Types, contexts and terms

Types:
$A, B ::= A \to B \mid A \times B \mid A + B \mid \mathbf{1} \mid \mathbf{0}$

Typing contexts:
$\Gamma ::= \cdot \mid \Gamma, x : A$

Extrinsic STLC

Extrinsic STLC is the simplest version of typed lambda calculus. The terms are the same as in untyped lambda calculus (with the addition of terms for products, sums, unit and empty). The typing relation takes the form of a type assignment system – we describe which terms have which types, without worrying about issues such as implementation.

Terms

Terms:

$$
\begin{aligned}
e ::= {} & \\
& x \mid \\
& \lambda x.e \mid e_1\ e_2 \mid \\
& (e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid \\
& \texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (e_1, e_2) \mid \\
& \texttt{unit} \mid \texttt{exfalso}\ e
\end{aligned}
$$

Declarative typing – basics

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\textsc{Var}$$

Declarative typing – type-directed rules

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \texttt{outl}\ e : A} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \texttt{outr}\ e : B}$$

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \texttt{inl}\ e : A + B} \qquad \frac{\Gamma \vdash e : B}{\Gamma \vdash \texttt{inr}\ e : A + B}$$

$$\frac{\Gamma \vdash e : A + B \quad \Gamma \vdash f : A \to C \quad \Gamma \vdash g : B \to C}{\Gamma \vdash \mathsf{case}\ e\ \mathsf{of}\ (f, g) : C}$$

$$\frac{}{\Gamma \vdash \texttt{unit} : \mathbf{1}} \qquad \frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \texttt{exfalso}\ e : A}$$

Metatheory

Extrinsic STLC enjoys strong metatheoretical properties:

- Confluence: if a term can be reduced to two different terms, these two can in turn be reduced to a common result.
- Termination: computation on well-typed terms always terminates.
- Type preservation: computing with a well-typed term preserves its type.
- Canonicity: in the empty context, normal forms are inductively generated from term constructors.

Intuitively: given a well-typed term, in finite time it computes to another term of the same type which cannot compute anymore. In the empty context, we know the result of this computation must be a constructor.

Uniqueness of typing

However, extrinsic STLC does not enjoy another important property: **uniqueness of typing**. This means that there are terms which can be assigned multiple types. For example, $\lambda x.x$ can be assigned the type $A \to A$ for any type A. There four culprits of this failure are lambda abstractions, sum constructors and exfalso.

Bidirectional STLC

Bidirectional STLC is a version of simply typed lambda calculus which focuses on an efficient implementation of the typechecker. The terms are as in Extrinsic STLC, but with the addition of a general type annotation construct that can appear anywhere and is not mandatory. The typing relation can be presented as type assignment system for reference, but more important are the type checking and type inference relations, both of which are algorithmic, i.e. easily implementable.

Terms

Note: green color marks terms which were not present in Extrinsic STLC.

Terms:

$e ::=$

$x \mid (e : A) \mid$

$\lambda x.e \mid e_1\ e_2 \mid$

$(e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid$

$\texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (e_1, e_2) \mid$

$\texttt{unit} \mid \texttt{exfalso}\ e$

Declarative typing – new rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash (e : A) : A}\textsc{Annot}$$

Bidirectional typing – basics

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \Rightarrow A}\textsc{Var}$$

$$\frac{\Gamma \vdash e \Leftarrow A}{\Gamma \vdash (e : A) \Rightarrow A}\textsc{Annot}$$

$$\frac{\Gamma \vdash e \Rightarrow B \quad A = B}{\Gamma \vdash e \Leftarrow A}\textsc{Sub}$$

