Deadlock and lock freedom in the linear π-calculus

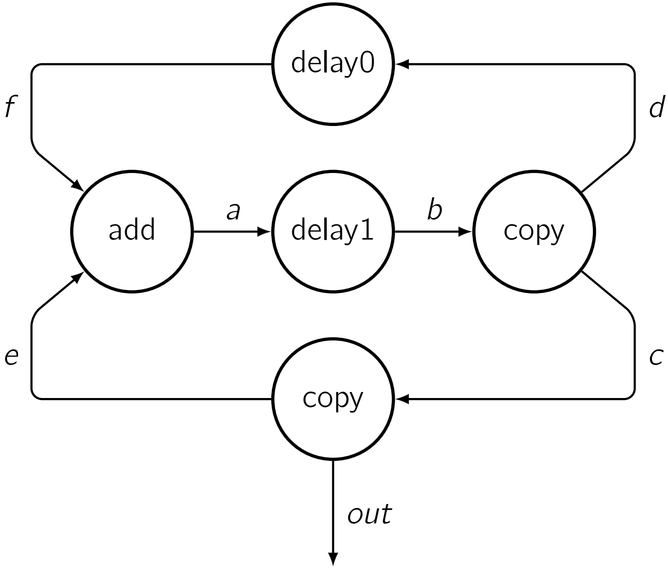
Luca Padovani

(with contributions from Tzu-Chun Chen, Luca Novara, Andrea Tosatto)

Dipartimento di Informatica – Università di Torino

Demo

Demo



Stages

1. linearity analysis
   ⇒ partition channels into **linear** and **non-linear**

2. protocol analysis (optional)
   ⇒ infer **communication structure**

3. deadlock analysis
   ⇒ no pending communications in **stable** states

4. lock analysis
   ⇒ pending communications in **all** states can be completed

Why the focus on linear channels?

- Kahn process networks [Kahn '74]
- $\sim 50\%$ channels are linear [Kobayashi, Pierce, Turner '99]
- binary sessions [Kobayashi '07, Demangeon and Honda '11, Dardha et al. '12]
- multiparty sessions [Padovani '13, Pérez et al. '14]

Outline

1. Introduction
2. Linearity analysis
3. Protocol analysis
4. Deadlock analysis
5. Lock analysis
6. Final remarks

Linearity analysis: how it works

$$^{\kappa_1,\kappa_2}[t]$$

`new a in { a!3 | a?x }`

- $\alpha_0 = \alpha_1 + \alpha_2$
- $^{\rho,\rho}[\mathtt{int}] = {}^{0,1}[\mathtt{int}] + {}^{1,0}[\mathtt{int}]$
- $\rho = 0 + 1$
- $\rho = 1 + 0$
- $\alpha_0 = {}^{1,1}[\mathtt{int}]$

type combination $\neq$ unification

Linearity analysis: how it works

$$^{\kappa_1,\kappa_2}[t]$$

$\alpha_1 = {}^{0,1}[\texttt{int}]$

`new a in { a!3 | a?x }`

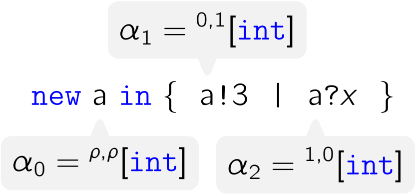
$\alpha_0 = {}^{\rho,\rho}[\texttt{int}]$ $\alpha_2 = {}^{1,0}[\texttt{int}]$

- $\alpha_0 = \alpha_1 + \alpha_2$
- ${}^{\rho,\rho}[\texttt{int}] = {}^{0,1}[\texttt{int}] + {}^{1,0}[\texttt{int}]$
- $\rho = 0 + 1$
- $\rho = 1 + 0$
- $\alpha_0 = {}^{1,1}[\texttt{int}]$

