Semantic subtyping for the π-calculus

Giuseppe Castagna, Rocco de Nicola and Daniele Varacca

Paris, Firenze, London

Cambridge, May 13th 2005

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Road Map

- Semantic subtyping
  - Syntactic and semantic subtyping
  - Types and subtyping for π
  - The intuitive semantic of types
  - The construction of a model
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Subtyping and subsumption

Relation on types $t \leq t'$

Usually defined structurally

$$\frac{t \leq t'}{broccoli(t) \geq broccoli(t')}$$

Subtyping and subsumption

Relation on types $t \leq t'$

Usually defined structurally

$$\frac{t \leq t'}{broccoli(t) \geq broccoli(t')}$$

$$\frac{}{broccoli(t) \leq zucchini(t)}$$

Subtyping and subsumption

Relation on types $t \leq t'$

Usually defined structurally

$$\frac{t \leq t'}{broccoli(t) \geq broccoli(t')}$$

$$\frac{}{broccoli(t) \leq zucchini(t)}$$

Subsumption rule

$$\frac{\Gamma \vdash e : t \quad t \leq t'}{\Gamma \vdash e : t'}$$

Semantic subtyping

The semantic of a type is the set of its values

$$[\![t]\!] = \{v \mid \vdash v : t\}$$

Subtyping is just set inclusion

$$t \leq t' \stackrel{def}{\Longleftrightarrow} [\![t]\!] \subseteq [\![t']\!]$$

Core of $\mathbb{C}$Duce, a functional programming language for XML manipulation

ℂDuce in one slide

Developped by V. Benzaken, G. Castagna, A. Frisch

- Based on Hosoya and Pierce's XDuce
- Native XML types
- Boolean combinations of types, function types, recursive types
- Semantic subtyping
- Pattern matching and typecase (based on subtyping)
- Efficiently implemented, and USED! Already available in Debian testing

Foundational paper: *Semantic subtyping*, by Frisch, Castagna, Benzaken, LICS 2002

Circularity of subtyping

- Need semantics of types to define the subtyping relation
- need typed values to define the semantics of types
- need subtyping relation to type values, using subsumption rule

A circle!

Breaking the circle

- Define some bootstrap semantics of types
- use it to define some subtyping relation
- use subtyping relation to type values, using subsumption rule
- use values to define another semantics of types
- use this semantics to define another subtyping relation
- . . . an so on

If we are lucky, we reach a fixed point

Closing the circle

- define some bootstrap semantics of types
- use it to define some subtyping relation
- use subtyping relation to type values, using subsumption rule
- use values to define another semantics of types
- use this semantics to define another subtyping relation
- . . . an so on

If we are lucky, we reach a fixed point

Indeed we do, after only one iteration

Want to know more?

Beppe Castagna is coming soon!

Typing channels in π

- Channels are typed according to the value they transport

$$x : ch(t)$$

- Processes are well typed when communication respects the typing
- Well typed processes don’t get stuck

Input/output channels

A more refined type system

- $x : ch^{-}(t)$
  The channel can be used only in output
- $x : ch^{+}(t)$
  The channel can be used only in input

Subtyping for π

Input types $ch^+(t)$:
what I expect to read from a channel
Covariant: channels carrying less don't harm

$$\frac{t \leq t'}{ch^+(t) \leq ch^+(t')}$$

Subtyping for π

Input types $ch^+(t)$:
what I expect to read from a channel
Covariant: channels carrying less don't harm

$$\frac{t \leq t'}{ch^+(t) \leq ch^+(t')}$$

Output types $ch^-(t)$:
what I expect I can write to a channel
Contravariant: channels carrying more don't harm

$$\frac{t \leq t'}{ch^-(t) \geq ch^-(t')}$$

Boolean combinators

What if we want unions, intersections, negations of types?

$$ch^+(t_1) \vee ch^+(t_2) \stackrel{?}{=} ch^+(t_1 \vee t_2)$$

$$ch^+(t_1) \wedge ch^+(t_2) \stackrel{?}{=} ch^+(t_1 \wedge t_2)$$

Our type system for π

$$\begin{array}{rrcl} \textit{Types} & t & ::= & b \mid ch(t) \mid ch^{+}(t) \mid ch^{-}(t) \\ & & \mid & \mathbf{0} \mid \mathbf{1} \mid \neg t \mid t \vee t \mid t \wedge t \end{array}$$

