CS4400 Lecture 11: Bidirectional Typing and STLC Metatheory

Feburary 6, 2026

1 Lecture outline

- Review: STLC
- Bidirectional typing for STLC
- Metatheory: progress and preservation

2 Review: STLC

Types:

$$\tau, \sigma ::= \mathsf{Unit} \mid \tau \times \sigma \mid \tau + \sigma \mid \tau \to \sigma$$

Expressions:

$$e ::= x \mid \lambda x.e \mid e_1\ e_2 \mid () \mid (e_1, e_2) \mid e.1 \mid e.2$$

$$\mid \mathsf{tag}_1 e \mid \mathsf{tag}_2 e \mid \mathsf{case}(e, x.e_1, y.e_2)$$

Contexts:

$$\Gamma ::= \cdot \mid \Gamma, x : \tau$$

Judgment $\boxed{\Gamma \vdash e : \tau}$:
Functions (→; lambda and application):

$$\frac{\Gamma, x : \tau \vdash e : \sigma}{\Gamma \vdash \lambda x.e : \tau \to \sigma} \to I$$

Products (×; pairs and projection):

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \times I \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.1 : \tau_1} \times E_1 \qquad \frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash e.2 : \tau_2} \times E_2$$

Sums (+; tags and case analysis):

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \mathsf{tag}_1(e) : \tau_1 + \tau_2} +I_1 \qquad \frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \mathsf{tag}_2(e) : \tau_1 + \tau_2} +I_2$$

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma, x : \tau_1 \vdash e_1 : \sigma \quad \Gamma, x : \tau_2 \vdash e_2 : \sigma}{\Gamma \vdash \mathsf{case}(e, x.e_1, y.e_2) : \sigma} +E$$

Unit (the empty tuple):

$$\frac{}{() : \mathsf{Unit}}\ \mathsf{Unit}/I \qquad \text{(no Unit E)}$$

3 Bidirectional Typing

Judgments:

$\boxed{\Gamma \vdash e \leftharpoonup \tau}$ means e checks at type τ under assumptions/variables in Γ. All of Γ, e, and τ are inputs.

$\boxed{\Gamma \vdash e \rightharpoonup \tau}$ means the expression e synthesizes the type τ in Γ. Γ and e are inputs; τ is an output.

Recommendation: fill in the blanks during lecture as you follow along.

Products:

$$\frac{\Gamma \vdash \boxed{} \quad \Gamma \vdash \boxed{}}{\Gamma \vdash (e_1, e_2) \leftharpoonup \tau_1 \times \tau_2}\times I \qquad \frac{\Gamma \vdash \boxed{}}{\Gamma \vdash e.1 \rightharpoonup \tau_1}\times E_1 \qquad \frac{\Gamma \vdash \boxed{}}{\Gamma \vdash e.2 \rightharpoonup \tau_2}\times E_2$$

Functions:

$$\frac{\Gamma, \boxed{} \vdash \boxed{}}{\Gamma \vdash \lambda x.e \leftharpoonup \tau \to \sigma}\to I \qquad \frac{\Gamma \vdash \boxed{} \quad \Gamma \vdash \boxed{}}{\Gamma \vdash e_1\ e_2 \rightharpoonup \sigma}\to E$$

Sums:

$$\frac{\Gamma \vdash \boxed{}}{\Gamma \vdash \mathsf{tag}_1(e) \leftharpoonup \tau_1 + \tau_2}+I_1 \qquad \frac{\Gamma \vdash \boxed{}}{\Gamma \vdash \mathsf{tag}_2(e) \leftharpoonup \tau_1 + \tau_2}+I_2$$

$$\frac{\Gamma \vdash \boxed{} \quad \Gamma, \boxed{} \vdash \boxed{} \quad \Gamma, \boxed{} \vdash \boxed{}}{\Gamma \vdash \mathsf{case}(e, x.e_1, y.e_2) \leftharpoonup \sigma}+E$$

Unit:

$$\frac{}{\Gamma \vdash \boxed{}}\mathsf{Unit}/I \qquad \text{(no Unit E)}$$

Bidirectional typing machinery: switching between synthesis and checking:

$$\frac{\Gamma \vdash \boxed{} \quad \boxed{}}{\Gamma \vdash (e) \leftharpoonup \tau}\mathsf{chk/syn} \qquad \frac{\Gamma \vdash \boxed{}}{\Gamma \vdash (e : \tau) \rightharpoonup \tau}\mathsf{anno}$$

3.1 Exercises

Write terms that inhabit the following types, and give their typing derivations:

1. Commutativity of sum: $(\tau_1 + \tau_2) \to (\tau_2 + \tau_1)$

2. Currying: $(\tau_1 \times \tau_2 \to \sigma) \to (\tau_1 \to \tau_2 \to \sigma)$

3. Uncurrying: $(\tau_1 \to \tau_2 \to \sigma) \to (\tau_1 \times \tau_2 \to \sigma)$

Add typing annotations, where needed, to synthesize types for the following terms.