type combination $\neq$ unification

Linearity analysis: how it works

$$^{\kappa_1,\kappa_2}[t]$$



$\alpha_1 = {}^{0,1}[\texttt{int}]$

`new a in { a!3 | a?x }`

$\alpha_0 = {}^{\rho,\rho}[\texttt{int}]$

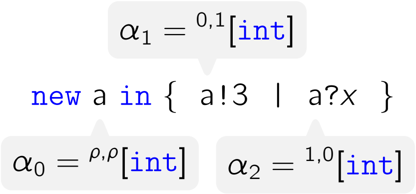
$\alpha_2 = {}^{1,0}[\texttt{int}]$

- $\alpha_0 = \alpha_1 + \alpha_2$
- ${}^{\rho,\rho}[\texttt{int}] = {}^{0,1}[\texttt{int}] + {}^{1,0}[\texttt{int}]$
- $\rho = 0 + 1$
- $\rho = 1 + 0$
- $\alpha_0 = {}^{1,1}[\texttt{int}]$

type combination $\neq$ unification

Linearity analysis: how it works

$$^{\kappa_1,\kappa_2}[t]$$



$\alpha_1 = {}^{0,1}[\texttt{int}]$

`new a in { a!3 | a?x }`

$\alpha_0 = {}^{\rho,\rho}[\texttt{int}]$

$\alpha_2 = {}^{1,0}[\texttt{int}]$

- $\alpha_0 = \alpha_1 + \alpha_2$
- ${}^{\rho,\rho}[\texttt{int}] = {}^{0,1}[\texttt{int}] + {}^{1,0}[\texttt{int}]$
- $\rho = 0 + 1$
- $\rho = 1 + 0$
- $\alpha_0 = {}^{1,1}[\texttt{int}]$

type combination $\neq$ unification

Linearity analysis: how it works

$$ {}^{\kappa_1,\kappa_2}[t] $$

$\alpha_1 = {}^{0,1}[\texttt{int}]$

$\texttt{new a in \{ a!3 | a?}x\texttt{ \}}$

$\alpha_0 = {}^{\rho,\rho}[\texttt{int}]$ $\alpha_2 = {}^{1,0}[\texttt{int}]$

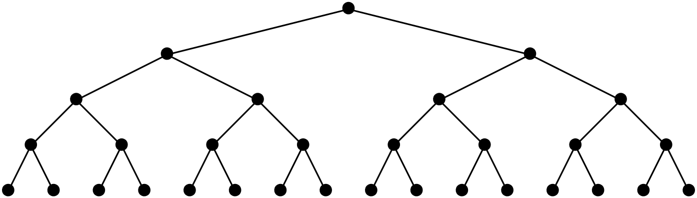
- $\alpha_0 = \alpha_1 + \alpha_2$
- ${}^{\rho,\rho}[\texttt{int}] = {}^{0,1}[\texttt{int}] + {}^{1,0}[\texttt{int}]$
- $\rho = 0 + 1$
- $\rho = 1 + 0$
- $\alpha_0 = {}^{1,1}[\texttt{int}]$

type combination $\neq$ unification

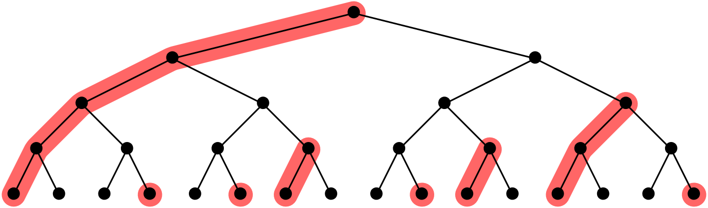
Demo: trees

```
  *case take? of
   { Leaf         ⇒ {}
   ; Node(c,l,r) ⇒ c!0 | take!l | skip!r }
|
  *case skip? of
   { Leaf         ⇒ {}
   ; Node(_,l,r) ⇒ skip!l | take!r }
|
  take!t | skip!t
```

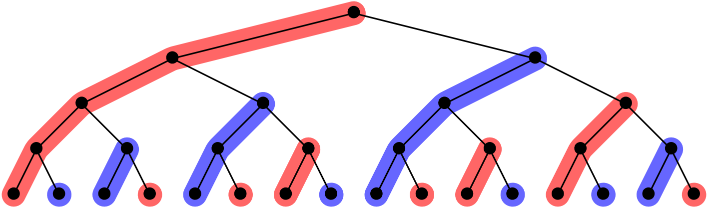
Channels used by `take` (red) and `skip` (blue)