Bidirectional typing – type-directed rules

$$\frac{\Gamma, x : A \vdash e \Leftarrow B}{\Gamma \vdash \lambda x.e \Leftarrow A \to B} \qquad \frac{\Gamma \vdash f \Rightarrow A \to B \quad \Gamma \vdash a \Leftarrow A}{\Gamma \vdash f\ a \Rightarrow B}$$

$$\frac{\Gamma \vdash a \Leftarrow A \quad \Gamma \vdash b \Leftarrow B}{\Gamma \vdash (a, b) \Leftarrow A \times B} \qquad \frac{\Gamma \vdash e \Rightarrow A \times B}{\Gamma \vdash \texttt{outl}\ e \Rightarrow A} \qquad \frac{\Gamma \vdash e \Rightarrow A \times B}{\Gamma \vdash \texttt{outr}\ e \Rightarrow B}$$

$$\frac{\Gamma \vdash e \Leftarrow A}{\Gamma \vdash \texttt{inl}\ e \Leftarrow A + B} \qquad \frac{\Gamma \vdash e \Leftarrow B}{\Gamma \vdash \texttt{inr}\ e \Leftarrow A + B}$$

$$\frac{\Gamma \vdash e \Rightarrow A + B \quad \Gamma \vdash f \Leftarrow A \to C \quad \Gamma \vdash g \Leftarrow B \to C}{\Gamma \vdash \texttt{case}\ e\ \texttt{of}\ (f, g) \Leftarrow C}$$

$$\frac{}{\Gamma \vdash \texttt{unit} \Leftarrow \mathbf{1}} \qquad \frac{\Gamma \vdash e \Leftarrow \mathbf{0}}{\Gamma \vdash \texttt{exfalso}\ e \Leftarrow A}$$

Bidirectional typing – additional rules

$$\frac{}{\Gamma \vdash \texttt{unit} \Rightarrow \mathbf{1}}$$

$$\frac{\Gamma \vdash e \Rightarrow A + B \quad \Gamma \vdash f \Rightarrow A \to C \quad \Gamma \vdash g \Rightarrow B \to C}{\Gamma \vdash \texttt{case}\ e\ \texttt{of}\ (f, g) \Rightarrow C}$$

$$\frac{\Gamma \vdash a \Rightarrow A \quad \Gamma \vdash b \Rightarrow B}{\Gamma \vdash (a, b) \Rightarrow A \times B}$$

The basic rules are as presented in the previous slide. However, it is possible to add some enhancements. First, we can replace the rule for `unit` with an inference rulem and if we need to check, we can use subsumption. Second, the paper argues that sum elimination is a general principle, and so it should allow both checking and inference versions. We could also add an inference rule for pairs. It seems there isn't a trade-off either – if the checking rule fails, we can use subsumption and try to infer.

Metatheory

Similarly to Extrinsic STLC, typing is not unique in Bidirectional STLC. This is because while we do have annotations in terms, we are not forced to use them. Therefore, we can check terms like $\lambda x.x$ with any type of the form $A \to A$. However, inference is unique.

Intrinsic STLC

Intrinsic STLC is the most widespread version of typed lambda calculus. The terms differ from Extrinsic STLC, as type annotations are mandatory on lambdas, sum constructors and the empty type eliminator.

Terms

Note: red color marks places which differ from Extrinsic STLC.

Terms:

$e ::=$

$x \mid$

$\lambda x : A.e \mid e_1\ e_2 \mid$

$(e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid$

$\texttt{inl}_A\ e \mid \texttt{inr}_A\ e \mid \texttt{case}\ e\ \texttt{of}\ (e_1, e_2) \mid$

$\texttt{unit} \mid \texttt{exfalso}_A\ e$

Declarative typing – differences

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x : A.e : A \to B}$$

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathtt{inl}_B\ e : A + B} \qquad \frac{\Gamma \vdash e : B}{\Gamma \vdash \mathtt{inr}_A\ e : A + B}$$

$$\frac{\Gamma \vdash e : \mathbf{0}}{\Gamma \vdash \mathtt{exfalso}_A\ e : A}$$

Algorithmic typing – basics

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \Rightarrow A}\textsc{Var}$$