Our type system for π

$$
\begin{array}{rrcl}
\textit{Types} & t & ::= & b \mid ch(t) \mid ch^{+}(t) \mid ch^{-}(t) \\
& & \mid & \mathbf{0} \mid \mathbf{1} \mid \neg t \mid t \vee t \mid t \wedge t \\
& & \mid & t \times t \mid t \to t \\
& & \mid & X \mid \mathbf{rec}\, X.t
\end{array}
$$

Intuitive semantics

A channel is like a box with a particular shape
The box can contain only objects that fit that shape

$[\![ch(t)]\!] = \{\text{channels that can contain objects of type } t\}$

Intuitive semantics

Assumption: channels can only be distinguished by their shape

Channels are their shape

$$[\![ch(t)]\!] = \{\text{shape for objects of type } t\}$$

Intuitive semantics

Assumption: channels can only be distinguished by their shape

Channels are their shape

$$[\![ch(t)]\!] = \{\text{set of objects of type } t\}$$

Intuitive semantics

Assumption: channels can only be distinguished by their shape

Channels are their shape

$$[\![ch(t)]\!] = \{[\![t]\!]\}$$

Invariance of channel types

Input types

Input/Output types express what can be safely done
If I am ready to read t on c, it is safe even if c carries less (gives me less)

$$[\![ch^+(t)]\!] = \{[\![t']\!] \mid t' \leq t\}$$

Covariance of input types

Output types

If I am ready to write t on c, it is safe even if c carries more (accepts more)

$$[\![ch^-(t)]\!] = \{[\![t']\!] \mid t' \geq t\}$$

Contravariance of output types

A bootstrap semantics of types

We want a set $\mathscr{D}$ such that every type is interpreted as a subset of $\mathscr{D}$
Booleans interpreted as corresponding set-theoretic operations

Problem: denotations should be elements of $\mathscr{D}$

$$[\![ch^-(t)]\!] = \{[\![t']\!] \mid t' \geq t\}$$

I.e. subsets of $\mathscr{D}$ should be elements of $\mathscr{D}$
Cantor is not happy

A bootstrap semantics of types

Idea: not all subsets of $\mathscr{D}$ are denoted by a type
We can thus construct a set $\mathscr{D}$, and an interpretation function $[\![-]\!]$ such that

- $[\![t]\!] \subseteq \mathscr{D}$, $[\![\mathbf{1}]\!] = \mathscr{D}$, $[\![\mathbf{0}]\!] = \varnothing$
- $[\![ch(t)]\!] \cup [\![b]\!] = \varnothing$
- $[\![\neg t]\!] = \mathscr{D} \setminus [\![t]\!]$
- $[\![t_1 \vee t_2]\!] = [\![t_1]\!] \cup [\![t_2]\!]$, $[\![t_1 \wedge t_2]\!] = [\![t_1]\!] \cap [\![t_2]\!]$
- $[\![ch^+(t)]\!] \sim \{[\![t']\!] \mid [\![t]\!]' \subseteq [\![t]\!]\}$
- $[\![ch^-(t)]\!] \sim \{[\![t']\!] \mid [\![t']\!] \supseteq [\![t]\!]\}$

A bootstrap semantics of types

We build such a model in steps

For the nth step, a model of channel types with n nestings

The set $\mathscr{D}$ is obtained at the limit

Important: not applicable to recursive types

Subtyping

$$t \leq t' \stackrel{def}{\Longleftrightarrow} [\![t]\!] \leq [\![t']\!]$$

Some induced equations

$$ch^-(t) \wedge ch^+(t) = ch(t)$$

$$ch^+(t) \wedge ch^+(t') = ch^+(t \wedge t')$$

$$ch^+(t) \vee ch^+(t') \leq ch^+(t \vee t')$$

Road Map

- Semantic subtyping
  - Syntactic and semantic subtyping
  - Types and subtyping for π
  - The intuitive semantic of types
  - The construction of a model
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Road Map