Channels used by `take` (red) and `skip` (blue)



Channels used by `take` (red) and `skip` (blue)



Outline

1. Introduction
2. Linearity analysis
3. **Protocol analysis**
4. Deadlock analysis
5. Lock analysis
6. Final remarks

$s?(x).s!(x+1)$

binary
sessions

linear
π-calculus

session
types

linear
channel
types

`?int.!int...`

$s?(x).s!(x+1)$ $\qquad$ $s?(x,s').\texttt{new}\ s''\ \texttt{in}\ s'!(x+1,s'')$

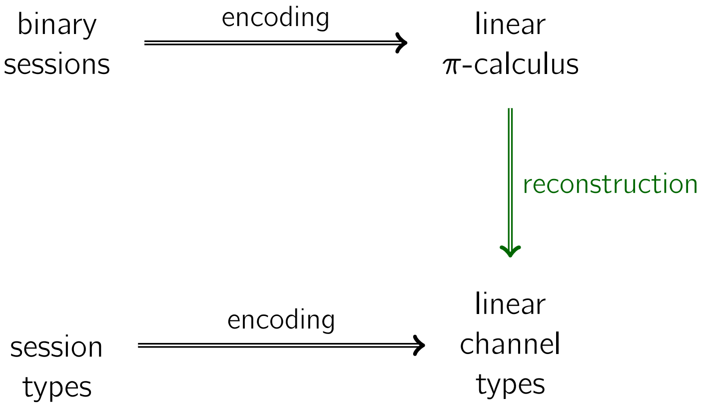
binary sessions $\xRightarrow{\text{encoding}}$ linear π-calculus

session types $\xRightarrow{\text{encoding}}$ linear channel types

$?\texttt{int}.!\texttt{int}\ldots$ $\qquad$ ${}^{1,0}[\texttt{int} \times {}^{0,1}[\texttt{int} \times \cdots]]$

$s?(x).s!(x+1)$

$s?(x,s').\texttt{new}\ s''\ \texttt{in}\ s'!(x+1,s'')$



binary sessions

encoding

linear π-calculus

reconstruction

session types

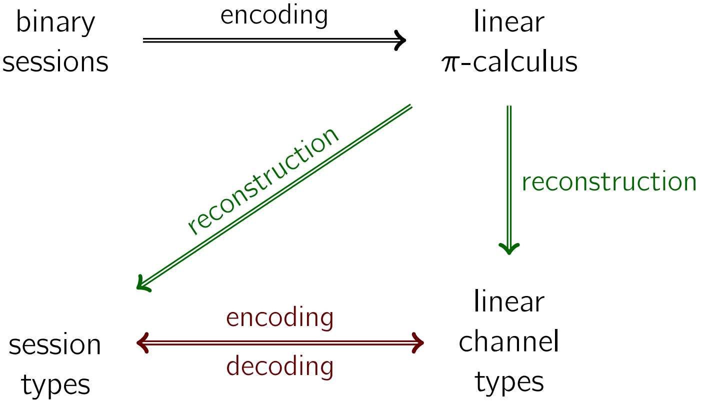
encoding

linear channel types

$?\texttt{int}.!\texttt{int}\ldots$

$^{1,0}[\texttt{int} \times {}^{0,1}[\texttt{int} \times \cdots]]$

$s?(x).s!(x+1)$ $\qquad$ $s?(x,s').\texttt{new}\ s''\ \texttt{in}\ s'!(x+1,s'')$



binary sessions

encoding

linear π-calculus

reconstruction

reconstruction

session types

encoding

decoding

linear channel types

$?\texttt{int}.!\texttt{int}\ldots$ $\qquad$ ${}^{1,0}[\texttt{int} \times {}^{0,1}[\texttt{int} \times \cdots]]$