Algorithmic typing – type-directed rules

$$\frac{\Gamma, x : A \vdash e \Rightarrow B}{\Gamma \vdash \lambda x : A.e \Rightarrow A \to B} \qquad \frac{\Gamma \vdash f \Rightarrow A \to B \quad \Gamma \vdash a \Rightarrow A}{\Gamma \vdash f\ a \Rightarrow B}$$

$$\frac{\Gamma \vdash a \Rightarrow A \quad \Gamma \vdash b \Rightarrow B}{\Gamma \vdash (a, b) \Rightarrow A \times B} \qquad \frac{\Gamma \vdash e \Rightarrow A \times B}{\Gamma \vdash \texttt{outl}\ e \Rightarrow A} \qquad \frac{\Gamma \vdash e \Rightarrow A \times B}{\Gamma \vdash \texttt{outr}\ e \Rightarrow B}$$

$$\frac{\Gamma \vdash e \Rightarrow A}{\Gamma \vdash \texttt{inl}_B\ e \Rightarrow A + B} \qquad \frac{\Gamma \vdash e \Rightarrow B}{\Gamma \vdash \texttt{inr}_A\ e \Rightarrow A + B}$$

$$\frac{\Gamma \vdash e \Rightarrow A + B \quad \Gamma \vdash f \Rightarrow A \to C \quad \Gamma \vdash g \Rightarrow B \to C}{\Gamma \vdash \texttt{case}\ e\ \texttt{of}\ (f, g) \Rightarrow C}$$

$$\frac{}{\Gamma \vdash \texttt{unit} \Rightarrow \mathbf{1}} \qquad \frac{\Gamma \vdash e \Rightarrow \mathbf{0}}{\Gamma \vdash \texttt{exfalso}_A\ e \Rightarrow A}$$

Metatheory

Because of the annotations on lambda, sum constructors and exfalso, Intrinsic STLC does enjoy uniqueness of typing. It is easy to prove this by induction: types of most terms are determined by the induction hypothesis, whereas for the aforementioned four we need to supplement the induction hypothesis with the annotation.

Dual Intrinsic STLC

If the usual Intrinsic STLC turns out to be a system in which we can infer all types, what would a system in which we can check all types look like?

Terms

Note: red color marks places which differ from Extrinsic STLC.

Terms:

$e ::=$

$x \mid$

$\lambda x.e \mid \mathtt{app}_A\ e_1\ e_2 \mid$

$(e_1, e_2) \mid \mathtt{outl}_A\ e \mid \mathtt{outr}_A\ e \mid$

$\mathtt{inl}\ e \mid \mathtt{inr}\ e \mid \mathtt{case}_{A,B}\ e\ \mathtt{of}\ (e_1, e_2) \mid$

$\mathtt{unit} \mid \mathtt{exfalso}\ e$

Declarative typing – differences

$$\frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash \mathtt{app}_A\ f\ a : B}$$

$$\frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \mathtt{outl}_B\ e : A} \qquad \frac{\Gamma \vdash e : A \times B}{\Gamma \vdash \mathtt{outr}_A\ e : B}$$

$$\frac{\Gamma \vdash e : A + B \quad \Gamma \vdash f : A \to C \quad \Gamma \vdash g : B \to C}{\Gamma \vdash \mathtt{case}_{A,B}\ e\ \mathtt{of}\ (f, g) : C}$$

Algorithmic typing – basics

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \Leftarrow A}\textsc{Var}$$

Algorithmic typing – type-directed rules

$$\frac{\Gamma, x : A \vdash e \Leftarrow B}{\Gamma \vdash \lambda x.e \Leftarrow A \to B} \qquad \frac{\Gamma \vdash f \Leftarrow A \to B \quad \Gamma \vdash a \Leftarrow A}{\Gamma \vdash \mathtt{app}_A\ f\ a \Leftarrow B}$$

$$\frac{\Gamma \vdash a \Leftarrow A \quad \Gamma \vdash b \Leftarrow B}{\Gamma \vdash (a, b) \Leftarrow A \times B} \qquad \frac{\Gamma \vdash e \Leftarrow A \times B}{\Gamma \vdash \mathtt{outl}_B\ e \Leftarrow A} \qquad \frac{\Gamma \vdash e \Leftarrow A \times B}{\Gamma \vdash \mathtt{outr}_A\ e \Leftarrow B}$$

$$\frac{\Gamma \vdash e \Leftarrow A}{\Gamma \vdash \mathtt{inl}\ e \Leftarrow A + B} \qquad \frac{\Gamma \vdash e \Leftarrow B}{\Gamma \vdash \mathtt{inr}\ e \Leftarrow A + B}$$

$$\frac{\Gamma \vdash e \Leftarrow A + B \quad \Gamma \vdash f \Leftarrow A \to C \quad \Gamma \vdash g \Leftarrow B \to C}{\Gamma \vdash \mathtt{case}_{A,B}\ e\ \mathtt{of}\ (f, g) \Leftarrow C}$$

$$\frac{}{\Gamma \vdash \mathtt{unit} \Leftarrow \mathbf{1}} \qquad \frac{\Gamma \vdash e \Leftarrow \mathbf{0}}{\Gamma \vdash \mathtt{exfalso}\ e \Leftarrow A}$$

