Subtyping

Stefan Muller

CSE 5095: Types and Programming Languages, Fall 2025
Lecture 17–18

1 Subsumption

Consider the types nat and int. We want to be able to consider anything of nat to be of type int. We'll say nat is a "subtype" of int:

$$\mathsf{nat} <: \mathsf{int}$$

In general, we'll use $\tau <: \tau'$ to say that τ is a subtype of τ'. There are a few ways of interpreting this:

- Any τ can behave like a τ'.
- Anything that expects a τ' can accept a τ.
- Anything of type τ can have type τ'.

The last point is explicitly allowed by the "subsumption rule":

$$\frac{\Gamma \vdash e : \tau \qquad \tau <: \tau'}{\Gamma \vdash e : \tau'}\ (\textsc{Subsumption})$$

2 Example: Products

Consider n-ary products $\tau_1 \times \cdots \times \tau_n$:

$$\frac{\forall i, \Gamma \vdash e_i : \tau_i}{(e_1, \ldots, e_n) : \tau_1 \times \cdots \times \tau_n}\ (\times\text{-I}) \qquad \frac{\Gamma \vdash e : \tau_1 \times \cdots \times \tau_n \qquad 1 \leq i \leq n}{\Gamma \vdash \pi_i\ e : \tau_i}\ (\times\text{-E})$$

Let:

$$\begin{aligned} \mathsf{fst2} &\triangleq \lambda x : \mathsf{int} \times \mathsf{int}. \pi_1\ x \\ \mathsf{fst3} &\triangleq \lambda x : \mathsf{int} \times \mathsf{int} \times \mathsf{int}. \pi_1\ x \end{aligned}$$

The expression $\mathsf{fst2}\ (1, 2, 3)$ is perfectly safe but not well-typed. In general, $\pi_i\ e$ is safe for $e : \tau_1 \times \cdots \times \tau_n$ as long as $i \leq n$. Since projection is the only thing that can "expect" (eliminate) something of product type, that should mean that

$$\mathsf{int} \times \mathsf{int} \times \mathsf{int} <: \mathsf{int} \times \mathsf{int}$$

and in general

$$\frac{}{\tau_1 \times \cdots \times \tau_n \times \cdots \times \tau_{n+k} <: \tau_1 \times \cdots \times \tau_n}\ (\textsc{Sub-Width})$$

We call this "width subtyping".

With this and the subsumption rule, we can type $\mathsf{fst2}\ (1, 2, 3)$:

$$\dfrac{\dfrac{\dots}{\bullet \vdash \mathsf{fst2} : (\mathsf{int} \times \mathsf{int}) \to \mathsf{int}} \quad \dfrac{\dfrac{\dots}{\bullet \vdash (1,2,3) : \mathsf{int} \times \mathsf{int} \times \mathsf{int}}\ (\times\text{-I}) \quad \dfrac{}{\mathsf{int} \times \mathsf{int} \times \mathsf{int} <: \mathsf{int} \times \mathsf{int}}\ (\textsc{Sub-W.})}{\bullet \vdash (1,2,3) : \mathsf{int} \times \mathsf{int}}\ (\textsc{Subs.})}{\bullet \vdash \mathsf{fst2}\ (1,2,3) : \mathsf{int}}\ (\to\text{-E})$$

Note that it would **not** be safe to say

$$\mathsf{int} \times \mathsf{int} <: \mathsf{int} \times \mathsf{int} \times \mathsf{int}$$

because we cannot allow something like $\pi_3\ (1,2)$.

What about

- $(\mathsf{int} \times \mathsf{int} \times \mathsf{int}) \times \mathsf{int} <: (\mathsf{int} \times \mathsf{int}) \times \mathsf{int}$
- $\mathsf{int} \times \mathsf{nat} <: \mathsf{int} \times \mathsf{int}$

These should be fine, but we don't have the rules to show them. Enter "depth subtyping":

$$\dfrac{\forall i, \tau_i <: \tau_i'}{\tau_1 \times \cdots \times \tau_n <: \tau_1' \times \cdots \times \tau_n'}\ (\textsc{Sub-Depth})$$

3 Properties of Subtyping

Note that to derive both of the subtyping relations above, we still need $\mathsf{int} <: \mathsf{int}$. In general, we require that subtyping is **reflexive and transitive**. There are two ways to do this:

3.1 Set up the rules carefully so we can prove it.

We need a rule for products that combines width and depth subtyping.

$$\dfrac{}{\mathsf{unit} <: \mathsf{unit}}\ (\textsc{Sub-Unit}) \qquad \dfrac{}{\mathsf{int} <: \mathsf{int}}\ (\textsc{Sub-Int}) \qquad \dfrac{\forall i \in [1,n]. \tau_i <: \tau_i'}{\tau_1 \times \cdots \times \tau_n \times \cdots \times \tau_{n+k} <: \tau_1' \times \cdots \times \tau_n'}\ (\textsc{Sub-Prod})$$

Lemma 1. *For all τ, $\tau <: \tau$.*

Proof. By induction on the structure of τ.

