Type Systems for Resource Use of Component Software

Marc Bezem Hoang Truong

Department of Informatics
University of Bergen

TYPES 2006
18-21 April, Nottingham, UK

Introduction

- We study some abstract component languages and develop type systems that
  - allow one to derive upper bounds of simultaneously active instances of every involved components.

- This talk describes the language with: instantiation, deallocation, composition and its type system.

Syntax

- Main features: instantiation, deallocation, and composition

$\mathbb{C}$	$=$	$a, .., z$	Component names
$Prog$	$::=$	$\overline{Decls}; E$	Program
$Decls$	$::=$	$\overline{x \prec E}$	Declarations, $x \in \mathbb{C}$
$A, .., E$	$::=$		Expressions
		ϵ	Empty
	$\mid$	$\texttt{new}\, x$	Instantiation
	$\mid$	$\texttt{del}\, x$	Deallocation
	$\mid$	$E\ E$	Sequential
	$\mid$	$(E + E)$	Choice
	$\mid$	$(E \parallel E)$	Parallel
	$\mid$	$\{E\}$	Scope

- Standard BNF; overbar for Kleene closure.

An example program

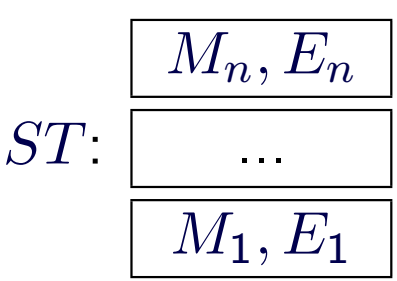
Example

$$
\begin{aligned}
& d \prec \epsilon \quad e \prec \epsilon \\
& a \prec (\{\,\texttt{new}\,d\,\}\,\texttt{new}\,e \parallel \texttt{new}\,d\,)\,\texttt{del}\,d \\
& b \prec (\,\texttt{new}\,a + \texttt{new}\,e\,\texttt{new}\,d\,)\,\texttt{del}\,e\,; \\
& \texttt{new}\,b
\end{aligned}
$$

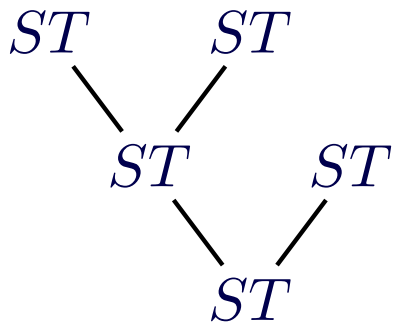
- d, e are primitive, $\texttt{new}\,b$ is the main expression.

Small-step operational semantics

- Transition between configurations
- A configuration $\mathbb{T}$ is a binary tree of threads.
  - A thread ST is a stack of pairs (M, E) of a local store M and an expression E.
  - Local store M is a multiset over component names. Each element of M represents an instance.



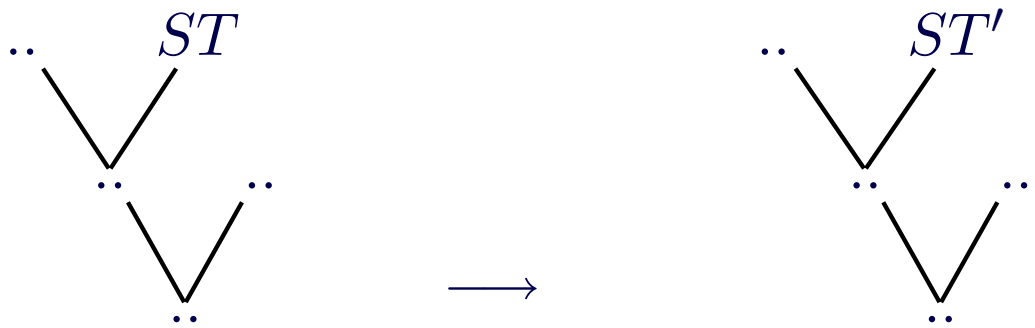
Stack/thread



Configuration

Small-step operational semantics

- Terminal configuration: (M, ϵ).
- Define one-step transition based on patterns of branches:



- Next: define rules for ST and ST':

Rules for patterns $\Rightarrow$

Rules for `new` and `del` :

(osNew)

$$\frac{M,\ \texttt{new}\ x\ E \qquad \vdots \qquad x \prec A \in Decls}{M + x,\ AE \qquad \vdots} \Rightarrow$$

