Semantic subtyping for the \( \pi \)-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 $\pi$
  - 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

$$t \leq t' \quad \Rightarrow \quad broccoli(t) \geq broccoli(t')$$
Subtyping and subsumption

Relation on types $t \leq t'$

Usually defined structurally

\[
\begin{align*}
  t \leq t' & \quad \Rightarrow \\
  broccoli(t) \geq broccoli(t') & \\
  broccoli(t) \leq zucchini(t)
\end{align*}
\]
Subtyping and subsumption

Relation on types $t \leq t'$

Usually defined structurally

\[ t \leq t' \]

\[ \text{broccoli}(t) \geq \text{broccoli}(t') \]

\[ \text{broccoli}(t) \leq \text{zucchini}(t) \]

Subsumption rule

\[ \Gamma \vdash e : t \quad t \leq t' \]

\[ \Gamma \vdash e : t' \]
Semantic subtyping

The semantic of a type is the set of its values

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

Subtyping is just set inclusion

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

Core of CDuce, a functional programming language for XML manipulation
CDuce in one slide

Developed 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 \( \pi \)

- 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 $\pi$

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 $\pi$

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?

\[ ch^+(t_1) \lor ch^+(t_2) = ch^+(t_1 \lor t_2) \]

\[ ch^+(t_1) \land ch^+(t_2) = ch^+(t_1 \land t_2) \]
Our type system for $\pi$

Types $t ::=$ $b$ | $ch(t)$ | $ch^+(t)$ | $ch^-(t)$
  | 0   | 1   | $\neg t$ | $t \lor t$ | $t \land t$
Our type system for $\pi$

$$Types \: t \::= \: b \mid ch(t) \mid ch^+(t) \mid ch^-(t) \mid 0 \mid 1 \mid \neg t \mid t \lor t \mid t \land t \mid t \times t \mid t \rightarrow t \mid X \mid rec \: X \cdot t$$
Intuitive semantics

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

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

\[
\llbracket ch(t) \rrbracket = \{ \text{set of objects of type } t \}
\]
Intuitive semantics

Assumption: channels can only be distinguished by their shape

Channels are their shape

\[
[\text{ch}(t)] = \{[t]\}
\]

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).

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

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)

$$\llbracket ch^{-}(t) \rrbracket = \{ \llbracket t' \rrbracket | t' \geq t \}$$

Contravariance of output types
A bootstrap semantics of types

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

Problem: denotations should be elements of $D$

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

I.e. subsets of $D$ should be elements of $D$
Cantor is not happy
A bootstrap semantics of types

Idea: not all subsets of $\mathcal{D}$ are denoted by a type
We can thus construct a set $\mathcal{D}$, and an interpretation function $\llbracket \cdot \rrbracket$ such that

- $\llbracket t \rrbracket \subseteq \mathcal{D}$, $\llbracket 1 \rrbracket = \mathcal{D}$, $\llbracket 0 \rrbracket = \emptyset$
- $\llbracket \text{ch}(t) \rrbracket \cup \llbracket b \rrbracket = \emptyset$
- $\llbracket \neg t \rrbracket = \mathcal{D} \setminus \llbracket t \rrbracket$
- $\llbracket t_1 \lor t_2 \rrbracket = \llbracket t_1 \rrbracket \cup \llbracket t_2 \rrbracket$, $\llbracket t_1 \land t_2 \rrbracket = \llbracket t_1 \rrbracket \cap \llbracket t_2 \rrbracket$
- $\llbracket \text{ch}^+(t) \rrbracket \sim \{ \llbracket t' \rrbracket \mid \llbracket t' \rrbracket \subseteq \llbracket t \rrbracket \}$
- $\llbracket \text{ch}^-(t) \rrbracket \sim \{ \llbracket t' \rrbracket \mid \llbracket t' \rrbracket \supseteq \llbracket t \rrbracket \}$

Semantic subtyping for the $\pi$-calculus – p.20
A bootstrap semantics of types

We build such a model in steps
For the $n$th step, a model of channel types with $n$ nestings
The set $\mathcal{D}$ is obtained at the limit
Important: not applicable to recursive types
Subtyping

\[ t \leq t' \overset{\text{def}}{\iff} \llbracket t \rrbracket \leq \llbracket t' \rrbracket \]

