Sound and complete subtyping between coinductive types for object-oriented languages

Davide Ancona and Andrea Corradi

Università di Genova

PRIN CINA WP1 Meeting, Genova, Italy, July 24st, 2014

Summary

- an example with record, union, and recursive types
- types as set of values, semantic subtyping
- proof of soundness for coinductive inference systems
- a sound and complete inference system for subtyping
- a top-down algorithm

A motivating example

```
class Node:
        def __init__(self,elem):
                self.elem = elem
                self.next = self

        def getNode(self,index):
                n = self
                for i in range(0,index):
                        n = n.next
                return n

class CircularList:
        def __init__(self):
                self.head = Node(None)
                self.size = 0

        def __checkBounds(self,index,limit):
                if index < 0 or index >= limit:
                        raise IndexError("list index out of range")

        def add(self,index,elem):
                self.__checkBounds(index,self.size+1)
                n = self.head.getNode(index)
                tmp = Node(elem)
                tmp.next = n.next
                n.next = tmp;
                self.size+=1

        def get(self,index):
                self.__checkBounds(index,self.size)
                return self.head.getNode(index+1).elem
```

Inductive versus coinductive interpretation of types

```
def getNode(self,index):
        n = self
        for i in range(0,index):
                n = n.next
        return n
```

- inductive interpretation: `self`$: T_1$

$$T_1 = \mathit{null} \vee \langle \mathit{elem}{:}\mathit{int}, \mathit{next}{:}T_1 \rangle$$

- coinductive interpretation: `self`$: T_2$

$$T_2 = \langle \mathit{elem}{:}\mathit{int}, \mathit{next}{:}T_2 \rangle$$

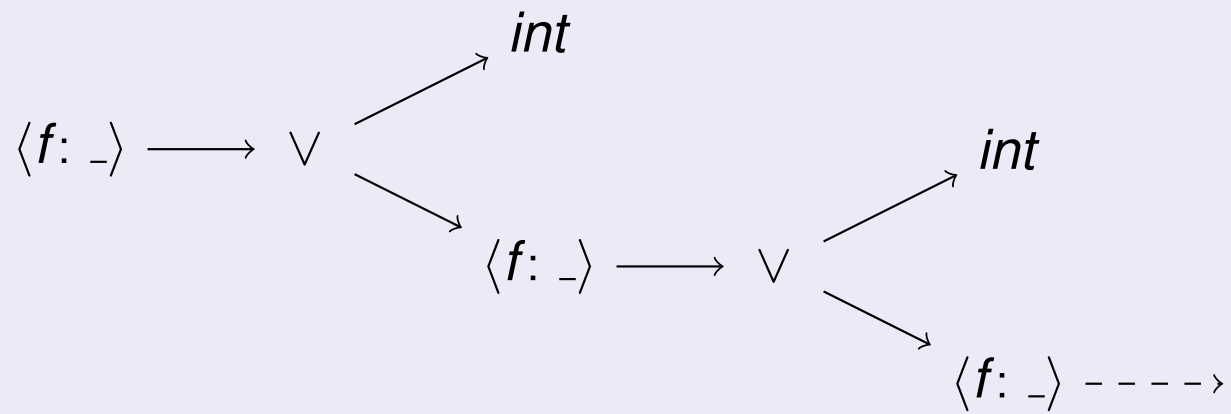
Types

Syntax

$$\tau \quad ::= \quad \mathbf{0} \mid \mathit{int} \mid \mathit{null} \mid \langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \mid \tau_1 \vee \tau_2$$