Metatheory

Even though Dual Intrinsic STLC has plenty of mandatory annotations, it does not enjoy uniqueness of typing for the usual reasons: we can type $\lambda x.x$ with any type of the form $A \to A$. The role of the annotations is not to force types to be unique, but to make it possible to implement type checking.

Hints

We introduce hints, which will be useful to describe two more variants of STLC.

$$H ::= ? \mid H_1 \to H_2 \mid H_1 \times H_2 \mid H_1 + H_2 \mid \mathbf{1} \mid \mathbf{0}$$

Intuitively, hints are partial types. They are built like types, except that there's one additional way to make hints: ?, which can be read as “unknown”, “hole” or “whatever”.

We use the letter H for hints. When we use letters like A, B, C which usually stand in for types, it means that the hint **is** a type, i.e. it doesn't contain any ?s.

Combining hints

$$
\begin{gathered}
? \sqcap H = H \\
H \sqcap ? = H \\
(H_1 \to H_2) \sqcap (H_1' \to H_2') = (H_1 \sqcap H_1') \to (H_2 \sqcap H_2') \\
(H_1 \times H_2) \sqcap (H_1' \times H_2') = (H_1 \sqcap H_1') \times (H_2 \sqcap H_2') \\
(H_1 + H_2) \sqcap (H_1' + H_2') = (H_1 \sqcap H_1') + (H_2 \sqcap H_2') \\
\mathbf{1} \sqcap \mathbf{1} = \mathbf{1} \\
\mathbf{0} \sqcap \mathbf{0} = \mathbf{0}
\end{gathered}
$$

Intuitively, $\sqcap$ is a partial binary operation which combines two hints which share the same structure, possibly filling the ?s in the leaves with something more informative coming from the other argument. For hints which have incompatible structure (e.g. function hint and product hint) the result is undefined.

Combining hints – properties

The operation $\sqcap$:

- Is associative: $(H_1 \sqcap H_2) \sqcap H_3 = H_1 \sqcap (H_2 \sqcap H_3)$
- Is commutative, i.e. $H_1 \sqcap H_2 = H_2 \sqcap H_1$
- Has neutral element: $? \sqcap H = H = H \sqcap ?$
- Is idempotent, i.e. $H \sqcap H = H$.

So, when $\sqcap$ happens to defined at all, it forms a commutative idempotent monoid. And as we all know, partial monoids are categories. So it seems hints form a thin groupoid... which they obviously do!

Order on hints

The operation $\sqcap$ induces an order on hints in the usual way: $H_1 \sqsubseteq H_2 :\equiv (H_1 \sqcap H_2 = H_2)$.

This order can be intuitively interpreted as an information increase: $H_1 \sqsubseteq H_2$ means that hint H_2 is more informative than H_1, but in a compatible way. In other words, H_1 and H_2 have the same structure, but some ?s from H_1 can be refined to something more informative in H_2.

Order on hints – explicitly

$$\frac{}{? \sqsubseteq H}$$

$$\frac{H_1 \sqsubseteq H_1' \quad H_2 \sqsubseteq H_2'}{H_1 \to H_2 \sqsubseteq H_1' \to H_2'}$$

$$\frac{H_1 \sqsubseteq H_1' \quad H_2 \sqsubseteq H_2'}{H_1 \times H_2 \sqsubseteq H_1' \times H_2'}$$

$$\frac{H_1 \sqsubseteq H_1' \quad H_2 \sqsubseteq H_2'}{H_1 + H_2 \sqsubseteq H_1' + H_2'}$$

$$\frac{}{\mathbf{1} \sqsubseteq \mathbf{1}} \qquad \frac{}{\mathbf{0} \sqsubseteq \mathbf{0}}$$

Hinting v1

STLC with hints (version 1) is a flavour of STLC inspired by Bidirectional STLC. The main insight behind it is that in Bidirectional STLC, we have a hard time deciding whether a rule should be in checking mode or in inference mode, so why not both? This way, we would have some input type that guides us, but also produce an output type, which is in some sense "better". This is a bit silly if we already have the type we need as input, but we can make this idea work by introducing hints, which are types with holes, and insisting that the input is not a type, but merely a hint.

