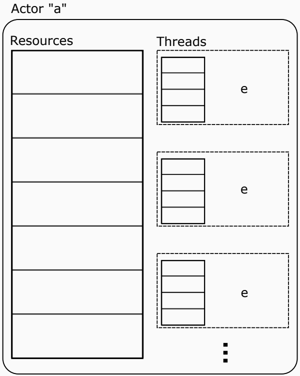
We work on a **resource aware active object language**.

Goal

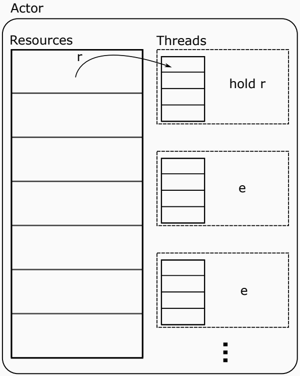
- model concurrent systems
  - where resources can be limited: linear, affine, bounded, etc.
- guarantee that the implementation has a *correct use of resources*

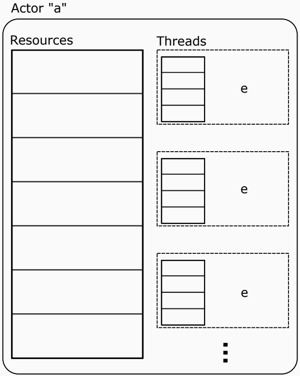
Resource aware active objects



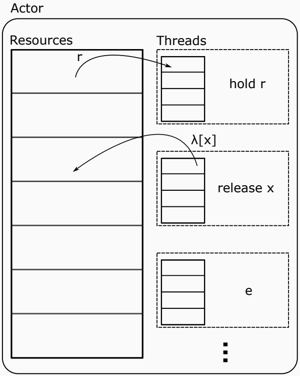


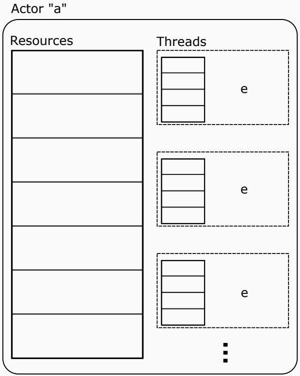
Resource aware active objects



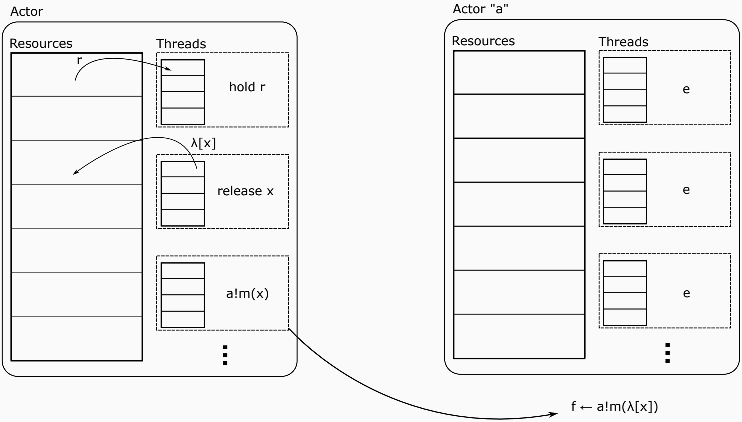


Resource aware active objects





Resource aware active objects



Modelling resources with grades

Resource aware active objects

Grades: Extra annotations on our syntax and type system

$$T^{g}, x^{g}, \texttt{hold}\ g\ r$$

- quantitatively: linearly, at most once, a bounded amount of times.
- qualitatively: privately or publicly, etc.

Subtractive grade algebra

A grade algebra parametrizes resource-awareness:

It is a structure $G = \langle |G|, \leq, +, \mathbf{0}, - \rangle$ where:

1. $\langle |G|, \leq, +, \mathbf{0} \rangle$ is an ordered commutative monoid
2. for all $g \leq \mathbf{0} \implies g = \mathbf{0}$

Subtractive grade algebra

A grade algebra parametrizes resource-awareness:

It is a structure $G = \langle |G|, \leq, +, \mathbf{0}, - \rangle$ where:

1. $\langle |G|, \leq, +, \mathbf{0} \rangle$ is an ordered commutative monoid
2. for all $g \leq \mathbf{0} \implies g = \mathbf{0}$
3. $-$ is a partial binary function such that forall $g, h, h' \in |G|$:
    3.1 if $h - g$ is defined and $h \leq h'$ then $h' - g$ is defined and monotone.
    3.2 $g + h' \leq h$ if and only if $h - g$ is defined and $h' \leq h - g$

Grades - Examples

Linear grade algebra

The linear modality is defined by $\langle |\mathrm{Lin}|, \leq, +, \mathbf{0}, - \rangle$ is defined by $|\mathrm{Lin}| = \{\mathbf{0}, 1, \infty\}$, with $0 \leq \infty$ and $1 \leq \infty$.

$$\mathbf{0} + x = x = x + \mathbf{0} \qquad x - \mathbf{0} = x$$
$$1 + 1 = \infty \qquad 1 - 1 = \mathbf{0}$$
$$\infty + x = \infty = x + \infty \qquad \infty - x = \infty$$

Privacy grade algebra

Is given by a join semi-lattice. With $+ = \vee$ and $-$ defined by $h - g = h \iff g \leq h$.