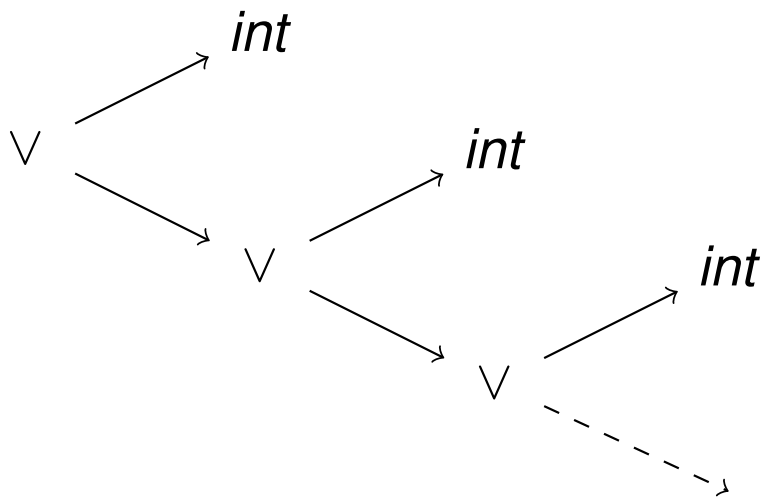
Remarks

- the definition above is interpreted coinductively
- types restricted to *regular trees*
- example: $T = \langle f{:}\ T \vee \mathit{int} \rangle$



Contractive types

- syntactic restriction: types must be *contractive*
- no paths where all nodes are the union type constructor
- $T = \langle f\colon T \vee \mathit{int} \rangle$ is contractive
- $T = T \vee \mathit{int}$ is *not* contractive



Type interpretation

Coinductive definition for values

$$v \quad ::= \quad i \mid \text{null} \mid \langle f_1 \mapsto v_1, \ldots, f_n \mapsto v_n \rangle$$

Values: finitely branching trees with possibly infinite depth

Coinductive rules for typing values

$$(\text{null} \in) \frac{}{\text{null} \in \mathit{null}} \qquad (\text{int} \in) \frac{}{i \in \mathit{int}}\ i \in \mathbb{Z} \qquad (\text{l-or} \in) \frac{v \in \tau_1}{v \in \tau_1 \vee \tau_2} \qquad (\text{r-or} \in) \frac{v \in \tau_2}{v \in \tau_1 \vee \tau_2}$$

$$(\text{rec} \in) \frac{v_1 \in \tau_1, \ldots, v_n \in \tau_n}{\langle f_1 \mapsto v_1, \ldots, f_n \mapsto v_n, \ldots \rangle \in \langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle}$$

Type interpretation and subtyping

- $[\![\tau]\!] = \{v \mid v \in \tau \text{ holds}\}$
- $\tau_1 \leq \tau_2 \overset{def}{\Leftrightarrow} [\![\tau_1]\!] \subseteq [\![\tau_2]\!]$

Step 1: checking non-emptiness of types

Coinductive rules for non-emptiness

$$\frac{}{int \not\cong \emptyset} \qquad \frac{}{null \not\cong \emptyset} \qquad \frac{\tau_i \not\cong \emptyset}{\tau_1 \vee \tau_2 \not\cong \emptyset}\ i \in 1..2 \qquad \frac{\tau_1 \not\cong \emptyset, \ldots, \tau_n \not\cong \emptyset}{\langle f{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \not\cong \emptyset}$$

- Completeness: $[\![\tau]\!] \neq \emptyset$ implies $\tau \not\cong \emptyset$ (easy coinduction on the rules)
- Soundness: $\tau \not\cong \emptyset$ implies $[\![\tau]\!] \neq \emptyset$ (proof?)
- Solution: $[\![\tau]\!] = \emptyset$ implies $\tau \cong \emptyset$ implies ($\tau \not\cong \emptyset$ does not hold)
- Clue: inference system for $\tau \cong \emptyset$ with a unique fixed point

Checking emptiness of types

Coinductive rules for emptiness

$$\frac{}{\Xi \vdash \mathbf{0} \cong \emptyset} \qquad \frac{\Xi \vdash \tau_1 \cong \emptyset \quad \Xi \vdash \tau_1 \cong \emptyset}{\Xi \vdash \tau_1 \vee \tau_2 \cong \emptyset} \qquad \frac{\Xi \cup \{\tau\} \vdash \tau_i \cong \emptyset}{\Xi \vdash \tau \cong \emptyset} \quad \begin{array}{l} \tau = \langle f{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \\ \tau \notin \Xi \\ i \in \{1, \ldots, n\} \end{array}$$

