Types for Complexity of Parallel Computation in π-Calculus

Alexis Ghyselen, joint work with Patrick Baillot

University of Bologna

GT Scalp, 3rd of November 2021

Introduction

Goal

Obtaining time complexity properties or bounds with a type system

Typical Result

If $\vdash t : \mathsf{Nat} \to \mathsf{Nat}$ then, for any integer input n, we can extract a bound on the computation time of $t\ n$

Examples, for functional languages

Hughes, Pareto, Sabry '96: Sized Types
Hofmann '03 : Non-size-increasing Types
Dal Lago, Gaboardi '11: Linear Dependent Types

Type-Based Complexity Analysis

Important Questions

- *Soundness*: Can we extract a complexity bound from a type derivation ?

Type-Based Complexity Analysis

Important Questions

- *Soundness*: Can we extract a complexity bound from a type derivation ?
- *Type-Inference*: Can we automatically obtain complexity bounds by inferring a type ?

Type-Based Complexity Analysis

Important Questions

- *Soundness*: Can we extract a complexity bound from a type derivation ?
- *Type-Inference*: Can we automatically obtain complexity bounds by inferring a type ?
- *Expressivity*: What are the useful programs that can be typed ?

Type-Based Complexity Analysis

Important Questions

- *Soundness*: Can we extract a complexity bound from a type derivation ?
- *Type-Inference*: Can we automatically obtain complexity bounds by inferring a type ?
- *Expressivity*: What are the useful programs that can be typed ?
- *Precision*: How sharp are the complexity bounds ?

Parallel Complexity

Complexity in a Calculus with Parallelism

- Work : Total time complexity without parallelism
- Span : Time complexity with maximal parallelism

Parallel Complexity

Complexity in a Calculus with Parallelism

- Work : Total time complexity without parallelism
- Span : Time complexity with maximal parallelism
- Width : Number of processors for maximal parallelism
- Practical Complexity : Time complexity with p processors

Parallel Complexity

Complexity in a Calculus with Parallelism

- Work : Total time complexity without parallelism
- Span : Time complexity with maximal parallelism
- Width : Number of processors for maximal parallelism
- Practical Complexity : Time complexity with p processors

A Classical Result

From work and span, we can deduce a bound on practical complexity

Parallel Complexity

Complexity in a Calculus with Parallelism

- Work : Total time complexity without parallelism
- Span : Time complexity with maximal parallelism
- Width : Number of processors for maximal parallelism
- Practical Complexity : Time complexity with p processors

A Classical Result

From work and span, we can deduce a bound on practical complexity

A Calculus for Concurrent and Parallel Computation

We work on the π-calculus, because it is simple, expressive and wide-spread

Goal

Define several sized type systems to obtain complexity bounds in π-calculus.

The π-calculus

Paradigm of the π-calculus

Parallelism

Communication with channels

Channels can send values and names of channels

The π-calculus

Paradigm of the π-calculus

Parallelism

Communication with channels

Channels can send values and names of channels

Dynamic Aspects

Dynamic creation of new processes

Dynamic creation of channels

The π-calculus

Paradigm of the π-calculus

Parallelism

Communication with channels

Channels can send values and names of channels

Dynamic Aspects

Dynamic creation of new processes

Dynamic creation of channels

Model of Concurrency

Useful to study the equivalence of processes

Encoding of functional languages in the π-calculus

The π-calculus

Base Syntax

$$P := 0 \mid (P \mid Q) \mid (\nu a)P \mid \overline{a}\langle \tilde{e} \rangle \mid a(\tilde{v}).P \mid !a(\tilde{v}).P \mid \texttt{tick}.P$$

The π-calculus

Base Syntax

$$P := 0 \mid (P \mid Q) \mid (\nu a)P \mid \overline{a}\langle \tilde{e} \rangle \mid a(\tilde{v}).P \mid !a(\tilde{v}).P \mid \texttt{tick}.P$$

Example with integers

$$Q = a(r).r(n).\overline{r}\langle n+1 \rangle$$

The π-calculus

Base Syntax

$P := 0 \mid (P \mid Q) \mid (\nu a)P \mid \overline{a}\langle \tilde{e} \rangle \mid a(\tilde{v}).P \mid !a(\tilde{v}).P \mid \texttt{tick}.P$

Example with integers

$Q = a(r).r(n).\overline{r}\langle n+1 \rangle$

Structural Congruence

Associativity and Commutativity of Parallel Composition . . .