Resource aware active objects

Some characteristics of our calculus

- Actors hold resources only accessible to their executing threads.
- Futures are first-class citizens, but linear.
- Non-deterministic branching.
- Recursion.

```
Actor a {
  ρ: [r^g]

  m1 (...) {...}
  m2 (...) {...}
  ...
}
```

The expression level is the code that is executed by *actors*.

The expression level

$$\begin{array}{rcl} e & ::= & a!m(\overline{ve}) \mid ve? \\ & \mid & \texttt{hold}\ g\ r \mid \texttt{release}\ g\ ve \\ & \mid & \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \mid e_1 \oplus e_2 \\ & \mid & \texttt{return}\ ve \\ ve & ::= & x^g \mid x \mid v \\ v & ::= & r^g \mid f \mid \texttt{unit} \end{array}$$

The behaviour of expressions is defined by a labeled relation:

$$e \xrightarrow[r]{w} e'$$

The semantics

The process level

$$
\begin{array}{rcl}
P & ::= & a^{\bullet}[\lambda \mid e]^{f} \mid a^{\circ}[\lambda \mid e]^{f} \\
 & \mid & \texttt{idle}^{a} \mid \\
 & \mid & f \leftarrow v \mid f \leftarrow a!m(\overline{v}) \\
 & \mid & P \parallel Q
\end{array}
$$

$$
\begin{array}{rcl}
\rho & ::= & \overline{r^{g}} \\
\Phi & ::= & \overline{a : \rho} \\
\sigma & ::= & \Phi \mid\mid P
\end{array}
$$

The semantics

The process level

$$
\begin{array}{rcl}
P & ::= & a^{\bullet}[\lambda \mid e]^{f} \mid a^{\circ}[\lambda \mid e]^{f} \\
 & \mid & \texttt{idle}^{a} \mid \\
 & \mid & f \leftarrow v \mid f \leftarrow a!m(\overline{v}) \\
 & \mid & P \parallel Q
\end{array}
\qquad
\begin{array}{rcl}
\rho & ::= & \overline{r^{g}} \\
\Phi & ::= & \overline{a : \rho} \\
\sigma & ::= & \Phi \parallel P
\end{array}
$$

$$P \parallel Q \bowtie Q \parallel P \qquad \begin{array}{ll} \text{if} & fp(P) \cap fr(Q) = \emptyset \\ \text{and} & fp(Q) \cap fr(P) = \emptyset \end{array}$$

$$\texttt{idle}^{a} \parallel a^{\circ}[\lambda \mid e]^{f} \triangleright a^{\bullet}[\lambda \mid e]^{f}$$

$$a^{\bullet}[\lambda \mid e]^{f} \triangleright \texttt{idle}^{a} \parallel a^{\circ}[\lambda \mid e]^{f} \quad \text{if} \quad \lambda \mid e \xrightarrow[f' \leftarrow v]{}$$

The semantics

The process level

$$
\begin{array}{lcl}
P & ::= & a^{\bullet}[\lambda \mid e]^f \mid a^{\circ}[\lambda \mid e]^f \\
 & \mid & \mathtt{idle}^a \mid \\
 & \mid & f \leftarrow v \mid f \leftarrow a!m(\overline{v}) \\
 & \mid & P \parallel Q
\end{array}
\qquad
\begin{array}{lcl}
\rho & ::= & \overline{r^g} \\
\Phi & ::= & \overline{a : \rho} \\
\sigma & ::= & \Phi \parallel P
\end{array}
$$

$$
\textsc{(hold)}\ \frac{\lambda \mid e \xrightarrow[\mathtt{hold}\ r^g]{} \lambda' \mid e'}{\Phi, a : \rho, r^h \parallel a^{\bullet}[\lambda \mid e]^f \longrightarrow \Phi, a : \rho, r^{h-g} \parallel a^{\bullet}[\lambda' \mid e']^f}
$$

The Type System

The type system

Contexts

$$\Gamma ::= \overline{x : T} \qquad \Sigma ::= \overline{f : \mathtt{Fut}\langle T, \Phi \rangle} \qquad \Phi ::= \overline{a : \rho}$$

Types

$$T ::= \mathtt{Unit} \mid r^g \mid \mathtt{Fut}\langle T, \Phi \rangle$$

The type system

Contexts

$$\Gamma ::= \overline{x : T} \qquad \Sigma ::= \overline{f : \texttt{Fut}\langle T, \Phi \rangle} \qquad \Phi ::= \overline{a : \rho}$$

Types

$$T ::= \texttt{Unit} \mid r^g \mid \texttt{Fut}\langle T, \Phi \rangle$$

To type an expression we consider resources going in and out:

$$\Phi \vdash e : T; \Phi'$$

The type system

Contexts

$$\Gamma ::= \overline{x : T} \qquad \Sigma ::= \overline{f : \texttt{Fut}\langle T, \Phi \rangle} \qquad \Phi ::= \overline{a : \rho}$$