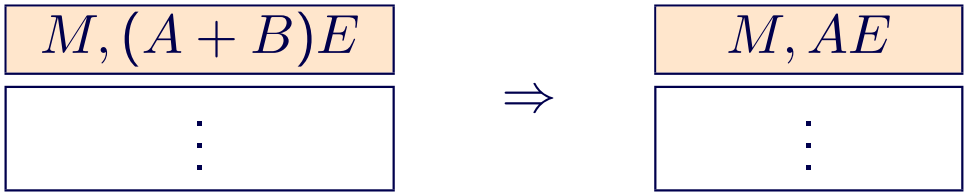
(osDel)

$$\frac{M,\ \texttt{del}\ x\ E \qquad \vdots \qquad x \in M}{M - x,\ E \qquad \vdots} \Rightarrow$$

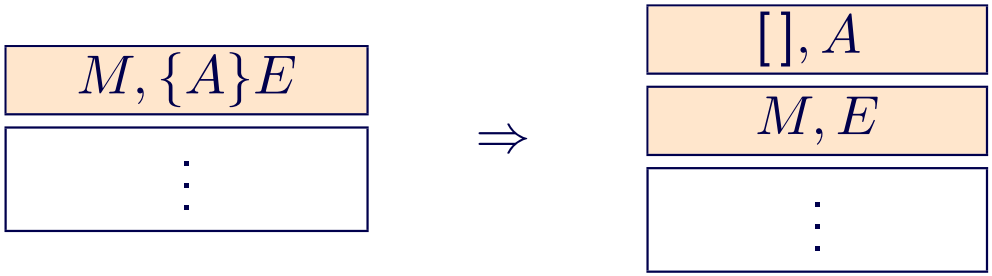
Rules for patterns (cont.)

Rules for choice and scope:

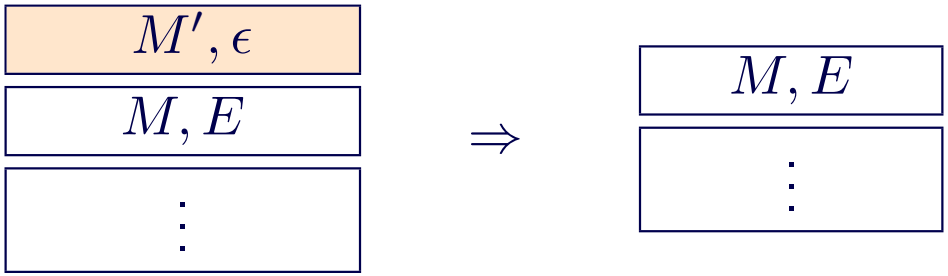
(osChoice)



(osPush)



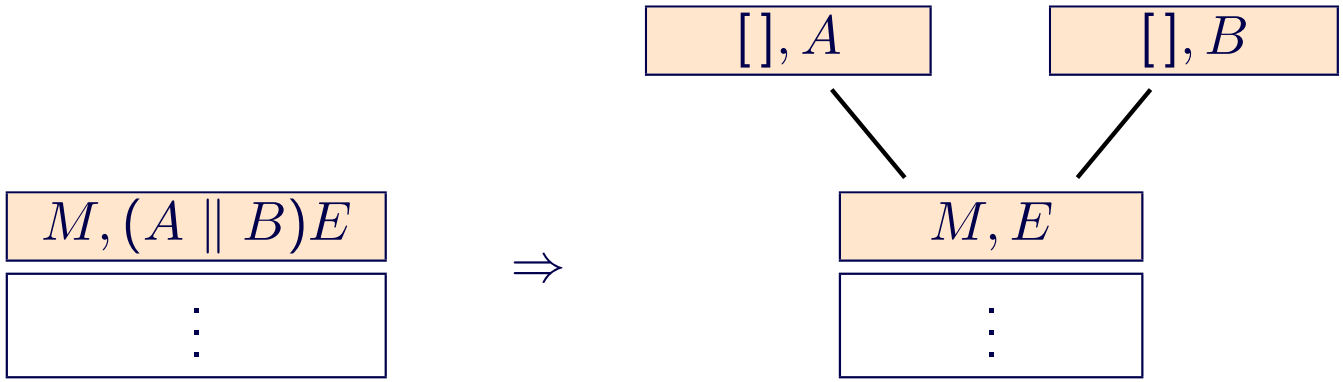
(osPop)