1. $\mathsf{case}(\mathsf{tag}_2(), y.y, z.z)$

2. $(\lambda x.(), \lambda y.()).1$

3. $(\lambda x.\ x.2)\ ((), ())$

4 Dynamic Semantics and Metatheory

The definitional interpreter we will provide you for the homework uses the following (big-step) semantics:

4.1 Big-step evaluation semantics

Immediate values:

$$\frac{}{() \Downarrow ()} \Downarrow/() \qquad \frac{}{\lambda x.e \Downarrow \lambda x.e} \Downarrow/\lambda$$

Pairs (eager):

$$\frac{e_1 \Downarrow v_1 \quad e_2 \Downarrow v_2}{(e_1, e_2) \Downarrow (v_1, v_2)} \Downarrow \mathsf{pair} \qquad \frac{e \Downarrow (v_1, v_2)}{e.1 \Downarrow v_1} \Downarrow \mathsf{proj}_1 \qquad \frac{e \Downarrow (v_1, v_2)}{e.2 \Downarrow v_2} \Downarrow \mathsf{proj}_2$$

Sums:

$$\frac{e \Downarrow v}{\mathsf{tag}_1\ e \Downarrow \mathsf{tag}_1\ v} \Downarrow \mathsf{tag}_1 \qquad \frac{e \Downarrow v}{\mathsf{tag}_2\ e \Downarrow \mathsf{tag}_1\ v} \Downarrow \mathsf{tag}_2$$

$$\frac{e \Downarrow \mathsf{tag}_i\ v \quad [v/x]e_i \Downarrow v'}{\mathsf{case}(e, x.e_1, x.e_2) \Downarrow v'} \Downarrow \mathsf{case}_i$$

Function application:

$$\frac{e_1 \Downarrow \lambda x.e \quad e_2 \Downarrow v_2 \quad [v_2/x]e \Downarrow v}{e_1\ e_2 \Downarrow v} \Downarrow/\mathsf{app}$$

4.2 Metatheory

Previously (for IfLang), we were able to show a very strong theorem about the relationship between typing and evaluation:

Theorem 1 (Type soundness). *For all e and τ, if $e : \tau$, then there exists a value v such that $e \Downarrow v$ and $v : \tau$.*

This theorem is *true* for the simply-typed lambda calculus. However, showing it is much less trivial, and outside the scope of this class. We can instead show something slightly weaker, but also more general, in the sense that it does not assume all programs terminate.

To do this we need to define *small-step semantics*, which we derive from the "reduction" rules we saw last time:

4.2.1 Small-step semantics

Define the judgment $e \mapsto e'$ as:

$$\frac{}{(\lambda x.e)\ v \mapsto [v/x]e} \qquad \frac{}{(v_1, v_2).1 \mapsto v_1} \qquad \frac{}{(v_1, v_2).2 \mapsto v_2}$$

$$\frac{}{\mathsf{case}(\mathsf{tag}_1 v, x.e_1, y.e_2) \mapsto [v/x]e_1} \qquad \frac{}{\mathsf{case}(\mathsf{tag}_2 v, x.e_1, y.e_2) \mapsto [v/y]e_2}$$

In addition to these primary "computation" rules (also known as β-reduction rules), we need several "boring" rules (also known as congruence rules) that crawl the term to find the redex, step it, and put the term back together:

$$\frac{e_1 \mapsto e_1'}{e_1\ e_2 \mapsto e_1'\ e_2} \qquad \frac{e_2 \mapsto e2'}{v\ e_2 \mapsto v\ e_2'} \qquad \frac{e \mapsto e'}{e.1 \mapsto e'.1} \qquad \frac{e \mapsto e'}{e.2 \mapsto e'.2}$$

$$\frac{e \mapsto e'}{\mathsf{tag}_1 e \mapsto \mathsf{tag}_1 e'} \qquad \frac{e \mapsto e'}{\mathsf{tag}_2 e \mapsto \mathsf{tag}_2 e'} \qquad \frac{e \mapsto e'}{\mathsf{case}(e, x.e_1, y.e_2) \mapsto \mathsf{case}(e', x.e_1, y.e_2)}$$

Note that for this rule set, values do not step.

4.2.2 Progress and Preservation

We are now able to state and prove the following two theorems:

Theorem 2 (Modified type soundness)**.** *The following both hold:*

1. *(Progress.) If* $\cdot \vdash e : \tau$*, then either there exists* e' *such that* $e \mapsto e'$*, or* e *is already a value.*
2. *(Preservation.) If* $\cdot \vdash e : \tau$ *and* $e \mapsto e'$*, then* $e' : \tau$*.*