Some induced equations

\[ ch^-(t) \land ch^+(t) = ch(t) \]

\[ ch^+(t) \land ch^+(t') = ch^+(t \land t') \]

\[ ch^+(t) \lor ch^+(t') \leq ch^+(t \lor t') \]
Road Map

- Semantic subtyping
  - Syntactic and semantic subtyping
  - Types and subtyping for $\pi$
  - The intuitive semantic of types
  - The construction of a model
- Decidability
- The $\mathcal{C}\pi$ 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:

\[ t \leq t' \iff t \land \neg t' = 0 \]
Decidability of subtyping

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

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

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

\[ \bigwedge_{i \in I} ch^+(t_1^i) \land \bigwedge_{j \in J} ch^-(t_2^j) \leq \bigvee_{h \in H} ch^+(t_3^h) \lor \bigvee_{k \in K} ch^-(t_4^k) \]
Decidability of subtyping

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

\[
ch^+(t_1) \land ch^-(t_2) \leq \bigvee_{h \in H} ch^+(t_3^h) \lor \bigvee_{k \in K} ch^-(t_4^k)
\]
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 $\text{err}_1, \text{err}_2, \text{exc}$

$t_2 = \text{int}$

$t_1 = t_2 \lor \text{err}_1 \lor \text{err}_2 \lor \text{exc}$

$t_3 = t_2 \lor \text{exc}$

$t_4 = t_2 \lor \text{err}_1 \lor \text{err}_2$

\[
\begin{align*}
ch^+(t_1) \land ch^-(t_2) & \not\subseteq ch^+(t_3) \lor ch^-(t_4)
\end{align*}
\]
Example of subtyping

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

\[
\begin{align*}
    t_2 &= \text{int} \\
    t_1 &= t_2 \lor \text{err}_1 \lor \text{err}_2 \lor \text{exc} \\
    t_3 &= t_2 \lor \text{exc} \\
    t_4 &= t_2 \lor \text{err}_1 \lor \text{err}_2
\end{align*}
\]

\[
ch^+(t_1) \land ch^-(t_2) \leq ch^+(t_3) \lor ch^-(t_4)
\]

if $\text{err}_1 = \text{err}_2$
Example of subtyping

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

\[
\begin{align*}
t_2 & = \text{int} \\
t_1 & = t_2 \lor \text{err}_1 \lor \text{err}_2 \lor \text{exc} \\
t_3 & = t_2 \lor \text{exc} \\
t_4 & = t_2 \lor \text{err}_1 \lor \text{err}_2
\end{align*}
\]

It depends whether $\text{err}_1 \lor \text{err}_2$ 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 $\mathbb{CP}$ calculus
- Extensions
Road Map

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

\[ p ::= x \quad \text{capture variable} \]
\[ \quad | \quad p \land t \quad \text{conjunction} \]
\[ \quad | \quad p_1 | p_2 \quad \text{alternative} \]

In \( p_1 | p_2 \) we have \( \text{Var}(p_1) = \text{Var}(p_2) \)
Pattern matching

Matching an element of the model of the types

\[ d/x = \{x \mapsto d\} \]

\[ d/p \land t = d/p \quad \text{if } d \in \llbracket t \rrbracket \]

\[ d/p_1|p_2 = d/p_1 \quad \text{if } d/p_1 \text{ does not fail} \]

\[ d/p_1|p_2 = d/p_2 \quad \text{if } d/p_1 \text{ fails} \]
**Cπ calculus**

**Channels** \( \alpha \) ::= \( x \) variables
\[ \mid c^t \] channel constants

**Messages** \( M \) ::= \( n \) constant
\[ \mid \alpha \] channel

**Processes** \( P \) ::= \( \overline{\alpha}M \) output
\[ \mid \sum_{i \in I} \alpha(p_i).P_i \] patterned input
\[ \mid P_1 \parallel P_2 \] 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

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

\[
\Gamma \vdash n : b_n \quad \text{(const)}
\]

\[
\Gamma \vdash c^t : ch(t) \quad \text{(chan)}
\]

\[
\Gamma \vdash x : \Gamma(x) \quad \text{(var)}
\]

\[
\Gamma \vdash M : s \leq t \quad \text{(subs)}
\]

\[
\Gamma \vdash M : t
\]
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

\[ \frac{\Gamma \vdash P}{\Gamma \vdash (\nu c^t)P} \] (new) \[ \frac{\Gamma \vdash P}{\Gamma \vdash !P} \] (repl)

\[ \frac{\Gamma \vdash P_1 \quad \Gamma \vdash P_2}{\Gamma \vdash P_1 \parallel P_2} \] (para)

\[ t \leq \bigvee_{i \in I} \lambda p_i \} \quad \lambda p_i \} \wedge t \not= 0 \quad \frac{\Gamma \vdash \alpha : ch^+(t) \quad \Gamma, t/p_i \vdash P_i}{\Gamma \vdash \Sigma_{i \in I} \alpha(p_i).P_i} \] (input)