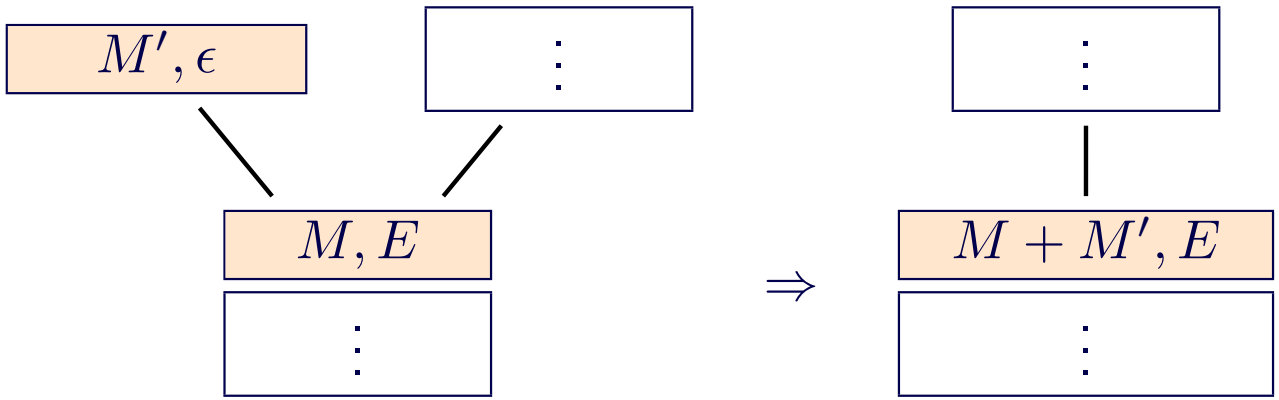
Rules for patterns (cont.)

Rules for parallel composition:

(osParIntr)



(osParElim)



Running the example program

- Running the example program

Start $\boxed{[\,],\ \texttt{new}\ b}$

osNew $\longrightarrow$ $\boxed{[b],\ (\ \texttt{new}\ a\ +\ \texttt{new}\ e\ \texttt{new}\ d\ )\ \texttt{del}\ e}$

osChoice $\longrightarrow$ $\boxed{[b],\ \texttt{new}\ a\ \texttt{del}\ e}$ (or $\boxed{[b],\ \texttt{new}\ e\ \texttt{new}\ d\ \texttt{del}\ e}$)

(osNew) $\longrightarrow$ $\boxed{[b, a],\ (\{\ \texttt{new}\ d\ \}\ \texttt{new}\ e\ \parallel\ \texttt{new}\ d\ )\ \texttt{del}\ d\ \texttt{del}\ e}$

$$a \prec (\{\ \texttt{new}\ d\ \}\ \texttt{new}\ e\ \parallel\ \texttt{new}\ d\ )\ \texttt{del}\ d$$
$$b \prec (\ \texttt{new}\ a\ +\ \texttt{new}\ e\ \texttt{new}\ d\ )\ \texttt{del}\ e\,;$$

Running the example program (cont.)

$$[b, a], (\{\,\texttt{new}\ d\,\}\,\texttt{new}\ e \parallel \texttt{new}\ d)\,\texttt{del}\ d\ \texttt{del}\ e$$