- unit, int. By Sub-Unit, Sub-Int.
- $\tau_1 \times \cdots \times \tau_n$. By induction, $\tau_i <: \tau_i$. Apply Sub-Prod.

□

Lemma 2. *If $\tau_1 <: \tau_2$ and $\tau_2 <: \tau_3$, then $\tau_1 <: \tau_3$.*

Proof. By induction on the derivation of $\tau_1 <: \tau_2$ and $\tau_2 <: \tau_3$

- If the first derivation is by Sub-Unit or Sub-Int, then $\tau_1 = \tau_2$, so we have $\tau_1 <: \tau_3$ by assumption.
- If the second derivation is by Sub-Unit or Sub-Int, then $\tau_2 = \tau_3$, so we have $\tau_1 <: \tau_3$ by assumption.
- Sub-Prod, Sub-Prod. We have

  $$\tau_1 \times \cdots \times \tau_{n+k+l} <: \tau_1' \times \cdots \times \tau_{n+k}' <: \tau_1'' \times \cdots \times \tau_n''$$

  where $\tau_i <: \tau_i' <: \tau_i''$. Apply Sub-Prod

□

This gets a lot harder as we add more rules.

3.2 Add explicit rules for reflexivity and transitivity

$$\frac{}{\tau <: \tau}\ (\textsc{Sub-Refl}) \qquad \frac{\tau_1 <: \tau_2 \qquad \tau_2 <: \tau_3}{\tau_1 <: \tau_3}\ (\textsc{Sub-Trans})$$

This generally makes the theory cleaner and easier, but is a huge pain to actually implement in a type checker.

4 Other Subtyping Rules

What about n-ary sums?

$$\frac{}{\tau_1 + \cdots + \tau_n <: \tau_1 + \cdots + \tau_n + \cdots + \tau_{n+k}}\ (\textsc{Sub-Width-Sum}) \qquad \frac{\tau_i <: \tau_i'}{\tau_1 + \cdots + \tau_n <: \tau_1' + \cdots + \tau_n'}\ (\textsc{Sub-Depth-Sum})$$

How about functions?

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'}\ (\textsc{Sub-Fun})$$

Covariance and Contravariance Note that the argument types don't go the way you expect! Rules Sub-Depth and Sub-Depth-Sum are *covariant:* the subtyping relations of the types go in the same direction as their components. Function subtyping is covariant in the result types but *contra*variant in the argument types!

Consider whether we should allow

$$\mathsf{int} \times \mathsf{int} \times \mathsf{int} \to \mathsf{int} <: \mathsf{int} \times \mathsf{int} \to \mathsf{int}$$

Take the function

$$\mathsf{thd} \triangleq \lambda x : \mathsf{int} \times \mathsf{int} \times \mathsf{int}. \pi_3\ x$$

This would allow us to type $\bullet \vdash \mathsf{thd} : \mathsf{int} \times \mathsf{int} \to \mathsf{int}$ and therefore type $\mathsf{thd}\ (1, 2)$, but this is clearly unsafe!

When can we use something of type $\tau_1 \to \tau_2$ when we expect a $\tau_1' \to \tau_2'$? If we expect a $\mathsf{int} \to \mathsf{int}$, we might give it an int, like -1, but this would be a problem if we got a $\mathsf{nat} \to \mathsf{nat}$! (On the other hand, if we got an $\mathsf{int} \to \mathsf{nat}$, this would be fine: we'll never know it's only giving us nats back).

5 Examples

$$\begin{array}{lcl}
\mathsf{nat} \to \mathsf{int} & \not<: & \mathsf{int} \to \mathsf{nat} \\
\mathsf{int} \to \mathsf{nat} & <: & \mathsf{nat} \to \mathsf{int} \\
\mathsf{int} \times \mathsf{nat} & \not<: & \mathsf{nat} \times \mathsf{int} \\
\mathsf{nat} + \mathsf{nat} & <: & \mathsf{int} + \mathsf{int} \\
\mathsf{nat} + \mathsf{nat} & <: & \mathsf{int} + \mathsf{int} + \mathsf{int} \\
\mathsf{int} \to \mathsf{int} \times \mathsf{int} \times \mathsf{int} & <: & \mathsf{int} \to \mathsf{int} \times \mathsf{int}
\end{array}$$

6 Progress and Preservation

Lemma 3. *If $\tau <: \tau_1 \times \cdots \times \tau_n$ then $\tau = \tau_1' \times \cdots \times \tau_m'$ for some $m \geq n$ where for all $i \in [1, n]$, we have $\tau_i' <: \tau_i$.*

Proof. By induction on the derivation of $\tau <: \tau_1 \times \cdots \times \tau_n$.