- $\tau \cong \emptyset$ shortcut for $\emptyset \vdash \tau \cong \emptyset$
- Property 1: $\Xi \Vdash \tau \cong \emptyset$ implies $\Xi \vdash \tau \cong \emptyset$
- Property 2: if $\tau \notin \Xi$, and $[\![\tau_{|\Xi}]\!] = \emptyset$, then $\Xi \Vdash \tau \cong \emptyset$.
- Corollary: $[\![\tau]\!] = \emptyset$ implies $\tau \cong \emptyset$
- Property 3: if $\Xi \vdash \tau \cong \emptyset$, then $\tau \ncong \emptyset$ does not hold.

Step 2: type normalization

Coinductive rules for type normalization

$$(\text{prim} \triangleright)\ \frac{}{\tau \triangleright \tau}\ \tau \in \{\mathbf{0}, null, int\} \qquad (\text{or} \triangleright)\ \frac{\tau_1 \triangleright \tau_1' \quad \tau_2 \triangleright \tau_2'}{\tau_1 \vee \tau_2 \triangleright \tau_1' \vee \tau_2'}\ \begin{array}{l}\tau_1 \not\cong \emptyset \\ \tau_2 \not\cong \emptyset\end{array}$$

$$(\text{r-or} \triangleright)\ \frac{\tau_1 \triangleright \tau_1'}{\tau_1 \vee \tau_2 \triangleright \tau_1'}\ \tau_2 \cong \emptyset \qquad (\text{l-or} \triangleright)\ \frac{\tau_2 \triangleright \tau_2'}{\tau_1 \vee \tau_2 \triangleright \tau_2'}\ \tau_1 \cong \emptyset$$

$$(\text{rec} \triangleright)\ \frac{\tau_1 \triangleright \tau_1', \ldots, \tau_n \triangleright \tau_n'}{\langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \triangleright \langle f_1{:}\tau_1', \ldots, f_n{:}\tau_n' \rangle}\ \langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \not\cong \emptyset$$

$$(\text{e-rec} \triangleright)\ \frac{}{\langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \triangleright \mathbf{0}}\ \langle f_1{:}\tau_1, \ldots, f_n{:}\tau_n \rangle \cong \emptyset$$

- Example: if $\tau = \mathbf{0} \vee \langle f{:}\tau, g{:}\mathbf{0} \vee \mathbf{0} \rangle$ then $\tau \triangleright \mathbf{0}$
- Property: if $\tau \triangleright \tau'$, then $[\![\tau]\!] = [\![\tau']\!]$

Introduction to the subtyping rules

- Subtyping rules based on the identity $A \subseteq B \cup C \Leftrightarrow A \setminus B \subseteq C$
- Example:

$$\langle f{:}\mathit{null} \vee \mathit{int} \rangle \leq \langle f{:}\mathit{null} \rangle \vee \langle f{:}\mathit{int} \rangle \vee \mathit{int}$$
$$\Leftrightarrow$$
$$\langle f{:}\mathit{null} \vee \mathit{int} \rangle \setminus \langle f{:}\mathit{null} \rangle \leq \langle f{:}\mathit{int} \rangle \vee \mathit{int}$$
$$\Leftrightarrow$$
$$\langle f{:}\mathit{int} \rangle \leq \langle f{:}\mathit{int} \rangle \vee \mathit{int}$$
$$\Leftrightarrow$$
$$\langle f{:}\mathit{int} \rangle \setminus \langle f{:}\mathit{int} \rangle \leq \mathit{int}$$
$$\Leftrightarrow$$
$$\mathbf{0} \leq \mathit{int}$$

Types are not closed under complement

Problem

- $T = \langle elm{:}int, next{:}T \rangle \vee null$: infinite and finite lists of integers.
- $T' = \langle elm{:}int, next{:}T' \rangle$: infinite lists of integers.
- $T \setminus T'$: all finite lists of integers. Not representable without $\setminus$.