- Semantic subtyping
- Decidability
  - Disjunctive normal form
  - The role of atoms
- The $\mathbb{C}\pi$ calculus
- Extensions

Decidability of subtyping

It is enough to decide emptiness:

$$t \leq t' \iff t \wedge \neg t' = \mathbf{0}$$

Decidability of subtyping

Put the type in disjunctive normal form
A disjunction is empty is all the summands are empty
A summand:

$$\bigwedge_{i \in P} t_i \wedge \bigwedge_{j \in N} \neg t'_j = \mathbf{0}?$$

Equivalently

$$\bigwedge_{i \in P} t_i \leq \bigvee_{j \in N} t'_j \ ?$$

Decidability of subtyping

Assume for basic type is decidable
Mixed basic and channel is easy
For channel types it reduces to

$$\bigwedge_{i\in I} ch^{+}(t_1^i) \wedge \bigwedge_{j\in J} ch^{-}(t_2^j) \leq \bigvee_{h\in H} ch^{+}(t_3^h) \vee \bigvee_{k\in K} ch^{-}(t_4^k)$$

Decidability of subtyping

Assume for basic type is decidable
Mixed basic and channel is easy
For channel types it reduces to

$$ch^+(t_1) \wedge ch^-(t_2) \leq \bigvee_{h \in H} ch^+(t_3^h) \vee \bigvee_{k \in K} ch^-(t_4^k)$$

Atoms

The condition to check involves atoms
Types with a singleton interpretation

Decide whether a type is an atom
Decide whether a type is finite

Example of subtyping

Three constants $\mathtt{err}_1, \mathtt{err}_2, \mathtt{exc}$

$$
\begin{aligned}
t_2 &= \mathtt{int} \\
t_1 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2 \vee \mathtt{exc} \\
t_3 &= t_2 \vee \mathtt{exc} \\
t_4 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2
\end{aligned}
$$

$$ch^+(t_1) \wedge ch^-(t_2) \not\leq ch^+(t_3) \vee ch^-(t_4)$$

Example of subtyping

Three constants $\mathtt{err}_1, \mathtt{err}_2, \mathtt{exc}$

$$
\begin{aligned}
t_2 &= \mathtt{int} \\
t_1 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2 \vee \mathtt{exc} \\
t_3 &= t_2 \vee \mathtt{exc} \\
t_4 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2
\end{aligned}
$$

$$ch^+(t_1) \wedge ch^-(t_2) \leq ch^+(t_3) \vee ch^-(t_4)$$

if $\mathtt{err}_1 = \mathtt{err}_2$

Example of subtyping

Three constants $\mathtt{err}_1, \mathtt{err}_2, \mathtt{exc}$

$$
\begin{aligned}
t_2 &= \mathtt{int} \\
t_1 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2 \vee \mathtt{exc} \\
t_3 &= t_2 \vee \mathtt{exc} \\
t_4 &= t_2 \vee \mathtt{err}_1 \vee \mathtt{err}_2
\end{aligned}
$$

it depends whether $\mathtt{err}_1 \vee \mathtt{err}_2$ is an atom

Relevance of atomic types

Atomic types important also in presence of polymorphism
Hosoya, Frisch, Castagna *Parametric Polymorphism for XML*, POPL 2005

Atomic types are tricky!

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
  - Patterns
  - The language
  - Typing
  - Operational semantics
- Extensions

Patterns

$$
\begin{array}{rcll}
p & ::= & x & \text{capture variable} \\
 & | & p \wedge t & \text{conjunction} \\
 & | & p_1 | \, p_2 & \text{alternative}
\end{array}
$$

in $p_1 | p_2$ we have $Var(p_1) = Var(p_2)$

Pattern matching

Matching an element of the model of the types

$$
\begin{array}{lcll}
d/x & = & \{x \mapsto d\} & \\
d/p \wedge t & = & d/p & \text{if } d \in [\![t]\!] \\
d/p_1|p_2 & = & d/p_1 & \text{if } d/p_1 \text{ does not fail} \\
d/p_1|p_2 & = & d/p_2 & \text{if } d/p_1 \text{ fails}
\end{array}
$$