- Sub-Refl. The result is clear.
- Sub-Width. Then $\tau = \tau_1 \times \cdots \times \tau_m$ for $m \geq n$. We have $\tau_i <: \tau_i$ by Sub-Refl.
- Sub-Depth. Then $\tau = \tau_1' \times \cdots \times \tau_n'$ where $\tau_i' <: \tau_i$.

- Sub-Trans. Then $\tau <: \tau'$ and $\tau' <: \tau_1 \times \cdots \times \tau_n$. By induction, $\tau' = \tau_1' \times \cdots \times \tau_m'$ for some $m \geq n$ where for all $i \in [1, n]$, we have $\tau_i' <: \tau_i$. By another induction, $\tau = \tau_1'' \times \cdots \times \tau_k''$ for some $k \geq m$ where for all $i \in [1, m]$, we have $\tau_i'' <: \tau_i'$. Apply transitivity of $\geq$ and $<:$.

□

Lemma 4 (Canonical Forms (Old)). *If $\bullet \vdash e : \tau_1 \times \cdots \times \tau_n$ and e val, then $e = (v_1, \ldots, v_n)$ where v_i val.*

This is no longer true!

Lemma 5 (Canonical Forms (New)). *If $\bullet \vdash e : \tau_1 \times \cdots \times \tau_n$ and e val, then $e = (v_1, \ldots, v_m)$ where v_i val and $m \geq n$.*

Proof. By induction on the derivation of $\bullet \vdash e : \tau_1 \times \cdots \times \tau_n$.

- $\times$-I. Then $e = (v_1, \ldots, v_n)$. By inversion on the value rules, we have v_i val.
- Subsumption. Then $\bullet \vdash e : \tau$ and $\tau <: \tau_1 \times \cdots \times \tau_n$. By Lemma **??**, $\tau = \tau_1' \times \cdots \times \tau_m'$ for some $m \geq n$ where for all $i \in [1, n]$, we have $\tau_i' <: \tau_i$. By induction, $e = (v_1, \ldots, v_k)$ where v_i val and $k \geq m$. We have $k \geq n$.

□

This used to be obvious just by doing inversion on the typing rules; it's not anymore!

Lemma 6 (Inversion on Typing).

1. *If $\bullet \vdash (e_1, \ldots, e_n) : \tau$ then $\tau_1 \times \cdots \times \tau_n <: \tau$ and for all i, $\bullet \vdash e_i : \tau_i$.*
2. *If $\bullet \vdash \pi_i\ e : \tau$ then $\bullet \vdash e : \tau_1 \times \cdots \times \tau_n$ and $1 \leq i \leq n$ and $\tau_i <: \tau$.*

Lemma 7 (Preservation). *If $\bullet \vdash e : \tau$ and $e \mapsto e'$ then $\bullet \vdash e' : \tau$.*

Proof. Consider the case for $\pi_i\ (e_1, \ldots, e_n) \mapsto e_i$. Then by Lemma **??**, $\bullet \vdash (e_1, \ldots, e_n) : \tau_1 \times \cdots \times \tau_m$ and $1 \leq i \leq m$ and $\tau_i <: \tau$. By another application of Lemma **??**, we have $\tau_1' \times \cdots \times \tau_n' <: \tau_1 \times \cdots \times \tau_m$ and $\bullet \vdash e_i : \tau_i'$. By Lemma **??**, we have $n \geq m$ and $\tau_i' <: \tau_i$. By Subsumption, we have $\bullet \vdash e_i : \tau_i$ and $\bullet \vdash e_i : \tau$. □

Lemma 8 (Progress). *If $\bullet \vdash e : \tau$ then either e val or $e \mapsto e'$.*

Proof. By induction on the derivation of $\bullet \vdash e : \tau$.

- $\times$-E. Then $e = \pi_i\ e_0$ and $\bullet \vdash e_0 : \tau_1 \times \cdots \times \tau_n$ and $1 \leq i \leq n$. By induction, either e_0 val or $e_0 \mapsto e_0'$. Consider the case where e_0 val. Then, by Canonical Forms, $e_0 = (v_1, \ldots, v_m)$ where v_i val and $m \geq n$. We have $e \mapsto v_i$.
- Subsumption. Then $\bullet \vdash e : \tau'$ and $\tau' <: \tau$. By induction.

□