Resource Analysis: Lecure 1

JAN HOFFMANN

1 MOTIVATION OF RESOURCE ANALYSIS

How do we compare two implementations of the same function?

- efficiency
  - time complexity
  - memory usage
  - power consumption
- kinds of analysis we might be interested in
  - worst-case analysis
  - average-case analysis
- traditional tool: analysis of algorithms
  - algorithms are usually defined in pseudocode, which leads to the following problems:
    - algorithm description's relation to the actual implementation is unclear
    - cost model is not specified
    - the format of input is not specified
    - ...
  - Complexity is analyzed using big-O:
    - recap: $f \in O(g)$ means $\exists n_0, c. \forall n \geq n_0. f(n) \leq cg(n)$.
    - some algorithms might have large constants thus are actually slow in practice
    - cannot distinguish algorithms with the same asymptotic behavior
    - no information on "small" inputs
    - definitions are hard to generalize to multiple inputs

2 RESOURCE ANALYSIS

- in order to analyze programs (mathematical objects), we need
  - syntax and static semantics
  - cost semantics
- to take into account constant factors
  - no other way but have to figure out the constants even we want to use big O eventually
  - want some help from proof rules, type systems, and other automation techniques to figure out the exact factors
- bounds that come with certificates, i.e., proofs on how the bounds are obtained

In this short course series we focus on functional languages, worst-case bounds, and automatic bound inference.

Author's address: Jan Hoffmann, jhoffmann@cmu.edu.

3 A SIMPLE LANGUAGE

3.1 Syntax

Types and expressions are defined as follows:

⟨*Types*⟩ τ ::= $\mathrm{arr}(\tau_1, \tau_2)$ // function from τ_1 to τ_2, written as $\tau_1 \to \tau_2$
| unit

⟨*Expressions*⟩ e ::= x // variables, written as x
| $\mathrm{app}(e_1, e_2)$ // function application, written as $e_1(e_2)$
| $\mathrm{lam}\{\tau\}(x.e)$ // function with argument x of type τ and body e, written as $\lambda(x:\tau)e$
| ⟨⟩ // empty expression

Values are inductively defined as follows:

$$\frac{}{\langle\rangle \text{ val}} \qquad\qquad \frac{}{\lambda\,(x:\tau)\,.\,e \text{ val}}$$

3.2 Cost semantics I: structural dynamics (also known as small-step semantics)

$e \mapsto e'$ means expression e evaluates to e' in one step.

$$\frac{e_1 \mapsto e_1'}{e_1(e_2) \mapsto e_1'(e_2)} \qquad \frac{e_1 \text{ val} \qquad e_2 \mapsto e_2'}{e_1(e_2) \mapsto e_1(e_2')} \qquad \frac{e_2 \text{ val}}{(\lambda(x:\tau)e)e_2 \mapsto [e_2/x]e}$$

Fig. 1. Structural dynamics for the simple language.

Definition: let $e : \tau$, the evaluation cost of e is n if $e \mapsto^n v$ or ∞ otherwise.

Note that here the evaluation is deterministic and unique, so the cost definition makes sense.

Here we also assume that all steps cost the same as a simplification, but in reality this is not necessarily the case.

3.3 Evaluation dynamics (also known as big-step semantics)

We define the following notations:

$e \Downarrow^q v$: expression e evaluates to value v, and this costs q.

$V \vdash_M e \Downarrow^q v$: under environment V : Vars $\to$ Vals, with cost metric M, expression e evaluates to val v with cost $q \in \mathbb{Q}$.

M : {unit, app, lam, var} $\to \mathbb{Q}$: cost metric that maps each operation to a rational cost value. Here costs could be negative, indicating that more resources are becoming available.

Rules are given in Fig. 2.