Terms

Note: red color marks differences from Bidirectional STLC.

Terms:

$$
\begin{aligned}
e ::= \;& \\
& x \mid (e : H) \mid \\
& \lambda x.e \mid e_1\ e_2 \mid \\
& (e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid \\
& \texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (e_1, e_2) \mid \\
& \texttt{unit} \mid \texttt{exfalso}\ e
\end{aligned}
$$

Typing contexts assign types to variables, but annotations in terms are hints, not necessarily types.

Hinting – basic rules

$$\frac{(x : A) \in \Gamma \quad H \sqsubseteq A}{\Gamma \vdash x \Leftarrow H \Rightarrow A}\textsc{Var}$$

$$\frac{\Gamma \vdash e \Leftarrow H_1 \sqcap H_2 \Rightarrow A}{\Gamma \vdash (e : H_1) \Leftarrow H_2 \Rightarrow A}\textsc{Annot}$$

Hinting – type-directed rules 1

$$\frac{H \sqcap (? \to ?) = A \to H' \quad \Gamma, x : A \vdash e \Leftarrow H' \Rightarrow B}{\Gamma \vdash \lambda x.e \Leftarrow H \Rightarrow A \to B}$$

$$\frac{\Gamma \vdash f \Leftarrow ? \to H \Rightarrow H' \to B \quad \Gamma \vdash a \Leftarrow H' \Rightarrow A}{\Gamma \vdash f\ a \Leftarrow H \Rightarrow B}$$

$$\frac{H \sqcap (? \times ?) = H_1 \times H_2 \quad \Gamma \vdash a \Leftarrow H_1 \Rightarrow A \quad \Gamma \vdash b \Leftarrow H_2 \Rightarrow B}{\Gamma \vdash (a, b) \Leftarrow H \Rightarrow A \times B}$$

$$\frac{\Gamma \vdash e \Leftarrow H \times ? \Rightarrow A \times B}{\Gamma \vdash \texttt{outl}\ e \Leftarrow H \Rightarrow A} \qquad \frac{\Gamma \vdash e \Leftarrow H \times ? \Rightarrow A \times B}{\Gamma \vdash \texttt{outr}\ e \Leftarrow H \Rightarrow B}$$

Hinting – type-directed rules 2

$$\frac{H \sqcap (? + ?) = H' + B \quad \Gamma \vdash e \Leftarrow H' \Rightarrow A}{\Gamma \vdash \texttt{inl}\ e \Leftarrow H \Rightarrow A + B}$$

$$\frac{H \sqcap (? + ?) = A + H' \quad \Gamma \vdash e \Leftarrow H' \Rightarrow B}{\Gamma \vdash \texttt{inr}\ e \Leftarrow H \Rightarrow A + B}$$

$$\frac{\Gamma \vdash e \Leftarrow ? + ? \Rightarrow A + B \quad \begin{array}{c} \Gamma \vdash f \Leftarrow A \to H \Rightarrow A \to C \\ \Gamma \vdash g \Leftarrow B \to H \Rightarrow B \to C \end{array}}{\Gamma \vdash \mathsf{case}\ e\ \mathsf{of}\ (f, g) \Leftarrow H \Rightarrow C}$$

$$\frac{H \sqsubseteq \mathbf{1}}{\Gamma \vdash \texttt{unit} \Leftarrow H \Rightarrow \mathbf{1}} \qquad \frac{\Gamma \vdash e \Leftarrow \mathbf{0} \Rightarrow \mathbf{0}}{\Gamma \vdash \texttt{exfalso}\ e \Leftarrow A \Rightarrow A}$$

Metatheory

Similarly to Extrinsic STLC, STLC with hints v1 does not enjoy uniqueness of typing. This is because we still can have terms like $\lambda x.x$ with hint ?, which can be typed with any type of the form $A \to A$. However, if the hint is informative enough, then the type is unique. Moreover, every typable term can be given a hint which makes its type unique.