The π-calculus

Base Syntax

$$P := 0 \mid (P \mid Q) \mid (\nu a)P \mid \overline{a}\langle \tilde{e} \rangle \mid a(\tilde{v}).P \mid !a(\tilde{v}).P \mid \mathtt{tick}.P$$

Example with integers

$$Q = a(r).r(n).\overline{r}\langle n+1 \rangle$$

Structural Congruence

Associativity and Commutativity of Parallel Composition . . .

Semantics

$$a(\tilde{v}).P \mid \overline{a}\langle \tilde{e} \rangle \rightarrow P[\tilde{v} := \tilde{e}]$$
$$!a(\tilde{v}).P \mid \overline{a}\langle \tilde{e} \rangle \rightarrow P[\tilde{v} := \tilde{e}] \mid\ !a(\tilde{v}).P$$

The π-calculus

Base Syntax

$P := 0 \mid (P \mid Q) \mid (\nu a)P \mid \overline{a}\langle \tilde{e} \rangle \mid a(\tilde{v}).P \mid !a(\tilde{v}).P \mid \texttt{tick}.P$

Example with integers

$Q = a(r).r(n).\overline{r}\langle n+1 \rangle$

Structural Congruence

Associativity and Commutativity of Parallel Composition . . .

Semantics

$$a(\tilde{v}).P \mid \overline{a}\langle \tilde{e} \rangle \rightarrow P[\tilde{v} := \tilde{e}]$$
$$!a(\tilde{v}).P \mid \overline{a}\langle \tilde{e} \rangle \rightarrow P[\tilde{v} := \tilde{e}] \mid \; !a(\tilde{v}).P$$

Example

$Q \mid \overline{a}\langle b \rangle \mid \overline{b}\langle 4 \rangle \rightarrow b(n).\overline{b}\langle n+1 \rangle \mid \overline{b}\langle 4 \rangle \rightarrow \overline{b}\langle 5 \rangle$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n \ \textit{times}} \qquad W = n \qquad S = 1$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n\ \ times} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

Examples for Work and Span

$$\underbrace{\mathtt{tick} \mid \mathtt{tick} \mid \mathtt{tick} \mid \cdots}_{n\ \ times} \qquad W = n \qquad S = 1$$

$$a().\mathtt{tick} \mid \overline{a}\langle\rangle \mid \mathtt{tick} \qquad W = 2 \qquad S = 1$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n\ \textit{times}} \qquad W = n \qquad S = 1$$

$$\texttt{tick} \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n \ times} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

$$a().\texttt{tick}.P_0 \mid \texttt{tick}.a().P_1 \mid \overline{a}\langle\rangle \qquad S = \max(1 + C_0, 1 + C_1)$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n \ \textit{times}} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

$$\color{red}{a().}\color{black}\texttt{tick}.P_0 \mid \texttt{tick}.a().P_1 \mid \color{red}{\overline{a}\langle\rangle} \qquad \color{black} S = \max(1 + C_0, 1 + C_1)$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n\ \textit{times}} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

$$\texttt{tick}.P_0 \mid \texttt{tick}.a().P_1 \qquad S = \max(1 + C_0, 1 + C_1)$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n \textit{ times}} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

$$a().\texttt{tick}.P_0 \mid \texttt{tick}.a().P_1 \mid \overline{a}\langle\rangle \qquad S = \max(1 + C_0, 1 + C_1)$$

Examples for Work and Span

$$\underbrace{\texttt{tick} \mid \texttt{tick} \mid \texttt{tick} \mid \cdots}_{n\ times} \qquad W = n \qquad S = 1$$

$$a().\texttt{tick} \mid \overline{a}\langle\rangle \mid \texttt{tick} \qquad W = 2 \qquad S = 1$$

$$a().\texttt{tick}.P_0 \mid \texttt{tick}.a().P_1 \mid \overline{a}\langle\rangle \qquad S = \max(1 + C_0, 1 + C_1)$$

$$!a(n).\texttt{tick}.\texttt{if}\ (n = 0)\ \texttt{then}\ 0\ \texttt{else}\ \overline{a}\langle n-1\rangle \mid \overline{a}\langle n-1\rangle$$

$$W = O(2^{|n|}) \qquad S = O(|n|)$$

Reduction with Cost

Work

$$\frac{P \to Q \text{ in standard } \pi\text{-calculus}}{P \to^0 Q} \qquad \frac{}{\texttt{tick}.P \to^1 P}$$