The full typing judgement for expressions:

$$\Phi;\ \Sigma;\ \Gamma \vdash_a e : T;\ \Phi'$$

The type system

Contexts

$$\Gamma ::= \overline{x : T} \qquad \Sigma ::= \overline{f : \mathtt{Fut}\langle T, \Phi \rangle} \qquad \Phi ::= \overline{a : \rho}$$

The full typing judgement for expressions:

$$\Phi;\ \Sigma;\ \Gamma \vdash_a e : T;\ \Phi'$$

The typing judgement for processes:

$$\Phi;\ \Sigma \vdash P :: \Sigma'$$

The type system

Contexts

$$\Gamma ::= \overline{x : T} \qquad \Sigma ::= \overline{f : \texttt{Fut}\langle T, \Phi\rangle} \qquad \Phi ::= \overline{a : \rho}$$

One more thing:

The full typing judgement for expressions:

$$\Phi;\ \Sigma;\ \Gamma \vdash_a^{n} e : T;\ \Phi'$$

The typing judgement for processes:

$$\Phi;\ \Sigma \vdash^{n} P :: \Sigma'$$

A terminated configuration is only conformed of idle actors and resolved messages ($f \leftarrow v$)

The type system

Typing a let expression must track resources:

$$
(\textsc{T-let})\quad \frac{\Phi_1;\ \Sigma_1;\ \Gamma_1 \vdash_a^m e_1 : T';\ \Phi_1' + \Phi_2' \qquad \Phi_2 + \Phi_1';\ \Sigma_2;\ \Gamma_2, x : T' \vdash_a^n e_2 : T;\ \Psi_2}{\Phi_1 + \Phi_2;\ \Sigma_1, \Sigma_2;\ \Gamma_1 + \Gamma_2 \vdash_a^{1+n+m} \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : T;\ \Phi_2' + \Psi_2}
$$

The type system

Typing a let expression must track resources:

$$\text{(T-LET)}\ \frac{\Phi_1;\ \Sigma_1;\ \Gamma_1 \vdash_a^m e_1 : T';\ \Phi_1' + \Phi_2' \qquad \Phi_2 + \Phi_1';\ \Sigma_2;\ \Gamma_2, x : T' \vdash_a^n e_2 : T;\ \Psi_2}{\Phi_1 + \Phi_2;\ \Sigma_1, \Sigma_2;\ \Gamma_1 + \Gamma_2 \vdash_a^{1+n+m} \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : T;\ \Phi_2' + \Psi_2}$$

Typing a parallel composition must control futures:

$$\text{(T-PAR)}\ \frac{\Phi_1;\ \Sigma_1 \vdash^n P :: \Sigma_1', \Sigma_1'' \qquad \Phi_2;\ \Sigma_2, \Sigma_1' \vdash^m Q :: \Sigma_2'}{\Phi_1 + \Phi_2;\ \Sigma_1, \Sigma_2 \vdash^{n+m} P \parallel Q :: \Sigma_2', \mathsf{mark}(\Sigma_1'), \Sigma_1''} \quad \mathsf{dom}(\Sigma_1'') \cap \mathsf{dom}(\Sigma_2) = \emptyset$$

Soundness

We say that our system is correct if it is resource-safe.

Resource safe: every hold will be successful

Soundness

We say that our system is correct if it is resource-safe.

Resource safe: every hold will be successful

Theorem (Subject reduction)

If $\Phi;\ \Sigma \vdash^n_\Theta \sigma :: \Sigma'$ *and* $\sigma \longrightarrow \sigma'$*, then it exists* Ψ, m, Ω' *such that* $\Psi;\ \Sigma \vdash^m_\Theta \sigma' :: \Omega'$*, where* $\Omega' = \Sigma', \mathsf{mark}(\Omega'')$ *with* Ω'' *fresh.*

Soundness

Theorem (Weak termination)

If $\Phi; \emptyset \vdash^n \sigma :: \Sigma'$ then there exists a reduction $\sigma \longrightarrow^* \sigma'$ such that σ' is terminated.

Theorem (Fair termination)

If $\Phi; \emptyset \vdash^n \sigma :: \Sigma'$, then if $\sigma \longrightarrow \sigma'$ implies σ' is weakly terminating.

Well-typed configurations are fairly terminating,

$\implies$ we know it can never be stuck,

$\implies$ it's **resource-safe**.

Moreover, it cannot be live-locked, and there cannot be orphan messages.

Conclusion and Future work

We introduce an active object language for workflow modelling with parametrized resource-awareness.

We implement a type system that guarantees that all modelled workflow systems are resource-safe.

Conclusion and Future work

We introduce an active object language for workflow modelling with parametrized resource-awareness.

We implement a type system that guarantees that all modelled workflow systems are resource-safe.

Future work:

- Make the system fully object-oriented
- Graded futures

Conclusion and Future work

We introduce an active object language for workflow modelling with parametrized resource-awareness.

We implement a type system that guarantees that all modelled workflow systems are resource-safe.

Future work:

- Make the system fully object-oriented
- Graded futures

Thanks for your attention, questions?