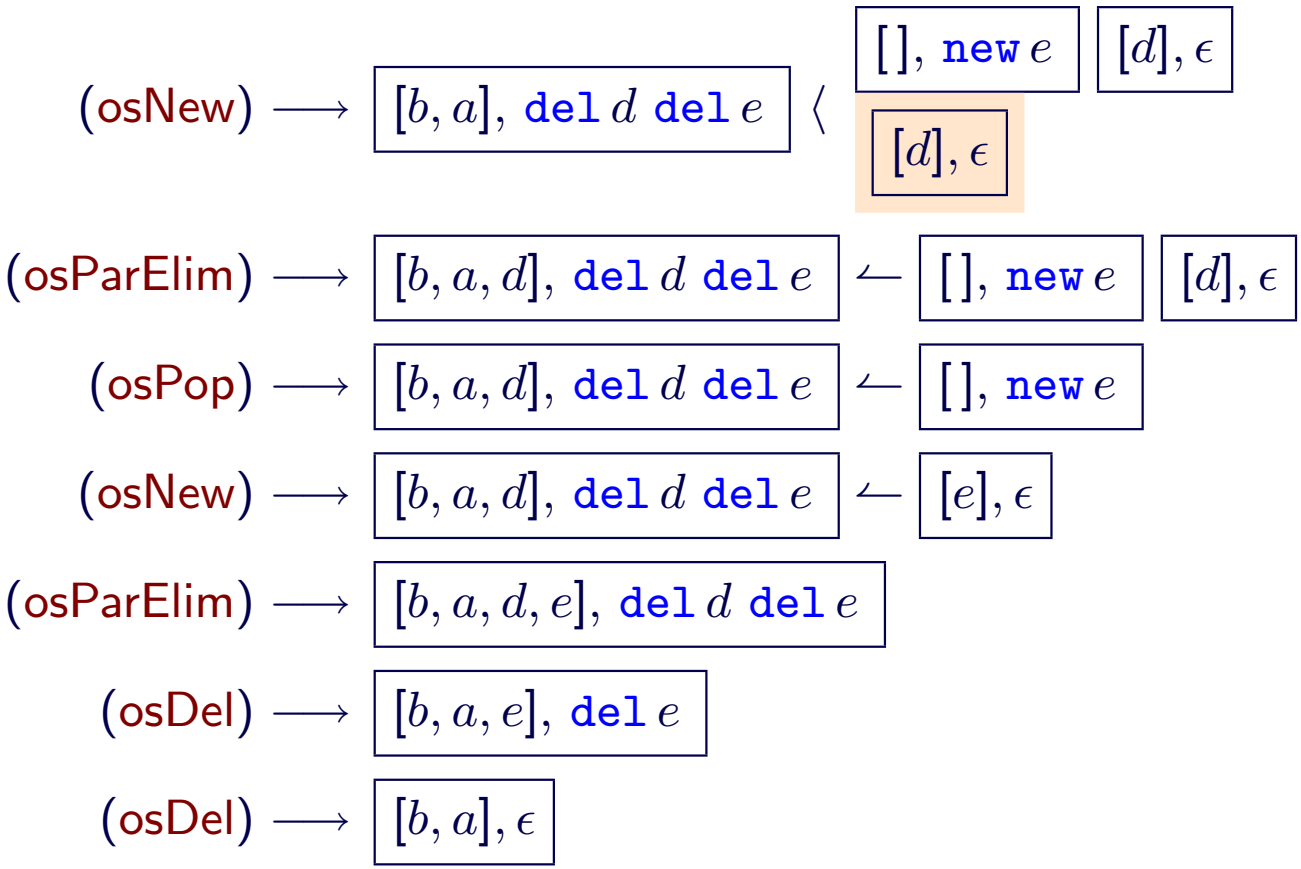
(osParIntr) $\longrightarrow$ $[b, a],\ \texttt{del}\ d\ \texttt{del}\ e$ $\langle$ $[], \{\,\texttt{new}\ d\,\}\,\texttt{new}\ e$ | $[],\ \texttt{new}\ d$

(osNew) $\longrightarrow$ $[b, a],\ \texttt{del}\ d\ \texttt{del}\ e$ $\langle$ $[], \{\,\texttt{new}\ d\,\}\,\texttt{new}\ e$ | $[d], \epsilon$

(osPush) $\longrightarrow$ $[b, a],\ \texttt{del}\ d\ \texttt{del}\ e$ $\langle$ $[],\ \texttt{new}\ e$ | $[],\ \texttt{new}\ d$ | $[d], \epsilon$

$$d \prec \epsilon \qquad e \prec \epsilon$$

Running the example program (cont.)



What is the maximum number of instances of each component in all possible states?

Type system goals

Goals of the type system:

- Find the maximum number of simultaneously active instances for every components,
- Ensure the safety of deallocation primitive `del` (cannot delete a nonexistent instance),
- Rule out recursion or mutual recursion in declarations,
- Be compositional (type of an expression can be computed from types of its subexpressions),
- Be decidable.

Type system

- Types are tuples of a multiset and two signed multisets over component names:

$$X = \langle X^i, X^o, X^l \rangle$$

  X^i: multiset, X^o, X^l: signed multisets.

- Typing judgment:

$$\sigma, \Gamma \vdash E : X$$

  where store σ is a multiset over component names (for the safety of `del` in E) and basis Γ is a list of declarations.

- $X^i(x)$: **maximum number of instances of x during the execution of E,**
- $X^o(x)$: maximum change in the number of instances of x after the execution of E,
- $X^l(x)$: minimum change in the number of instances of x after the execution of E.