- Solution: complement is computed lazily
- Example: $\langle f{:}\tau_1, g{:}\tau_2 \rangle \setminus \langle f{:}\tau_3, h{:}\tau_4 \rangle = \langle f{:}\tau_1 - \tau_3, g{:}\tau_2 \rangle \vee \langle f{:}\tau_1, g{:}\tau_2, h?{:} - \tau_4 \rangle$
- Definition type complement based on this identity

$$
\begin{aligned}
(A_1 \times \ldots \times A_n) \setminus (B_1 \times \ldots \times B_n) = \quad & (A_1 \setminus B_1) \times A_2 \times \ldots \times A_n \\
\cup \quad & A_1 \times (A_2 \setminus B_2) \times A_3 \times \ldots \times A_n \\
\cup \quad & \ldots \\
\cup \quad & A_1 \times \ldots \times A_{n-1} \times (A_n \setminus B_n).
\end{aligned}
$$

- Extended types

$$
\begin{aligned}
\pi &::= \tau \mid \langle f_1{:}\rho_1, \ldots, f_n{:}\rho_n, f'_1?{:}\varrho_1, \ldots, f_k?{:}\varrho_k \rangle \mid \pi_1 \vee \pi_2 \\
\rho &::= \tau \mid \rho - \tau \\
\varrho &::= -\tau \mid \varrho - \tau
\end{aligned}
$$

Subtyping rules

$$(\text{empty} \leq)\ \frac{}{\mathbf{0} \leq \Xi} \qquad (\text{left-or} \leq)\ \frac{\pi_1 \leq \Xi \quad \pi_2 \leq \Xi}{\pi_1 \vee \pi_2 \leq \Xi} \qquad (\text{right-or} \leq)\ \frac{\pi \leq \Xi \cup \{\tau_1, \tau_2\}}{\pi \leq \Xi \cup \{\tau_1 \vee \tau_2\}}\ \tau_1 \vee \tau_2 \notin \Xi$$

$$(\text{comp} \leq)\ \frac{\pi' \leq \Xi}{\pi \leq \Xi \cup \{\tau\}}\ \begin{array}{l}\tau \notin \Xi \\ \pi \setminus \tau = \pi'\end{array} \qquad (\text{rec} \leq)\ \frac{\tau' \leq \Xi}{\langle \ldots f{:}\rho - \tau \ldots \rangle \leq \emptyset}\ \rho - \tau \rightsquigarrow \tau' - \Xi$$

- the judgment $\pi \leq \{\tau_1, \ldots, \tau_n\}$ corresponds to $\pi \leq \tau_1 \vee \ldots \vee \tau_n$
- all types are assumed to be regular, contractive and normalized

Subtyping rules

$$(\text{empty} \leq)\ \frac{}{\mathbf{0} \leq \Xi} \qquad (\text{left-or} \leq)\ \frac{\pi_1 \leq \Xi \quad \pi_2 \leq \Xi}{\pi_1 \vee \pi_2 \leq \Xi} \qquad (\text{right-or} \leq)\ \frac{\pi \leq \Xi \cup \{\tau_1, \tau_2\}}{\pi \leq \Xi \cup \{\tau_1 \vee \tau_2\}}\ \tau_1 \vee \tau_2 \notin \Xi$$

$$(\text{comp} \leq)\ \frac{\pi' \leq \Xi}{\pi \leq \Xi \cup \{\tau\}}\ \begin{array}{l}\tau \notin \Xi \\ \pi \setminus \tau = \pi'\end{array} \qquad (\text{rec} \leq)\ \frac{\tau' \leq \Xi}{\langle \ldots f{:}\rho - \tau \ldots \rangle \leq \emptyset}\ \rho - \tau \rightsquigarrow \tau' - \Xi$$