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

Semantic subtyping for the π-calculus – p.37
The operational semantics

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

“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 $C\pi$ 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:

\[ t = \text{int} \lor (ch(t) \land ch(\text{int})) \]

Question: is \( t = \text{int} \) 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 $ch(1)$
We can express the type of list
The model is much more complicated (CDuce)
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 $\lambda$)
- the type $ch^+(\cdot)$ is not needed
- full recursion, decidability with arrow types
Future Work

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

- $\pi$-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

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

$$\alpha : ch^+(ch^+(t_1 \lor t_2))$$
Examples

A server

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

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

A more efficient server

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

$$\alpha : ch^+(ch^+(t_1) \lor ch^+(t_2))$$
Examples

A CDuce server

\[ fun^{s \rightarrow t}(x).arg^{s}(y).result^{t}(x(y)) \]

A more liberal one

\[ compute^{((s \rightarrow t) \times s) \lor (s \times (s \rightarrow t))}(x, y \land s).result^{t}(x(y)) \]
\[ + \ compute^{((s \rightarrow t) \times s) \lor (s \times (s \rightarrow t))}(x \land s, y).result^{t}(y(x)) \]
Examples

A $\mathbb{C}_\pi$ function

\[
\text{fun assoc}(s : \text{string}, l : a\_\text{list}) : \text{ch}(\text{int}) = \\
\begin{align*}
\text{match } l \text{ with } \text{nil} & \rightarrow \text{fail} \\
| ((k,c),t) & \rightarrow \text{if } k = s \text{ then } c \\
& \quad \text{else } \text{assoc}(s,t)
\end{align*}
\]

A $\mathbb{C}_\pi$ process

\[
\text{announce}^{[m\_\text{list} \times a\_\text{list} \times \text{retr}]} (\text{marks}, \text{mails}, \text{getch}).
\]

\[
(\forall c^{m\_\text{list}}) \overline{c}(\text{marks}) \mid \\
\quad \text{!( } c((n,m),\text{rest}) \text{). ( } \text{getch}(n,\text{mails})(m) \mid \overline{c}(\text{rest}) \\
\quad + c(\text{nil}).0 \text{)}
\]

\[
m\_\text{list} = ((\text{string} \times \text{int}) \times m\_\text{list}) \lor \text{nil}
\]
\[
a\_\text{list} = ((\text{string} \times \text{ch}(\text{int})) \times a\_\text{list}) \lor \text{nil}
\]
\[
\text{retr} = \text{string} \times a\_\text{list} \rightarrow \text{ch}(\text{int})
\]
Examples

\[ \overline{\text{announce}}( (("Alice",6),(("Bob",8),\text{nil})), (("Bob",c_B),(("Alice",c_A),\text{nil})), \text{assoc} ) \]

Will reduce to process: \( \overline{c_A}(6) \mid \overline{c_B}(8) \).
Local $\mathcal{C}\pi$ calculus

Channels $\alpha ::= x$ variables
typed channel (box)

| $c^t$ |

Messages $M ::= n$ constant

| $\alpha$ channel |

Processes $P ::= \overline{\alpha}M$ output

| $\sum_{i \in I} c^t(p_i).P_i$ patterned input |

| $P_1 \parallel P_2$ parallel |

| $(\nu c^t)P$ restriction |

| $!P$ replication |
Typing Messages in local $\mathcal{C}\pi$

\[
\begin{align*}
\Gamma &\vdash n : b_n \quad \text{(const)} \\
\Gamma &\vdash x : \Gamma(x) \quad \text{(var)} \\
\Gamma &\vdash c^t : ch^-(t) \land \neg ch^-(t_1) \land \ldots \land \neg ch^-(t_k) \quad \text{(chan)} \\
\Gamma &\vdash M : s \leq t \quad \text{(subs)} \\
\Gamma &\vdash M : t
\end{align*}
\]