What metric S should we use?

$$\frac{}{V \vdash_M x \Downarrow^{M(\text{var})} V(x)} \qquad \frac{}{V \vdash_M \langle\rangle \Downarrow^{M(\text{var})} \langle\rangle} \qquad \frac{}{V \vdash_M \lambda(x:\tau)\,.\,e \Downarrow^{M(\text{var})} \lambda(x:\tau)\,.\,e}$$

$$\frac{V \vdash_M e_1 \Downarrow^{c_1} \lambda(x:\tau)e \qquad V \vdash_M e_2 \Downarrow^{c_2} v_2 \qquad V[x \mapsto v_2] \vdash_M e \Downarrow^{c} v}{V \vdash_M e_1(e_2) \Downarrow^{c+c_1+c_2+M(\text{app})} v}$$

Fig. 2. Evaluation dynamics

$$S(\text{app}) = 1 \tag{1}$$

$$S(\text{unit}) = 0 \tag{2}$$

$$S(\text{var}) = 0 \tag{3}$$

$$S(\text{lam}) = 0 \tag{4}$$

Using cost metric S, for all $e : \tau$ we can prove that $e \mapsto^n v$ iff $\vdash_S e \Downarrow^n v$

4 HIGH-WATER MARK

When resources come and go we want to know the high-water mark within the process.

An example is stack usage. The number of stack cells we occupied changes as time goes by, but we are most interested in the highest stack usage in the process.

For this purpose, notice that $V \vdash_M e \Downarrow^q v$ (where q acts like net cost) is not what we want.

We give two options in the following subsections.

4.1 Resource safety

$V_M \vdash^q_{q'} e \Downarrow v$: under environment V, metric M, and given that q resources are available, expression e evaluates to value v, and q' resource are available afterwards (we always require $q, q' \geq 0$ or it doesn't make sense)

Semantics are show in Fig. 3.

$$\frac{}{V_M \vdash^{q+M(\text{var})}_{q'} x \Downarrow V(x)} \qquad \frac{}{V_M \vdash^{q+M(\text{lam})}_{q} \lambda(x:\tau)\,.\,e \Downarrow \lambda(x:\tau)\,.\,e}$$

$$\frac{V_M \vdash^q_p e_1 \Downarrow \lambda(x:\tau)\,.\,e \qquad V_M \vdash^p_r e_2 \Downarrow v_2 \qquad V[x \mapsto v_2]_M \vdash^r_{q'} e \Downarrow v}{V_M \vdash^{q+M(\text{app})}_{q'} e_1(e_2) \Downarrow v} \qquad \frac{}{V_M \vdash^{q+M(\text{app})}_{q'} e_1(e_2) \Downarrow v}$$

Fig. 3. Big-step semantics for resource safety

We can prove the following results regarding resource safety.

Lemma 4.1. *Let $V_M \vdash^q_{q'} e \Downarrow v$ and $V_M \vdash^p_{p'} e \Downarrow v'$ then $q - q' = p - p'$ and $v = v'$.*

Theorem 4.2. *If $V_M \vdash^q_{q'} e \Downarrow v$ then $V \vdash_M e \Downarrow^{q-q'} v$.*

Note that the theorem is NOT saying: $V \vdash_M e \Downarrow^c v$ implies $V_M \vdash_0^c e \Downarrow v$.

4.2 Resource monoid

$V \vdash_M e \Downarrow v \mid (q, q')$: q is the high-water mark and q' is left-over resources ($q, q' \geq 0$).

$$\frac{M(var) \geq 0}{V \vdash_M x \Downarrow V(x) \mid (M(\text{var}), 0)} \qquad \frac{M(var) < 0}{V \vdash_M x \Downarrow V(x) \mid (0, -M(\text{var}))}$$

$$\frac{M(app) \geq 0 \qquad V \vdash_M e_1 \Downarrow \lambda(x:\tau)e \mid (q, q') \qquad V \vdash_M e_2 \Downarrow v_2 \mid (p, p') \qquad V[x \mapsto v_2] \vdash_M e \Downarrow v \mid (r, r')}{V \vdash_M e_1(e_2) \Downarrow v \mid (M(\text{app}), 0) \cdot (q, q') \cdot (p, p') \cdot (r, r')}$$

Fig. 4. Semantics for resource monoid method. For simplicity we only list the rule for app when its cost is nonnegative. The definition of the operator $\cdot$ is left as an exercise.

Homework: define the operator $\cdot$: $(p, p') \cdot (q, q') =?$ so that the following theorem holds

Theorem 4.3. (1) *If* $V \vdash_M e \Downarrow v \mid (q, q')$ *then* $V \vdash_{q'}^{q} e \Downarrow v$.
(2) *If* $V \vdash_{q'}^{q} e \Downarrow v$ *then* $V \vdash e \Downarrow v \mid (p, p')$ *and* $p \leq q$.