- the judgment $\pi \leq \{\tau_1, \ldots, \tau_n\}$ corresponds to $\pi \leq \tau_1 \vee \ldots \vee \tau_n$
- all types are assumed to be regular, contractive and normalized

$$\frac{\rho - \tau \rightsquigarrow \tau'' - \Xi}{(\rho - \tau) - \tau' \rightsquigarrow \tau'' - (\Xi \cup \{\tau'\})} \qquad \frac{}{\tau - \tau' \rightsquigarrow \tau - \{\tau'\}}$$

$$((\mathit{int} - \mathit{null}) - \langle f{:}\mathit{int}\rangle) - \langle f{:}\mathit{null}\rangle \rightsquigarrow \mathit{int} - \{\mathit{null}, \langle f{:}\mathit{int}\rangle, \langle f{:}\mathit{null}\rangle\}$$

Type complement

$\tau \setminus \tau = \mathbf{0}$

$\pi \setminus \mathbf{0} = \pi$

$\tau \setminus \tau' = \tau$ if $\tau \neq \tau', \tau \in \{int, null\}$ and $\tau' \neq \tau_1 \vee \tau_2$

$\mathbf{0} \setminus \tau = \mathbf{0}$

$\pi \setminus \tau = \pi$ if $\pi = \langle \ldots \rangle$ and $\tau \in \{int, null\}$

$\pi \setminus \tau = (\bigvee_{f \in dom(\pi) \cap dom(\tau)} \pi -_f \tau) \bigvee (\bigvee_{f \in dom(\tau) \setminus dom(\pi)} \pi \sim_f \tau)$ if $\pi, \tau = \langle \ldots \rangle$

where
$\pi -_f \tau = \pi[f{:}\rho - \tau']$ if $\pi = \langle \ldots f{:}\rho \ldots \rangle$ $\tau = \langle \ldots f{:}\tau' \ldots \rangle$
$\pi -_f \tau = \pi[f?{:}\varrho - \tau']$ if $\pi = \langle \ldots f?{:}\varrho \ldots \rangle$ $\tau = \langle \ldots f{:}\tau' \ldots \rangle$
$\pi \sim_f \tau = \pi[f?{:} - \tau']$ if $\tau = \langle \ldots f{:}\tau' \ldots \rangle$

Soundness

If $\pi \leq \{\tau_1, \ldots, \tau_n\}$ then $[\![\pi]\!] \subseteq [\![\tau_1 \vee \ldots \vee \tau_n]\!]$.

$$(\text{prim } \not\leq)\frac{}{\Psi \vdash \tau \not\leq \emptyset}\ \tau \in \{\mathit{null}, \mathit{int}\} \qquad (\text{comp } \not\leq)\frac{\Psi \vdash \pi' \not\leq \Xi}{\Psi \vdash \pi \not\leq \Xi \cup \{\tau\}}\ \begin{array}{l}\tau \notin \Xi \\ \pi \setminus \tau = \pi'\end{array}$$

$$(\text{l-left-or } \not\leq)\frac{\Psi \vdash \pi_1 \not\leq \Xi}{\Psi \vdash \pi_1 \vee \pi_2 \not\leq \Xi} \qquad (\text{r-left-or } \not\leq)\frac{\Psi \vdash \pi_2 \not\leq \Xi}{\Psi \vdash \pi_1 \vee \pi_2 \not\leq \Xi}$$

$$(\text{right-or } \not\leq)\frac{\Psi \vdash \pi \not\leq \Xi \cup \{\tau_1, \tau_2\}}{\Psi \vdash \pi \not\leq \Xi \cup \{\tau_1 \vee \tau_2\}}\ \tau_1 \vee \tau_2 \notin \Xi$$

$$(\text{rec } \not\leq)\frac{\forall\, f \in \mathit{dom}(\pi)\ \pi(f) = \rho - \tau \rightsquigarrow \tau' - \Xi \Rightarrow \Psi \cup \{\pi\} \vdash \tau' \not\leq \Xi}{\Psi \vdash \pi \not\leq \emptyset}\ \begin{array}{l}\pi = \langle \ldots \rangle \\ \pi \notin \Psi\end{array}$$

- Property 1: Ξ finite, $\Psi \Vdash \tau \not\leq \Xi$ implies $\Psi \vdash \tau \not\leq \Xi$
- Property 2: for all $\pi' \in \Psi$, π' has shape $\langle \ldots f{:}\rho - \tau \ldots \rangle$, $\pi \notin \Psi$ and $[\![\pi]\!] \not\subseteq [\![\tau_1 \vee \ldots \vee \tau_n]\!]$ imply $\Psi \Vdash \pi \not\leq \{\tau_1, \ldots, \tau_n\}$.
- Property 3: if $\Psi \vdash \pi \not\leq \Xi$, then $\pi \leq \Xi$ does not hold.