Typing rules

- Rules for startup, `new` and `del` :

(Axiom)

$$\frac{}{[\,],\varnothing \vdash \epsilon : \langle [\,],[\,],[\,]\rangle}$$

(New)

$$\frac{\sigma,\Gamma \vdash A : X \quad x \notin \mathsf{dom}(\Gamma)}{\sigma,\Gamma,x \prec A \vdash \texttt{new}\, x : \langle X^i + x, X^o + x, X^l + x\rangle}$$

(Del)

$$\frac{\sigma,\Gamma \vdash A : X \quad x \in \mathsf{dom}(\Gamma)}{[x],\Gamma \vdash \texttt{del}\, x : \langle [\,],[-x],[-x]\rangle}$$

Typing rules (cont.)

- Sequencing two expressions A and B:

(Seq)

$$\frac{\sigma_1, \Gamma \vdash A : X \quad \sigma_2, \Gamma \vdash B : Y \quad A, B \neq \epsilon}{\sigma_1 \cup (\sigma_2 - X^l), \Gamma \vdash AB : \langle X^i \cup (X^o + Y^i), X^o + Y^o, X^l + Y^l \rangle}$$

For safety of B, it is required that $\sigma_1 + X^l \supseteq \sigma_2$, so AB requires $\sigma_2 - X^l$ for the safety of B in composition.

Typing rules (cont.)

- Rules for choice, scope, and parallel composition:

(Choice)

$$\frac{\sigma_1, \Gamma \vdash A : X \quad \sigma_2, \Gamma \vdash B : Y}{\sigma_1 \cup \sigma_2, \Gamma \vdash (A + B) : \langle X^i \cup Y^i, X^o \cup Y^o, X^l \cap Y^l \rangle}$$

(Parallel)

$$\frac{[], \Gamma \vdash A : X \quad [], \Gamma \vdash B : Y}{[], \Gamma \vdash (A \parallel B) : \langle X^i + Y^i, X^o + Y^o, X^l + Y^l \rangle}$$

(Scope)

$$\frac{[], \Gamma \vdash A : X}{[], \Gamma \vdash \{A\} : \langle X^i, [], [] \rangle}$$

Typing rules (cont.)

- Weakening rules for store and basis:

$$\text{(WeakenS)}\quad \frac{\sigma, \Gamma \vdash A : X \quad \sigma \subseteq \sigma_1}{\sigma_1, \Gamma \vdash A : X}$$

$$\text{(WeakenB)}\quad \frac{\sigma_1, \Gamma \vdash A : X \quad \sigma_2, \Gamma \vdash B : Y \quad x \notin \mathsf{dom}(\Gamma)}{\sigma_1, \Gamma, x \prec B \vdash A : X}$$

Typing examples

$$\mathsf{We}\,\frac{\mathsf{Sc}\,\frac{\mathsf{Ne}\,\frac{\vdash \epsilon : \langle\,\rangle}{[\,], d \prec \epsilon \vdash \mathtt{new}\, d : \langle [d], [d], [d] \rangle}}{[\,], d \prec \epsilon \vdash \{\, \mathtt{new}\, d \,\} : \langle [d], [\,], [\,] \rangle} \quad \mathsf{We}\,\frac{\vdash \epsilon : \langle\,\rangle \quad \vdash \epsilon : \langle\,\rangle}{[\,], d \prec \epsilon \vdash \epsilon : \langle\,\rangle}}{[\,], d \prec \epsilon, e \prec \epsilon \vdash \{\, \mathtt{new}\, d \,\} : \langle [d], [\,], [\,] \rangle} \tag{1}$$

$$\mathsf{Se}\,\frac{(1) \quad \mathsf{Ne}\,\frac{\mathsf{We}\,\frac{\vdash \epsilon : \langle\,\rangle \quad \vdash \epsilon : \langle\,\rangle}{[\,], d \prec \epsilon \vdash \epsilon : \langle\,\rangle}}{[\,], d \prec \epsilon, e \prec \epsilon \vdash \mathtt{new}\, e : \langle [e], [e], [e] \rangle}}{[\,], d \prec \epsilon, e \prec \epsilon \vdash \{\, \mathtt{new}\, d \,\}\, \mathtt{new}\, e : \langle [d, e], [e], [e] \rangle} \tag{2}$$

Soundness and Type Inference

- We proved the soundness using the standard technique:
  - notion of well-typed configurations,
  - Preservation and Progress lemmas.

- We have a polynomial type inference algorithm. (All possible runs are exponential.)

- More info.: http://www.ii.uib.no/~hoang/

Thank you.