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 $\langle e, p \rangle \mapsto \langle e', p' \rangle$. $\qquad\square$

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

From the typing rules defined, it follows that:

$\vdash fix(id.\lambda(x : L(unit))matL(X, nil, y, ys, icons(y, tick\{2\}if(ys)))) : L^2(unit) \to^{0/0} L^0(unit)$

See figure 2 for the deriviation tree.
For type inference we need algorithmic (or syntax-directed) rules.

Example 1. $\dfrac{q \geq q' \qquad \tau <: \tau'}{\Gamma, x : \tau \vdash^q_{q'} x : \tau}$

Algorithm for type inference:

1. Infer usual tpyes (without annotations), which results in a type derication (like example in figure 2 with all annotations removed);

2. Add potential variables where a potential annotation is required;

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

4. Solve constraints with LP solver;

5. Objective is the sum of initial potential annotations.

3   Implementation in RAML and examples

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

$$\dfrac{\text{app}\ \dfrac{id : \tau_{id}, ys : L^2(1) \mid_0^0 id(us) : L^0(1)}{\dfrac{id : \tau_{id}, yss : L^2(1) \mid_0^2 tick\{2\}(id(ys)) : L^0(1)}{id : \tau_{id}, ys :: L^2(1) \mid_0^2 cons(y, tick\{2\}id(ys)) : L^0(1)}}}{}$$

$$\dfrac{\text{relax}\ \dfrac{y : 1 \mid_2^0 y : 1}{id : \tau_{id}, y : 1, ys :: L^2(1) \vdash e_{id} : L^0(1)}}{\dfrac{id : \tau_{id}, x : L^2(1) \mid_0^0 \lambda(x) e_{id} : \tau_{id}}{\dfrac{id : \tau_{id} \mid_0^0 \lambda(x) e_{id} : \tau_{id}}{\mid_0^0 fix(id\lambda(x)e_{id}) : \tau_{id}}}}$$

$$x : L^2(1) \mid_0^0 z : L^2(1) \qquad id : \tau_{id} \mid_0^0 nil : L^0(1)$$

Figure 1: Example of type inference

```
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 2: Code for binary counter example