Reduction with Cost

Work

$$\frac{P \rightarrow Q \text{ in standard } \pi\text{-calculus}}{P \rightarrow^0 Q} \qquad \frac{}{\texttt{tick}.P \rightarrow^1 P}$$

Work = Maximal number of $\rightarrow^1$

Reduction with Cost

Work

$$\frac{P \to Q \text{ in standard } \pi\text{-calculus}}{P \to^0 Q} \qquad \frac{}{\texttt{tick}.P \to^1 P}$$

Work $=$ Maximal number of $\to^1$

Span

We give a new formalization by defining annotated processes

Annotated Processes

Syntax

New constructor "$n : P$"

Annotated Processes

Syntax

New constructor "$n : P$"
"P with n ticks before"

Annotated Processes

Syntax

New constructor "$n : P$"
"P with n ticks before"

Standard rules for structural congruence
$+$

Annotated Processes

Syntax

New constructor "$n : P$"
"P with n ticks before"

Standard rules for structural congruence

$$+$$

$$m : (P \mid Q) \equiv (m : P) \mid (m : Q) \qquad m : (\nu a)P \equiv (\nu a)(m : P)$$

$$m : (n : P) \equiv (m + n) : P \qquad 0 : P \equiv P$$

Parallel Complexity

$$\frac{}{\texttt{tick}.P \Rightarrow (1 : P)}$$

Parallel Complexity

$$\frac{}{\texttt{tick}.P \Rightarrow (1 : P)}$$

$$\frac{}{(n : a(\tilde{v}).P) \mid (m : \overline{a}\langle\tilde{e}\rangle) \Rightarrow \max(m, n) : P[\tilde{v} := \tilde{e}]}$$

Parallel Complexity

$$\frac{}{\texttt{tick}.P \Rightarrow (1 : P)}$$

$$\frac{}{(n : a(\tilde{v}).P) \mid (m : \overline{a}\langle\tilde{e}\rangle) \Rightarrow \max(m,n) : P[\tilde{v} := \tilde{e}]}$$

$$\frac{}{(n :!a(\tilde{v}).P) \mid (m : \overline{a}\langle\tilde{e}\rangle) \Rightarrow (\max(m,n) : P[\tilde{v} := \tilde{e}]) \mid (n :!a(\tilde{v}).P)}$$

Parallel Complexity

$$\frac{}{\texttt{tick}.P \Rightarrow (1 : P)}$$

$$\frac{}{(n : a(\tilde{v}).P) \mid (m : \overline{a}\langle\tilde{e}\rangle) \Rightarrow \max(m, n) : P[\tilde{v} := \tilde{e}]}$$

$$\frac{}{(n : !a(\tilde{v}).P) \mid (m : \overline{a}\langle\tilde{e}\rangle) \Rightarrow (\max(m, n) : P[\tilde{v} := \tilde{e}]) \mid (n : !a(\tilde{v}).P)}$$

Parallel Complexity of P (Span)

Maximal n such that $P \Rightarrow^* Q$ and $Q \equiv n : Q_0 \mid Q_1$

Parallel Complexity

$$\frac{}{\texttt{tick}.P \Rightarrow (1 : P)}$$

$$\frac{}{(n : a(\tilde{v}).P) \mid (m : \overline{a}\langle \tilde{e} \rangle) \Rightarrow \max(m,n) : P[\tilde{v} := \tilde{e}]}$$

$$\frac{}{(n : !a(\tilde{v}).P) \mid (m : \overline{a}\langle \tilde{e} \rangle) \Rightarrow (\max(m,n) : P[\tilde{v} := \tilde{e}]) \mid (n : !a(\tilde{v}).P)}$$

Parallel Complexity of P (Span)

