Recitation 8: Bidirectional Typechecking and Sequent Calculus

Jon Sterling

1 Bidirectional Typechecking

The proof term assignment for the logic of verifications and uses can be construed as an algorithmic specification for a typechecker; this is because the rules of this logic are all syntax-directed.

To expose the algorithmic character of the judgments, we write in blue the inputs of a judgment and in red the outputs of a judgment.

\[
\begin{align*}
() : \top & \uparrow \triangledown I \\
R : A \land B & \downarrow \land E_1 \\
\text{fst}(R) : A & \downarrow \\
R : A \land B & \downarrow \land E_2 \\
snd(R) : B & \downarrow \\
M : A \uparrow & \land 1 \\
(M, N) : A \land B & \uparrow \\
N : C & \uparrow \\
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\frac{M : B \uparrow}{\text{inr}(M) : A \lor B \uparrow} & \land 2 \\
& \vdots \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
R : A \lor B & \downarrow \\
N_1 : C & \uparrow \\
N_2 : C & \uparrow \\
\text{case } R \text{ of } \text{inl}(u) & \Rightarrow \{ N_1 \mid \text{inr}(v) \Rightarrow N_2 \} & \ort E_{u,v}
\end{align*}
\]

\[
\begin{align*}
u : A & \downarrow \lor I_u \\
v : B & \downarrow \lor I_v
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\frac{R : A \supset B \downarrow}{(R(N)) : B \downarrow} & \supset E \\
\frac{N : A \supset B \uparrow}{(M : A) : A \downarrow} & \text{ann}
\end{array}
\end{align*}
\]

If the last rule ann is omitted, then the terms of the verifications and uses calculus will have no redexes; if it is added, then redexes can be formed, and the calculus is merely a more verbose version of the ordinary λ-calculus. This week, we are studying a version of verifications and uses with the ann rule, because it is necessary in order to give a natural proof term assignment to Dyckhoff’s contraction-free sequent calculus.
the algorithm  We will use $\Delta \vdash \mathcal{J}$ to abbreviate $\mathcal{J}$ in what follows, where $\Delta$ is a sequence of assumptions $x : A \downarrow$. The typechecking algorithm for bidirectional type-checking is the computational content of the following (constructive) metatheorem:

1. For all $\Delta, N, A$, either $\Delta \vdash N : A \, \uparrow$ or not.

2. For all $\Delta, R$, either there exists some $A$ such that $\Delta \vdash R : A \, \downarrow$ or there does not.

Exercise. Try and remember how this proof works. You will have to extract its algorithmic content in order to complete the next homework assignment.

Remark. This metatheorem is only interesting if the ambient mathematics is constructive; otherwise, it is trivial and its proof provides no useful algorithmic content.

2 Proof terms for sequent calculus

We can give a proof term assignment to the sequent calculus based on a single form of judgment $\Gamma \Rightarrow N : A$; in this form of judgment, $\Gamma$ is now a list of formal type assignments $R : A$, where $R$ is an arbitrary neutral term rather than only a variable. The dynamics are as follows:

1. The list of type assignments $\Gamma \equiv R : A$ are inputs.

2. The goal type $A$ is an input.

3. The proof term / witness $N$ is an output.

These dynamics correspond to proof refinement with extraction in sequent-calculus-based proof systems like Nuprl and RedPRL.

$$\begin{align*}
\Gamma, R : A &\Rightarrow R : A \quad \text{init} \\
\Gamma &\Rightarrow () : \top & \Gamma R &\Rightarrow \text{abort}(R) : C & \bot \\
\Gamma &\Rightarrow N_1 : A & \Gamma &\Rightarrow (N_1, N_2) : A \land B & \land R \\
\Gamma, R : A \land B &\Rightarrow N : C & \Gamma, R : A \land B &\Rightarrow N : C & \land L_1 \\
\Gamma, R : A \land B, \text{snd}(R) : B &\Rightarrow N : C & \Gamma, R : A \land B &\Rightarrow N : C & \land L_2 \\
\Gamma &\Rightarrow N : B & \Gamma &\Rightarrow \text{inl}(N) : A \lor B & \lor R_1 \\
\Gamma &\Rightarrow N : B & \Gamma &\Rightarrow \text{inr}(N) : A \lor B & \lor R_2 \\
\Gamma, R : A \lor B, u : A &\Rightarrow N_1 : C & \Gamma, R : A \lor B, v : A &\Rightarrow N_2 : C & \lor \text{L}^{u,v}
\end{align*}$$
Exercise. In sequent calculus, we try to prove the admissibility of the following cut rule:

\[
\frac{\Gamma, R : A \supset B \Rightarrow M : A \quad \Gamma, R : A \supset B, R(M) : B \Rightarrow N : C}{\Gamma, R : A \supset B \Rightarrow N : C} \quad \text{cut}
\]

Try to invent a proof term assignment for this rule.

Solution.

\[
\frac{\Gamma \Rightarrow M : A \quad \Gamma, (M : A) : A \Rightarrow N : B}{\Gamma \Rightarrow N : B} \quad \text{cut}_A
\]