Assignment 7:
Adventures in Subtyping

15-312: Foundations of Programming Languages
Joshua Dunfield (joshuad@cs.cmu.edu)

Out: Thursday, November 7, 2002
Due: Thursday, November 14, 2002 (1:30 pm)

50 points total + 15 points extra credit

Subtyping products (15 points)

The subtyping rule for products

$$\frac{\sigma_1 \leq \tau_1 \qquad \sigma_2 \leq \tau_2}{\sigma_1 * \sigma_2 \leq \tau_1 * \tau_2} \ (*)$$

is analogous to the rule for "depth subtyping" in records (shown in lecture). Suppose we added rules corresponding to "width subtyping", as follows:

$$\frac{\sigma_1 \leq \tau}{\sigma_1 * \sigma_2 \leq \tau} \ (*\text{Width1}) \qquad \frac{\sigma_2 \leq \tau}{\sigma_1 * \sigma_2 \leq \tau} \ (*\text{Width2})$$

Question 1 (5 points). What's wrong with this? Explain in (at most) a few sentences.

Question 2 (10 points). Suppose we dropped the second "width" rule, ($*$Width2), leaving only

$$\frac{\sigma_1 \leq \tau}{\sigma_1 * \sigma_2 \leq \tau} \ (*\text{Width1})$$

Is there a problem? Explain in (at most) a few sentences.

Demonic nondeterminism (35 points)

Consider a *demonic nondeterministic choice* operator $\oplus$. To evaluate $e_1 \oplus e_2$, choose either e_1 or e_2 at random, then evaluate the chosen expression. For example,

$$f(x) \oplus 0$$

will evaluate either to the result of evaluating $f(x)$, or to 0. Note that if either expression does not terminate, the whole expression potentially does not terminate.[1]

The dynamic semantics of this construct is very simple:

$$\frac{}{e_1 \oplus e_2 \mapsto e_1}\ (\oplus\text{L}) \qquad \frac{}{e_1 \oplus e_2 \mapsto e_2}\ (\oplus\text{R})$$

Applying either of these rules eliminates the $\oplus$, and evaluation proceeds normally. Clearly, this semantics is not deterministic. If we add a requirement that whenever ($\oplus$L) and ($\oplus$R) could both be applied, each is chosen with probability $1/2$, we obtain a source of random bits ($\mathsf{true} \oplus \mathsf{false}$). However, for this assignment the choice between ($\oplus$L) and ($\oplus$R) may be considered to be arbitrary.

The typing rule for $\oplus$ is in the style of our typing rules for **if** and **case**:

$$\frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \oplus e_2 : \tau}\ (\oplus\text{Type})$$

But we might wish to allow expressions such as $(x + 2) \oplus \mathsf{true}$. If the program is well-typed whether this expression has type int or type bool, this is perfectly sensible (if unconventional!). One can model this with *union types*, in which $\sigma \vee \tau$ is the type of values that have type σ, or type τ, but it is unknown which.

In the subset interpretation of subtyping, the following subtyping rules for union types $\vee$ are easily justified. For instance, rule ($\vee R1$) is justified by the set-theoretic statement

If $A \subseteq B_1$ then $A \subseteq B_1 \cup B_2$.

$$\frac{\sigma \leq \tau_1}{\sigma \leq \tau_1 \vee \tau_2}\ (\vee R1) \qquad \frac{\sigma \leq \tau_2}{\sigma \leq \tau_1 \vee \tau_2}\ (\vee R2) \qquad \frac{\sigma_1 \leq \tau \qquad \sigma_2 \leq \tau}{\sigma_1 \vee \sigma_2 \leq \tau}\ (\vee L)$$

Of course, for the subtyping rules to be useful, we need the subsumption rule

[1] In contrast, "angelic" nondeterministic choice operators evaluate the arguments in parallel, and therefore terminate if *either* of the arguments terminate. For this terminology—attributed to Hoare—to make sense, one has to believe that nontermination is morally suspect.

$$\frac{\Gamma \vdash e : \sigma \qquad \sigma \leq \tau}{\Gamma \vdash e : \tau} \ (\textit{Sub})$$

Then $(x + 2) \oplus \mathsf{true}$ is well-typed if we use subsumption to show $(x + 2) : \mathsf{int} \vee \mathsf{bool}$ and $\mathsf{true} : \mathsf{int} \vee \mathsf{bool}$, and then use ($\oplus$Type).

Thus, the resulting type system includes the rules for the usual MinML constructs[2], plus (*Sub*) and ($\oplus$Type). The subtyping system has the usual rules plus ($\vee R1$), ($\vee R2$), and ($\vee L$).

The proof of preservation for this system is by rule induction on the given derivations of *both* judgments: $e \mapsto e'$ and $\cdot \vdash e : \tau$. Thus, you can apply the IH if (1) each of the two derivations you are applying it to is either one of the given derivations or smaller than (e.g. a subderivation of) a given derivation, and (2) at least one is strictly smaller than a given derivation.

Question 3 (15 points). Do the cases of the proof of preservation corresponding to the new rules, ($\oplus$L) and ($\oplus$R). Show each proof step. (If the cases turn out to be symmetric, just say so and omit one of them.) If you need a lemma such as value inversion, substitution, weakening, etc. you may omit the lemma's proof if you state it clearly, but be sure it's correct!

Question 4 (10 points). In a bidirectional system (cf. Assignment 6), what rule(s) should replace the non-bidirectional rule ($\oplus$Type)? In a sentence or two, justify the correctness of your rule(s). There may be more than one "right" answer.

Question 5 (10 points). **Coercion interpretation:** Take the subtyping rules ($\vee$R1), ($\vee$R2), ($\vee$L) and add appropriate coercions. Your rules should be *coherent* in the sense discussed in class, but you don't need to prove this. **Hint:** Think about the coercion interpretation of intersection types.

Question 6 (15 points EXTRA CREDIT). Is this rule a good one?

$$\frac{\Gamma \vdash e' : \sigma_1 \vee \sigma_2 \qquad \Gamma, x{:}\sigma_1 \vdash e : \tau \qquad \Gamma, x{:}\sigma_2 \vdash e : \tau}{\Gamma \vdash \{e'/x\}e : \tau} \ (?)$$

Support your answer. Note that it is assumed that x is completely fresh, that is, does not appear anywhere in Γ or e. (As usual for extra credit questions, some credit will be given for any thoughtful answer. Simply demonstrating a general understanding of how the rule works, without arguing whether it is good or bad, is worth something! You may consult any source, such as Pierce's book, to answer this question—provided you cite it properly!)

[2]Whether we happen to have pairs, sums, etc. doesn't really matter; only the new $\oplus$ construct and $\vee$ type are of interest.