Resource Analysis
Lecture 4

June 22, 2019

1 Recap: Soundness

Theorem 1.1 (Progress). *If* $\vdash^{q}_{q'} e : \tau$ *and* $p \geq q$ *then either* e *is a value or* $\exists e', p'$ *s.t.* $\langle e, p\rangle \mapsto \langle e', p'\rangle$.

Theorem 1.2 (Preservation). *If* $\vdash^{q}_{q'} e : \tau$, $p \geq q$ *and* $\langle e, p\rangle \mapsto \langle e', p'\rangle$ *then* $\vdash^{p'}_{q'} e' : \tau$.

Proof notes. Proved by nested induction on $\vdash^{q}_{q'} e : \tau$ and $\langle e, p\rangle \mapsto \langle e', p'\rangle$. □

Alternative soundness theorem: Recall the judgement $V \vdash e \Downarrow v \mid (q, q')$

Definition 1.2.1. $\phi(V : \Gamma) = \sum_{x \in dom(\Gamma)} \phi(V(x) : \Gamma(x))$

where Γ assigns types to variables.

Theorem 1.3. *Let* $V : \Gamma$ *and* $\Gamma \vdash^{q}_{q'} e : \tau$ *and* $V \vdash e \Downarrow v \mid (p, p')$ *then* $\phi(V : \Gamma) + q \geq p$ *and* $\phi(V : \Gamma) + q - \phi(v : \tau) - q' \geq p - p'$

This theorem shows that the type derivation is a certificate for bound correctness.

2 Type inference

Example 1. *We want to find a derivation for:*

$$\vdash^{0}_{0} fix\big(id.\ \lambda(x : L(unit))\ matL(x;\ nil;\ y, ys.cons(y;\ tick\{2\}(id(ys))))\big) : L^2(unit) \to^{0/0} L^0(unit)$$

See figure 1 for the deriviation tree, where $e_{id} = fix\big(id.\ \lambda(x : L(unit))\ matL(x;\ nil;\ y, ys.cons(y;\ tick\{2\}(id$ and $\tau_{id} = L^2(unit) \to^{0/0} L^0(unit)$.

For type inference we need algorithmic (or syntax-directed) rules. We change all the typing rules to incorporate the structural rules:

Example 2. $$\frac{q \geq q' \qquad \tau <: \tau'}{\Gamma, x : \tau \vdash^{q}_{q'} x : \tau'}$$

Algorithm for type inference:

1. Infer usual tpyes (without annotations), which results in a type derivation (like example in figure 1 with all annotations removed);

2. Add potential variables where a potential annotation is required, both in τ_{id} and in the derivation. See figure 2. Note the this step is only partially done in the figure;

$$\tau_{id} \text{ becomes } L^p(unit) \to^{q/q'} L^{p'}(unit)$$

3. Derive from the typing rules linear constraints on potential variables;

   Example 3. *For the fix example, some of the constraints are:*

$$\begin{array}{rcccc} r_0 \geq r_0' & r_2 \leq q & p_1 \leq p & r_3 \geq r_3' & p_2 \leq p_1 \\ r_1 = 0 & r_2' \geq q' & p_1' \geq p' & r_3 \leq r_2 & p_2' \geq p_2 \\ r_1' = 0 & & & & \\ r_4 \leq r_3' + p_1 & r_4' < r_2' & & & \\ s_1 \geq s_2 + 2 & s_1' = s_2' & & & \end{array}$$

4. Solve constraints with LP solver;

5. Objective is the sum of initial potential annotations.

   Example 4. *For the fix example the objective is to minimize $p + q$.*

3 Implementation in RAML and examples

Live RAML (Resource aware ML) demo showing binary counter, using the source code displayed in figure 3. Second example with queue.

$$
\dfrac{\dfrac{\dfrac{x : L^2(1) \vdash^{0}_{0} x : L^2(1) \qquad id : \tau_{id} \vdash^{0}_{0} nil : L^0(1) \qquad \dfrac{\text{relax}\,\dfrac{}{y : 1 \vdash^{2}_{2} y : 1} \qquad \dfrac{\text{app}\,\dfrac{}{id : \tau_{id}, ys : L^2(1) \vdash^{0}_{0} id(ys) : L^0(1)}}{id : \tau_{id}, ys : L^2(1) \vdash^{2}_{0} tick\{2\}(id(ys)) : L^0(1)}}{id : \tau_{id}, y : 1, ys :: L^2(1) \vdash^{2}_{0} cons(y, tick\{2\}(id(ys)) : L^0(1)}}{id : \tau_{id}, x : L^2(1) \vdash e_{id} : L^0(1)}}{id : \tau_{id} \vdash^{0}_{0} \lambda(x)e_{id} : \tau_{id}}}{\vdash^{0}_{0} fix(id\lambda(x)e_{id} : \tau_{id}}
$$

Figure 1: Example of type inference

$$
\dfrac{
x : L^{p_2}(1) \vdash^{r_3}_{r_3'} x : L^{p_2'}(1)
\qquad
id : \tau_{id} \vdash nil : L(1)
\qquad
\dfrac{
\text{relax}\,\dfrac{}{y : 1 \vdash y : 1}
\qquad
\dfrac{
\text{app}\,\dfrac{}{id : \tau_{id}, ys : L(1) \vdash^{s_2}_{s_2'} id(ys) : L(1)}
}{id : \tau_{id}, ys : L(1) \vdash^{s_1}_{s_1'} tick\{2\}(id(ys)) : L(1)}
}{id : \tau_{id}, y : 1, ys :: L^2(1) \vdash^{r_4}_{r_4'} cons(y, tick\{2\}(id(ys)) : L(1)}
}{
\dfrac{
\dfrac{id : \tau_{id}, x : L^{p_1}(1) \vdash^{r_3}_{r_3'} e_{id} : L^{p_1'}(1)}{id : \tau_{id} \vdash^{r_1}_{r_1'} \lambda(x) e_{id} : \tau_{id}}
}{\vdash^{r_0}_{r_0'} fix(id\lambda(x)e_{id} : \tau_{id}}
}
$$

Figure 2: Type inference, step 2 of the algorithm

```
let rec id x =
    match x with
    | [] -> []
    | y::ys -> y::(let _ = Raml.tick 2.0 in ys)

type bit = Zero | One

let rec inc counter =
    match counter with
    | [] -> [One]
    | Zero::bs -> One::bs
    | One::bs -> Zero::(inc bs)

let rec in_many n =
    match n with
    | Z -> []
    | S n'-> inc (inc_many n')
```

Figure 3: Code for binary counter example