Demo: math server

```
*server?s.
 case s? of
 { Quit    ⇒ {}
 ; Plus c1 ⇒ c1?(x,c2).
             c2?(y,c3).
             new c4 in { c3!(x + y, c4) | server!c4 }
 ; Eq c1   ⇒ c1?(x:Int,c2).
             c2?(y,c3).
             new c4 in { c3!(x = y, c4) | server!c4 }
 ; Neg c1  ⇒ c1?(x,c2).
             new c3 in { c2!(0 - x, c3) | server!c3 } }
```

Outline

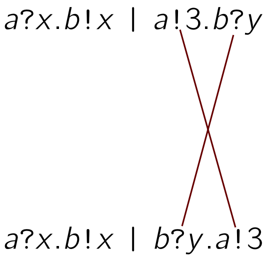
Different processes, same typing

$$a?x.b!x \mid a!3.b?y$$

Different processes, same typing

$$\mathtt{a} : {}^{1,1}[\mathtt{int}], \mathtt{b} : {}^{1,1}[\mathtt{int}] \vdash a\mathtt{?}x.b\mathtt{!}x \mid a\mathtt{!}3.b\mathtt{?}y$$

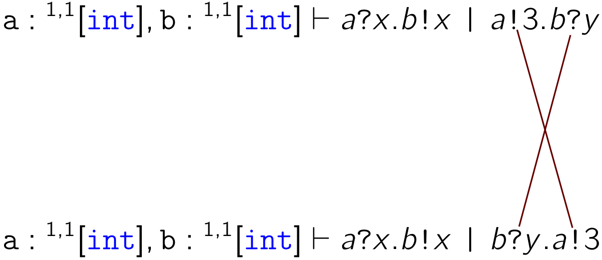
Different processes, same typing



$$\mathtt{a} : {}^{1,1}[\mathtt{int}], \mathtt{b} : {}^{1,1}[\mathtt{int}] \vdash a?x.b!x \mid a!3.b?y$$

$$a?x.b!x \mid b?y.a!3$$

Different processes, same typing



$$\mathtt{a} : {}^{1,1}[\mathtt{int}], \mathtt{b} : {}^{1,1}[\mathtt{int}] \vdash a\mathtt{?}x.b\mathtt{!}x \mid a\mathtt{!}3.b\mathtt{?}y$$

$$\mathtt{a} : {}^{1,1}[\mathtt{int}], \mathtt{b} : {}^{1,1}[\mathtt{int}] \vdash a\mathtt{?}x.b\mathtt{!}x \mid b\mathtt{?}y.a\mathtt{!}3$$

Basic strategy for deadlock analysis

❶ assign each linear channel a level $\in \mathbb{Z}$

$${}^{\kappa_1,\kappa_2}[t]^{h}$$

❷ make sure that channels are used in strict order

$$a\ \ ?x.b\ \ !x \mid b\ ?y.a\ \ !3$$

Basic strategy for deadlock analysis

1. assign each linear channel a level $\in \mathbb{Z}$

$$\kappa_1,\kappa_2[t]^h$$

2. make sure that channels are used in strict order

$$a^m?x.b^n!x \mid b^n?y.a^m!3$$

Basic strategy for deadlock analysis

❶ assign each linear channel a level $\in \mathbb{Z}$

$$^{\kappa_1,\kappa_2}[t]^{h}$$

❷ make sure that channels are used in strict order

$$\overbrace{a^{m}?x.b^{n}}^{<}!x \mid \overbrace{b^{n}?y.a^{m}}^{<}!3$$

A problem with recursive processes

```
*link?(x ,y ).
 x ?(z,a ).              -- x blocks y and a
 new b  in y !(z,b ).    -- y blocks a and b
 link!(a ,b )
```

A problem with recursive processes

```
*link?(x^0,y^1).
 x^0?(z,a ).                -- x blocks y and a
 new b  in y^1!(z,b ).      -- y blocks a and b
 link!(a ,b )
```

A problem with recursive processes

```
*link?(x⁰,y¹).
 x⁰?(z,a²).               -- x blocks y and a
 new b  in y¹!(z,b ).     -- y blocks a and b
 link!(a²,b )
```

A problem with recursive processes

```
*link?(x^0,y^1).
 x^0?(z,a^2).                -- x blocks y and a
 new b^3 in y^1!(z,b^3).     -- y blocks a and b
 link!(a^2,b^3)
```