Maximal n such that $P \Rightarrow^* Q$ and $Q \equiv n : Q_0 \mid Q_1$

Remark

Complexity does not necessarily decrease with a reduction step

Standard Simple Types for π-Calculus

Syntax

$T := \mathsf{Nat} \mid \mathsf{Bool} \mid \cdots \mid \mathsf{ch}(\tilde{T})$

Context

A context Γ gives a type to channel names and variables

Standard Simple Types for π-Calculus

Syntax

$T := \mathsf{Nat} \mid \mathsf{Bool} \mid \cdots \mid \mathsf{ch}(\tilde{T})$

Context

A context Γ gives a type to channel names and variables

$$\frac{\Gamma \vdash a : \mathsf{ch}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P}{\Gamma \vdash a(\tilde{v}).P}$$

Standard Simple Types for π-Calculus

Syntax

$T := \mathsf{Nat} \mid \mathsf{Bool} \mid \cdots \mid \mathsf{ch}(\tilde{T})$

Context

A context Γ gives a type to channel names and variables

$$\frac{\Gamma \vdash a : \mathsf{ch}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P}{\Gamma \vdash a(\tilde{v}).P} \qquad \frac{\Gamma \vdash a : \mathsf{ch}(\tilde{T}) \qquad \Gamma \vdash \tilde{e} : \tilde{T}}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle}$$

Goal

If $\Gamma \vdash P \lhd K$ then the worst-case work for P is bounded by K.

Simple Types with Sizes

Integer Expressions

$I, J, K := i, j, k \mid f(I_1, \ldots, I_n)$

Simple Types with Sizes

Integer Expressions

$I, J, K := i, j, k \mid f(I_1, \ldots, I_n)$

Base Types with Sizes

$\mathsf{Nat}[I, J]$ is a type for integers n with $I \leq n \leq J$

Types

$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})$

Simple Types with Sizes

Integer Expressions

$I, J, K := i, j, k \mid f(I_1, \dots, I_n)$

Base Types with Sizes

Nat$[I, J]$ is a type for integers n with $I \leq n \leq J$

Types

$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})$

Example

$\mathsf{ch}(\mathsf{Nat}[2, 7])$ $\mathsf{ch}(\mathsf{Nat}[0, i], \mathsf{ch}(\mathsf{Nat}[0, i]))$

Simple Types with Sizes

Integer Expressions

$I, J, K := i, j, k \mid f(I_1, \ldots, I_n)$

Base Types with Sizes

$\mathsf{Nat}[I, J]$ is a type for integers n with $I \leq n \leq J$

Types

$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})$

Example

$\mathsf{ch}(\mathsf{Nat}[2, 7])$ $\quad$ $\mathsf{ch}(\mathsf{Nat}[0, i], \mathsf{ch}(\mathsf{Nat}[0, i]))$

And Subtyping ?

Usual subtyping can be recovered with input/output types

Some Work Typing Rules

$$\frac{\Gamma \vdash P \triangleleft K}{\Gamma \vdash \mathtt{tick}.P \triangleleft K + 1}$$

Some Work Typing Rules

$$\frac{\Gamma \vdash P \lhd K}{\Gamma \vdash \mathtt{tick}.P \lhd K + 1}$$

$$\frac{\Gamma \vdash P \lhd K \qquad \Gamma \vdash Q \lhd K'}{\Gamma \vdash P \mid Q \lhd K + K'}$$

Example for replicated input

$$P := !a(n).\texttt{tick}.\texttt{if}\ (n = 0)\ \texttt{then}\ 0\ \texttt{else}\ \overline{a}\langle n-1\rangle \mid \overline{a}\langle n-1\rangle$$

Example for replicated input

$$P := !a(n).\mathtt{tick}.\mathtt{if}\ (n = 0)\ \mathtt{then}\ 0\ \mathtt{else}\ \overline{a}\langle n-1\rangle \mid \overline{a}\langle n-1\rangle$$

$$\text{Work} = 2^{|n|+1} - 1$$

Example for replicated input

$$P :=\, !a(n).\mathtt{tick}.\mathtt{if}\ (n = 0)\ \mathtt{then}\ 0\ \mathtt{else}\ \overline{a}\langle n - 1\rangle \mid \overline{a}\langle n - 1\rangle$$

Work $= 2^{|n|+1} - 1$

We need a complexity that depends on the size of n

The complexity can only be known when an actual integer is sent

Type for replicated input ($!a(\widetilde{v}).P$)

$\forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T})$

K stands for complexity and can depend on $\tilde{i}$

Type for replicated input ($!a(\widetilde{v}).P$)