Completeness

If $[\![\pi]\!] \subseteq [\![\tau_1 \vee \ldots \vee \tau_n]\!]$, then $\pi \leq \{\tau_1, \ldots, \tau_n\}$ holds.

- Property 1: if $\pi \setminus \tau = \pi'$, then $[\![\pi]\!] \setminus [\![\tau]\!] = [\![\pi']\!]$
- Property 2: π is a record type s.t. $[\![\pi]\!] = \emptyset$ if and only if there exist f, ρ, and τ s.t. π has shape $\langle \ldots f{:}\rho - \tau \ldots \rangle$, and $[\![\rho - \tau]\!] = \emptyset$
  - fields f:τ can not be empty because of normalization
  - π cannot be empty because of an optional field f, it always contain all record values that do not have field f?
- Property 3: for all ρ, τ, there exist unique τ', Ξ s.t. $\rho - \tau \rightsquigarrow \tau' - \Xi$ holds
- Property 4: if $\rho - \tau \rightsquigarrow \tau' - \{\tau_1, \ldots, \tau_n\}$,
  then $[\![\rho - \tau]\!] = [\![\tau' - (\tau_1 \vee \ldots \vee \tau_n)]\!]$

Implementation

```
// pre-condition: π and all types in Ξ are normalized
boolean subtype(Set<Pair<ExtType,Set<Type>>> Ψ, ExtType π,Set<Type> Ξ) {
    if (π==0)   // rule (empty ≤)
        return true
    if (∃(π′,Ξ′) ∈ Ψ s.t. π′==π && Ξ′ ⊆ Ξ) // termination condition
        return true
    while(∃τ1,τ2 s.t. τ1 ∨ τ2 ∈ Ξ) // rule (right-or ≤)
        Ξ=(Ξ \ τ1 ∨ τ2) ∪ {τ1,τ2}
    if (∃π1,π2 s.t. π==π1 ∨ π2) // rule (left-or ≤)
        return subtype(Ψ, π1, Ξ) && subtype(Ψ, π2, Ξ)
    else if (∃τ ∈ Ξ) { // rule (comp ≤)
            π′=π \ τ
            Ξ′=Ξ \ {τ}
            return subtype(Ψ ∪ {(π,{τ})},π′,∅) ||
                   Ξ′!=∅ && subtype(Ψ ∪ {(π,Ξ)},π′,Ξ′)
        }
    else if (π==⟨...⟩){ // rule (rec ≤)
            foreach f ∈ dom(π)
                if (∃ρ,τ s.t. π(f)==ρ − τ) {
                    ρ − τ ⇝ τ′ − Ξ′
                    if (subtype(Ψ,τ′,Ξ′))
                        return true
                }
            return false
        }
    else
        return false          // int ≤ ∅ and null ≤ ∅ do not hold
}
```

Related work

- Semantic subtyping in XDuce [HosoyaPierce03] and CDuce [FrischCastagnaBenzaken08], inductive semantics
- Subtyping equivalent to the inclusion problem of regular tree languages (EXPTIME-complete)
- Sound and complete algorithm to avoid systematic realization of any potential exponential blowup in the tree automata
- More recently semantic subtyping between coinductive types have been considered [AnconaLagorio09] for static type analysis of object-oriented languages
- Ongoing work [BonsangueRotAnconaDeBoerRutten] coalgebraic foundation for semantic subtyping between coinductive types

Future work

Extensions to

- updatable records
- intersection types
- polymorphism, subtyping between non ground types

Thank you