$\mathbb{C}\pi$ calculus

Channels	α	$::=$	x	variables
		$\mid$	c^t	channel constants
Messages	M	$::=$	n	constant
		$\mid$	α	channel
Processes	P	$::=$	$\overline{\alpha}M$	output
		$\mid$	$\sum_{i\in I}\alpha(p_i).P_i$	patterned input
		$\mid$	$P_1 \| P_2$	parallel
		$\mid$	$(\nu c^t)P$	restriction
		$\mid$	$!P$	replication

Distinctive features

- Asynchrony
- Pattern Matching
- Distinction between variables and constants (as in Nielsen & Engberg ECCS)
- Channel constants are sorted

Typing Messages

$(t \leq t' \text{ iff } [\![t]\!] \leq [\![t']\!])$

$$\frac{}{\Gamma \vdash n : b_n} \text{(const)} \qquad \frac{}{\Gamma \vdash c^t : ch(t)} \text{(chan)}$$

$$\frac{}{\Gamma \vdash x : \Gamma(x)} \text{(var)} \qquad \frac{\Gamma \vdash M : s \leq t}{\Gamma \vdash M : t} \text{(subs)}$$

The model of values

Values are constants (basic and channels)

New semantics of types

$$[\![t]\!] = \{v \mid \vdash v : t\}$$

It induces the same subtyping relation
The circle is closed

It allows the pattern matching on values

Typing processes

$$\frac{\Gamma \vdash P}{\Gamma \vdash (\nu c^t)P}\ \text{(new)} \qquad \frac{\Gamma \vdash P}{\Gamma \vdash !P}\ \text{(repl)}$$

$$\frac{\Gamma \vdash P_1 \quad \Gamma \vdash P_2}{\Gamma \vdash P_1 \| P_2}\ \text{(para)}$$

$$t \leq \bigvee_{i \in I} \wr p_i \int \qquad \wr p_i \int \wedge t \neq \mathbf{0} \qquad \frac{\Gamma \vdash \alpha : ch^+(t) \quad \Gamma, t/p_i \vdash P_i}{\Gamma \vdash \sum_{i \in I} \alpha(p_i).P_i}\ \text{(input)}$$

$$\frac{\Gamma \vdash M : t \quad \Gamma \vdash \alpha : ch^-(t)}{\Gamma \vdash \overline{\alpha} M}\ \text{(output)}$$

The operational semantics

$$\overline{c^t}v \parallel \sum_{i \in I} c^t(p_i).P_i \longrightarrow P_j[v/p_j]$$

"Call by value"

Subject reduction

Decidability of typing

Decidability of subtyping does not entail decidability of typing!
Need to find principal types

This possible!

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions

Road Map

- Semantic subtyping
- Decidability
- The $\mathbb{C}\pi$ calculus
- Extensions
  - Product and arrow types
  - Recursive types

Products

Type system and language can be extended with product types
A polyadic version

The subtyping is still decidable

Recursive types

Only a limited form of recursive type:

$$t = \mathtt{int} \vee (ch(t) \wedge ch(\mathtt{int}))$$

Question: is $t = \mathtt{int}$ or not?

Both lead to contradiction

Recursive types

A recursion variable cannot occur inside a channel type

We can still type a channel carrying itself using $ch(\mathbf{1})$

We can express the type of list

The model is much more complicated ($\mathbb{C}$Duce)

Decidability? May be yes...

Arrow types

Arrow types: passing higher order functions (ℂDuce values)

Weak version: passing ℂDuce functions that cannot act on channels

Strong version: ℂDuce functions using channels

Decidability? Undecidability?

Local $\mathbb{C}\pi$

- received channels cannot be used in input
- known to be expressive enough (encodes λ)
- the type $ch^{+}()$ is not needed
- full recursion, decidability with arrow types

Future Work

- behavioural equivalences
- implementing local $\mathbb{C}\pi$
- encoding $\mathbb{C}$Duce in $\mathbb{C}\pi$ (ongoing)

Conclusion

- π-calculus with pattern matching
- a very rich type system
- advantages of semantic subtyping
- integration with ℂDuce, a functional programming language for XML documents

Any questions?