$\forall \tilde{i}.\mathsf{serv}^K(\tilde{T})$

K stands for complexity and can depend on $\tilde{i}$

$$P := !a(n).\texttt{tick}.\texttt{if }(n = 0)\texttt{ then }0\texttt{ else }\overline{a}\langle n-1\rangle \mid \overline{a}\langle n-1\rangle$$

Type for replicated input ($!a(\widetilde{v}).P$)

$\forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T})$

K stands for complexity and can depend on $\tilde{i}$

$$P := !a(n).\texttt{tick}.\texttt{if}\ (n = 0)\ \texttt{then}\ 0\ \texttt{else}\ \overline{a}\langle n-1\rangle \mid \overline{a}\langle n-1\rangle$$

$$a : \forall i.\mathsf{serv}^{(2^{i+1}-1)}(\mathsf{Nat}[0,i])$$

Typing Rules for Simple Types (Reminder)

$$\frac{\Gamma \vdash a : \mathsf{ch}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P}{\Gamma \vdash !a(\tilde{v}).P}$$

$$\frac{\Gamma \vdash a : \mathsf{ch}(\tilde{T}) \qquad \Gamma \vdash \tilde{e} : \tilde{T}}{\Gamma \vdash \overline{a}\langle\tilde{e}\rangle}$$

Typing Rules for Work

$$\frac{}{\Gamma \vdash !a(\tilde{v}).P}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T})}{\Gamma \vdash !a(\tilde{v}).P}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \triangleleft K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \triangleleft K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P \triangleleft 0}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \lhd K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P \lhd 0}$$

$$\frac{}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \lhd K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P \lhd 0}$$

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T})}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \lhd K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P \lhd 0}$$

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T}) \qquad \Gamma \vdash \tilde{e} : \tilde{T}\{\tilde{J}/\tilde{i}\}}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle}$$

Typing Rules for Work

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T}) \qquad \Gamma, \tilde{v} : \tilde{T} \vdash P \triangleleft K \qquad \tilde{i} \text{ fresh}}{\Gamma \vdash !a(\tilde{v}).P \triangleleft 0}$$

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}^{K}(\tilde{T}) \qquad \Gamma \vdash \tilde{e} : \tilde{T}\{\tilde{J}/\tilde{i}\}}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle \triangleleft K\{\tilde{J}/\tilde{i}\}}$$

Methodology: Subject Reduction

If $\Gamma \vdash P \lhd K$ and $P \rightarrow^0 Q$ then $\Gamma \vdash Q \lhd K$

If $\Gamma \vdash P \lhd K$ and $P \rightarrow^1 Q$ then $\Gamma \vdash Q \lhd K - 1$

Methodology: Subject Reduction

If $\Gamma \vdash P \triangleleft K$ and $P \to^0 Q$ then $\Gamma \vdash Q \triangleleft K$

If $\Gamma \vdash P \triangleleft K$ and $P \to^1 Q$ then $\Gamma \vdash Q \triangleleft K - 1$

Theorem

If $\Gamma \vdash P \triangleleft K$ then K is a bound on the work of P

Types for Span

Extending the Previous Type System

We need some time information

Types for Span

Extending the Previous Type System

We need some time information

Syntax with Time Indications

$$T := \mathsf{Nat}[I, J] \mid \cdots \mid \mathsf{ch}_{I}(\tilde{T}) \mid \forall \tilde{i}.\mathsf{serv}_{I}^{K}(\tilde{T})$$

Types for Span

Extending the Previous Type System

We need some time information

Syntax with Time Indications

$T := \mathsf{Nat}[I, J] \mid \cdots \mid \mathsf{ch}_{I}(\tilde{T}) \mid \forall \tilde{i}.\mathsf{serv}_{I}^{K}(\tilde{T})$

I : time at which the channel is ready to communicate

Types for Span

Extending the Previous Type System

We need some time information

Syntax with Time Indications

$T := \mathsf{Nat}[I, J] \mid \cdots \mid \mathsf{ch}_{I}(\tilde{T}) \mid \forall \tilde{i}.\mathsf{serv}_{I}^{K}(\tilde{T})$