A problem with recursive processes

```
*link?(x^0,y^1).
 x^0?(z,a^2).                   -- x blocks y and a
 new b^3 in y^1!(z,b^3).        -- y blocks a and b
 link!(a^2,b^3)
```

Problem

- the levels of `a` and `b` don't match those of `x` and `y`
- type error

A problem with recursive processes

```
*link?(x^0,y^1).
 x^0?(z,a^2).                 -- x blocks y and a
 new b^3 in y^1!(z,b^3).      -- y blocks a and b
 link!(a^2,b^3)
```

Problem

- ⊖ the levels of `a` and `b` don't match those of `x` and `y`
- ⊖ type error

Solution

- ⊕ the mismatch is OK as long as it is a translation
- ⊕ allow level polymorphism

Type reconstruction: how it works

```
*link?(x ,y ).
 x ?(z,a ).             --
 new b  in y !(z,b ).   --
 link!(a ,b )           --
```

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Type reconstruction: how it works

```
*link?(x^n,y^m).
 x^n?(z,a^h).              --
 new b^k in y^m!(z,b^k).   --
 link!(a^h,b^k)            --
```

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Type reconstruction: how it works

$\texttt{*link?(x}^{n}\texttt{,y}^{m}\texttt{).}$

$\quad\texttt{x}^{n}\texttt{?(z,a}^{h}\texttt{).}$ $\qquad\qquad$ -- $n < m \wedge n < h$

$\quad\texttt{new b}^{k}\texttt{ in y}^{m}\texttt{!(z,b}^{k}\texttt{).}$ $\qquad$ --

$\quad\texttt{link!(a}^{h}\texttt{,b}^{k}\texttt{)}$ $\qquad\qquad$ --

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Type reconstruction: how it works

$\texttt{*link?(x}^{n}\texttt{,y}^{m}\texttt{).}$

$\texttt{x}^{n}\texttt{?(z,a}^{h}\texttt{).}$ -- $n < m \wedge n < h$

$\texttt{new b}^{k}\texttt{ in y}^{m}\texttt{!(z,b}^{k}\texttt{).}$ -- $m < h \wedge m < k$

$\texttt{link!(a}^{h}\texttt{,b}^{k}\texttt{)}$ --

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Type reconstruction: how it works

$$\begin{array}{ll}
\texttt{*link?(x}^{n}\texttt{,y}^{m}\texttt{).} & \\
\quad\texttt{x}^{n}\texttt{?(z,a}^{h}\texttt{).} & \texttt{--}\ n < m \wedge n < h \\
\quad\texttt{new b}^{k}\texttt{ in y}^{m}\texttt{!(z,b}^{k}\texttt{).} & \texttt{--}\ m < h \wedge m < k \\
\quad\texttt{link!(a}^{h}\texttt{,b}^{k}\texttt{)} & \texttt{--}\ h = n + t \wedge k = m + t
\end{array}$$

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Type reconstruction: how it works

$\texttt{*link?(x}^{n}\texttt{,y}^{m}\texttt{).}$

$\texttt{ x}^{n}\texttt{?(z,a}^{h}\texttt{).}$ -- $n < m \land n < h$

$\texttt{ new b}^{k}\texttt{ in y}^{m}\texttt{!(z,b}^{k}\texttt{).}$ -- $m < h \land m < k$

$\texttt{ link!(a}^{h}\texttt{,b}^{k}\texttt{)}$ -- $h = n + t \land k = m + t$

1. perform linearity analysis
2. put integer variables in place of (unknown) levels
3. compute constraints
4. use ILP solver

Outline

Deadlocks vs locks

$$\texttt{new}\ \underset{{}^{1,1}[\texttt{int}]^{n}}{\mathsf{a}}\ \texttt{in}\ \{\ \overset{{}^{0,1}[\texttt{int}]^{n}}{\mathsf{a}}\,!\,3 \mid c\,!\,\underset{{}^{1,0}[\texttt{int}]^{n}}{\mathsf{a}} \mid *c?x.c\,!\,x\ \}$$