Hinting v2

STLC with hints (version 2) is a slight variation on the first one. The problem is that the rules of v1 look ugly, because we need to do so much management of the hints. To deal with this, we relegate all this work into a single separate rule (which uses a new operation), making all the remaining ones much more readable. The terms are the same as in STLC with hints v1.

Hinting v2 – hints for term constructors

$$
\begin{gathered}
H \bigtriangleup \lambda x.e = H \sqcap (? \to ?) \\
H \bigtriangleup (e_1, e_2) = H \sqcap (? \times ?) \\
H \bigtriangleup \texttt{inl}\ e = H \sqcap (? + ?) \\
H \bigtriangleup \texttt{inr}\ e = H \sqcap (? + ?) \\
H \bigtriangleup \texttt{unit} = H \sqcap \mathbf{1}
\end{gathered}
$$

Hinting v2 – basic rules

$$\frac{(x : A) \in \Gamma \quad H \sqsubseteq A}{\Gamma \vdash x \Leftarrow H \Rightarrow A}\textsc{Var}$$

$$\frac{\Gamma \vdash e \Leftarrow H_1 \sqcap H_2 \Rightarrow A}{\Gamma \vdash (e : H_1) \Leftarrow H_2 \Rightarrow A}\textsc{Annot}$$

$$\frac{e\ \texttt{constructor} \quad H \triangle e = H' \quad H \neq H' \quad \Gamma \vdash e \Leftarrow H' \Rightarrow A}{\Gamma \vdash e \Leftarrow H \Rightarrow A}\textsc{HintFor}$$

Hinting v2 – type-directed rules

$$\frac{\Gamma, x : A \vdash e \Leftarrow H \Rightarrow B}{\Gamma \vdash \lambda x.e \Leftarrow A \to H \Rightarrow A \to B}$$

$$\frac{\Gamma \vdash f \Leftarrow ? \to H \Rightarrow H' \to B \quad \Gamma \vdash a \Leftarrow H' \Rightarrow A}{\Gamma \vdash f\ a \Leftarrow H \Rightarrow B}$$

$$\frac{\Gamma \vdash a \Leftarrow H_1 \Rightarrow A \quad \Gamma \vdash b \Leftarrow H_2 \Rightarrow B}{\Gamma \vdash (a, b) \Leftarrow H_1 \times H_2 \Rightarrow A \times B}$$

$$\frac{\Gamma \vdash e \Leftarrow H \times ? \Rightarrow A \times B}{\Gamma \vdash \texttt{outl}\ e \Leftarrow H \Rightarrow A} \qquad \frac{\Gamma \vdash e \Leftarrow H \times ? \Rightarrow A \times B}{\Gamma \vdash \texttt{outr}\ e \Leftarrow H \Rightarrow B}$$

Hinting v2 – type-directed rules

$$\frac{\Gamma \vdash e \Leftarrow H \Rightarrow A}{\Gamma \vdash \texttt{inl}\ e \Leftarrow H + B \Rightarrow A + B}$$

$$\frac{\Gamma \vdash e \Leftarrow H \Rightarrow B}{\Gamma \vdash \texttt{inr}\ e \Leftarrow A + H \Rightarrow A + B}$$

$$\frac{\Gamma \vdash e \Leftarrow ? + ? \Rightarrow A + B \qquad \begin{array}{c}\Gamma \vdash f \Leftarrow A \to H \Rightarrow A \to C \\ \Gamma \vdash g \Leftarrow B \to H \Rightarrow B \to C\end{array}}{\Gamma \vdash \texttt{case}\ e\ \texttt{of}\ (f, g) \Leftarrow H \Rightarrow C}$$

$$\frac{}{\Gamma \vdash \texttt{unit} \Leftarrow \mathbf{1} \Rightarrow \mathbf{1}} \qquad \frac{\Gamma \vdash e \Leftarrow \mathbf{0} \Rightarrow \mathbf{0}}{\Gamma \vdash \texttt{exfalso}\ e \Leftarrow A \Rightarrow A}$$

Metatheory

Metatheoretically, STLC with hints v2 is similar to v1.