I : time at which the channel is ready to communicate

$$\frac{\langle \Gamma \rangle_{-1} \vdash P \lhd K}{\Gamma \vdash \mathtt{tick}.P \lhd K + 1}$$

$$\frac{\Gamma \vdash P \lhd K \qquad \Gamma \vdash Q \lhd K'}{\Gamma \vdash P \mid Q \lhd max(K, K')}$$

Rules for Servers

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}_{I}^{K}(\tilde{T}) \qquad \langle \Gamma \rangle_{-I}, \tilde{v} : \tilde{T} \vdash P \triangleleft K}{\Gamma \vdash !a(\tilde{v}).P \triangleleft I}$$

$$\frac{\Gamma \vdash a : \forall \tilde{i}.\mathtt{serv}_{I}^{K}(\tilde{T}) \qquad \langle \Gamma \rangle_{-I} \vdash \tilde{e} : \tilde{T}\{\tilde{J}/\tilde{i}\}}{\Gamma \vdash \overline{a}\langle \tilde{e} \rangle \triangleleft K\{\tilde{J}/\tilde{i}\} + I}$$

Subject Reduction

If $\Gamma \vdash P \triangleleft K$ and $P \Rightarrow Q$ then $\Gamma \vdash Q \triangleleft K$

If $\Gamma \vdash P \triangleleft K$ and $P \equiv n : P_1 \mid P_2$ then $K \geq n$

Subject Reduction

If $\Gamma \vdash P \triangleleft K$ and $P \Rightarrow Q$ then $\Gamma \vdash Q \triangleleft K$

If $\Gamma \vdash P \triangleleft K$ and $P \equiv n : P_1 \mid P_2$ then $K \geq n$

Theorem

If $\Gamma \vdash P \triangleleft K$ then K is a bound on the span of P

Simple Semaphore

Limitations of the span type system:

$$a().\mathtt{tick}.\overline{a}\langle\rangle \mid a().\mathtt{tick}.\overline{a}\langle\rangle \mid \cdots \mid a().\mathtt{tick}.\overline{a}\langle\rangle \mid \overline{a}\langle\rangle$$

Simple Semaphore

Limitations of the span type system:

$$a().\mathtt{tick}.\overline{a}\langle\rangle \mid a().\mathtt{tick}.\overline{a}\langle\rangle \mid \cdots \mid a().\mathtt{tick}.\overline{a}\langle\rangle \mid \overline{a}\langle\rangle$$

Span type system cannot count the number of similar parallel processes.
Also, it cannot give a "time" to a.

Usage Type System, Briefly

$$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})/U \mid \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T})/U$$

Usage Type System, Briefly

$$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})/U \mid \forall \tilde{i}.\mathsf{serv}^{K}(\tilde{T})/U$$

Intuitively, U described the behavior of the channel in a process independently from other channels.

Usage Type System, Briefly

$$T := \mathsf{Nat}[I,J] \mid \mathsf{ch}(\tilde{T})/U \mid \forall \tilde{i}.\mathsf{serv}^K(\tilde{T})/U$$

Intuitively, U described the behavior of the channel in a process independently from other channels.

$$U, V \approx 0 \mid (U \mid V) \mid \mathtt{In}^{t_o}_{t_c}.U \mid \mathtt{Out}^{t_o}_{t_c}.U$$

Usage Type System, Briefly

$$T := \mathsf{Nat}[I, J] \mid \mathsf{ch}(\tilde{T})/U \mid \forall \tilde{i}.\mathsf{serv}^K(\tilde{T})/U$$

Intuitively, U described the behavior of the channel in a process independently from other channels.

$$U, V \approx 0 \mid (U \mid V) \mid \mathtt{In}_{t_c}^{t_o}.U \mid \mathtt{Out}_{t_c}^{t_o}.U$$

We need usages adapted to span, joint work with Naoki Kobayashi.

Sum Up

Contributions

- Simple definition of Parallel Complexity
- Size-based type system for π-calculus
- Elegant proof method for complexity soundness

Typable Process for Span

Bitonic Sort with $O(log(n)^2)$ comparisons

Sum Up

Contributions

- Simple definition of Parallel Complexity
- Size-based type system for π-calculus
- Elegant proof method for complexity soundness

Typable Process for Span

Bitonic Sort with $O(log(n)^2)$ comparisons

Perspective

- Full Type Inference
- Analysis of Width
- Amortized Complexity

Thank you for your attention.