Secret Slides

Examples

A server

$$\alpha(x).!(x(y \wedge t_1).P_1 + x(y \wedge t_2).P_2)$$

$$\alpha : ch^+(ch^+(t_1 \vee t_2))$$

Examples

A server

$$\alpha(x).!(x(y \wedge t_1).P_1 + x(y \wedge t_2).P_2)$$

$\alpha : ch^+(ch^+(t_1 \vee t_2))$

A more efficient server

$$\alpha(x \wedge ch(t_1)).!x(y).P_1 + \alpha(x \wedge ch(t_2)).!x(y).P_2$$

$\alpha : ch^+(ch^+(t_1) \vee ch^+(t_2))$

Examples

A $\mathbb{C}$Duce server

$$fun^{s\to t}(x).arg^{s}(y).\overline{result}^{t}(x(y))$$

A more liberal one

$$\begin{aligned} & compute^{((s\to t)\times s)\vee(s\times(s\to t))}(x, y\wedge s).\overline{result}^{t}(x(y)) \\ + \; & compute^{((s\to t)\times s)\vee(s\times(s\to t))}(x\wedge s, y).\overline{result}^{t}(y(x)) \end{aligned}$$

Examples

A $\mathbb{C}\pi$ function

$$\texttt{fun}\ assoc(s:\texttt{string}, l:\texttt{a\_list}):ch(\texttt{int}) = \texttt{match}\ l\ \texttt{with}\ \texttt{nil} \rightarrow \texttt{fail} \mid ((k,c),t) \rightarrow \texttt{if}\ k = s\ \texttt{then}\ c\ \texttt{else}\ assoc(s,t)$$

A $\mathbb{C}\pi$ process

$$announce^{[\mathsf{m\_list}\times\mathsf{a\_list}\times\mathsf{retr}]}(marks,mails,getch).\ (\nu c^{\mathsf{m\_list}})\ \overline{c}(marks)\ \mid\ !(\ c(\ ((n,m),rest)\ ).\ (\ \overline{\overline{getch}(n,mails)}(m)\ \mid\ \overline{c}(rest)\ +\ c(\ \texttt{nil}).0\ )$$

$$\mathsf{m\_list} = ((\texttt{string}\times\texttt{int})\times\mathsf{m\_list})\vee\texttt{nil}$$

$$\mathsf{a\_list} = ((\texttt{string}\times ch(\texttt{int}))\times\mathsf{a\_list})\vee\texttt{nil}$$

$$\mathsf{retr} = \texttt{string}\times\mathsf{a\_list}\rightarrow ch(\texttt{int})$$

Examples

$\overline{announce}(\ (\ (\texttt{"Alice"},6),((\texttt{"Bob"},8),\texttt{nil}))\ ,$
$(\ (\texttt{"Bob"},c_B),((\texttt{"Alice"},c_A),\texttt{nil}))\ ,$
$assoc\ \ )$

Will reduce to process: $\overline{c_A}(6)\ \mid\ \overline{c_B}(8)$.

Local $\mathbb{C}\pi$ calculus

Channels	α	$::=$	x	variables
		$\mid$	c^t	typed channel (box
Messages	M	$::=$	n	constant
		$\mid$	α	channel
Processes	P	$::=$	$\overline{\alpha}M$	output
		$\mid$	$\sum_{i\in I} c^t(p_i).P_i$	patterned input
		$\mid$	$P_1 \| P_2$	parallel
		$\mid$	$(\nu c^t)P$	restriction
		$\mid$	$!P$	replication

Typing Messages in local $\mathbb{C}\pi$

$$\frac{}{\Gamma \vdash n : b_n}\ (\text{const}) \qquad \frac{}{\Gamma \vdash x : \Gamma(x)}\ (\text{var})$$

$$\frac{t \not\geq t_i,\ 1 \leq i \leq k}{\Gamma \vdash c^t : ch^-(t) \wedge \neg ch^-(t_1) \wedge \ldots \wedge \neg ch^-(t_k)}\ (\text{chan})$$

$$\frac{\Gamma \vdash M : s \leq t}{\Gamma \vdash M : t}\ (\text{subs})$$