Strategy for lock analysis

❶ assign each linear channel a finite number $k \in \mathbb{N}$ of tickets

$$^{\kappa_1,\kappa_2}[t]^{h}_{k}$$

❷ each time a channel travels, one ticket is consumed

❸ channels with no tickets cannot travel

Lock analysis and type reconstruction

$\texttt{*link?(x}^{0}\texttt{,y}^{1}\texttt{).}$
$\quad\texttt{x}^{0}\texttt{?(z,a}^{2}\texttt{).}$
$\quad\texttt{new b}^{3}\texttt{ in y}^{1}\texttt{!(z,b}^{3}\texttt{).}$
$\quad\texttt{link!(a}^{2}\texttt{,b}^{3}\texttt{)}$

① perform linearity analysis

② put natural variables in place of (unknown) levels and tickets

③ compute constraints

④ use ILP solver

Lock analysis and type reconstruction

$$\begin{array}{l} *\texttt{link?}(\texttt{x}^{0}_{0},\texttt{y}^{1}_{0}). \\ \quad \texttt{x}^{0}_{0}\texttt{?}(\texttt{z},\texttt{a}^{2}). \\ \quad \texttt{new b}^{3}\ \texttt{in y}^{1}_{0}\texttt{!}(\texttt{z},\texttt{b}^{3}). \\ \quad \texttt{link!}(\texttt{a}^{2},\texttt{b}^{3}) \end{array}$$

1. perform linearity analysis
2. put natural variables in place of (unknown) levels and tickets
3. compute constraints
4. use ILP solver

Lock analysis and type reconstruction

$*\texttt{link?(x}^{0}_{0}\texttt{,y}^{1}_{0}\texttt{).}$

$\quad\texttt{x}^{0}_{0}\texttt{?(z,a}^{2}_{1}\texttt{).}$

$\quad\texttt{new b}^{3}\texttt{ in y}^{1}_{0}\texttt{!(z,b}^{3}\texttt{).}$

$\quad\texttt{link!(a}^{2}_{1}\texttt{,b}^{3}\texttt{)}$

1. perform linearity analysis
2. put natural variables in place of (unknown) levels and tickets
3. compute constraints
4. use ILP solver

Lock analysis and type reconstruction

$$\begin{array}{l}
*\mathtt{link?}(\mathtt{x}_0^0,\mathtt{y}_0^1). \\
\quad \mathtt{x}_0^0\mathtt{?}(\mathtt{z},\mathtt{a}_1^2). \\
\quad \mathtt{new}\ \mathtt{b}_2^3\ \mathtt{in}\ \mathtt{y}_0^1\mathtt{!}(\mathtt{z},\mathtt{b}_1^3). \\
\quad \mathtt{link!}(\mathtt{a}_1^2,\mathtt{b}_1^3)
\end{array}$$

1. perform linearity analysis
2. put natural variables in place of (unknown) levels and tickets
3. compute constraints
4. use ILP solver

Outline

1. Introduction
2. Linearity analysis
3. Protocol analysis
4. Deadlock analysis
5. Lock analysis
6. Final remarks

Related work

No levels

- sessions (without interleaving)
- session types as linear logic propositions

Discrete levels

- Kobayashi [2002, 2006, . . .]
- Bettini et al. [2008]
- Padovani, Vasconcelos, Vieira [2014]
- . . .

Dense levels

- Giachino, Kobayashi, Laneve [2014]

Summary of products

- 🗎 Padovani, **Deadlock and Lock Freedom in the Linear π-Calculus** (LICS'14)

- 🗎 Padovani, **Type Reconstruction for the Linear π-Calculus with Composite and Equi-Recursive Types** (FoSSaCS'14)

- 🗎 Padovani, Chen, Tosatto, **Type Reconstruction Algorithms for Deadlock-Free and Lock-Free Linear π-Calculi** (submitted)

- 🗎 Padovani and Novara, **Types and Effects for Deadlock-Free Higher-Order Programs** (submitted)

- 🗎 Padovani and Tosatto, **Hypha** (available at `http://di.unito.it/hypha`)