A Naive Time Analysis and its Theory of Cost Equivalence

Author: David Sands
Department of Computer Science,
University of Copenhagen, Denmark

Source: J. of Logic and Computation,
v. 5, n.1, 1995, p.1-46

Presenter: Nikolay V. Shilov, ROPAS

Date: 15^{th} of April, 2000

ROPAS workshop on
Resource Bound Checking

A direct (or *naive*) approach for reasoning about time cost of lazy evaluations comprises:

- *explicit time rules* (extracted from operational semantics),
- an *axiomatizable cost equivalence* (a timed bisimulation congruence).

In contrast, a *compositional* approach [M.Bjerner and S.Holmsrom, 1989]

1. parameterizes costs of needed results,
2. generates equations for costs,
3. resolves imposed constraints.

First-order case: syntax

Program are closed collections of function definitions which exploits expressions $\mathcal{E}$ constructed with *function call*, *primitive functions*, *conditions*, *case expressions*, *cons-lists*, *identifiers* and *constants*:

$$\begin{array}{c}
\mathcal{E} ::= \quad \underbrace{\mathcal{F}(\mathcal{E}^*)}_{function\ call} \quad | \quad \underbrace{\mathcal{P}(\mathcal{E}^*)}_{primitive\ function} \quad | \\
| \ \underbrace{if\ \mathcal{E}\ then\ \mathcal{E}\ else\ \mathcal{E}}_{condition} \ | \\
| \ \underbrace{case\ \mathcal{E}\ of\ nil\ \Rightarrow\ \mathcal{E}\ \mathcal{X} : \mathcal{X}\ \Rightarrow\ \mathcal{E}}_{case\ expressions} \ | \\
| \ \underbrace{\mathcal{E} : \mathcal{E}}_{cons-list} \ | \ \underbrace{\mathcal{X}}_{identifier} \ | \ \underbrace{\mathcal{C}}_{constant}
\end{array}$$

where $\mathcal{F}$, $\mathcal{P}$, $\mathcal{X}$ and $\mathcal{C}$ are alphabets of functional variables, primitive functions, identifiers and constants respectively.

First-order case: example

```
append(xs,ys) = case xs of
                  nil => ys
                  h:t => h:append(t,ys)

reverse(xs)   = case xs of
                  nil => nil
                  h:t =>
                  append(reverse(t),h:nil)

head(xs)      = case xs of
                  nil => undefined
                  x:t => h

-----------------------------------------
```

`head(reverse(v))` computes
the last element of non-empty list-value.

First-order case: lazy semantics

Let us fix a *deterministic* program P. Semantics of S is defined in terms of transformations of two kinds:

$$\mathcal{E} \to_N \mathcal{V} \text{ and } \mathcal{E} \to_H \mathcal{H}$$

where $\mathcal{V}$ are fully evaluated expressions (values) while $\mathcal{H}$ are head-normal forms:

$$\mathcal{V} ::= \mathcal{C} \mid \mathcal{V} : \mathcal{V} \text{ and } \mathcal{H} ::= \mathcal{C} \mid \mathcal{E} : \mathcal{E}$$

Let $\mathcal{E} \to_\alpha \mathcal{U}$ represent transformations of both kinds:

$$\alpha \in \{N, H\} \qquad\qquad \mathcal{U} ::= \mathcal{V} \mid \mathcal{H}$$

Deterministic assumptions are:

- $\frac{e \to_\alpha u' \quad e \to_\alpha u''}{u' \equiv u''}$ for all u' and u'',
- $\frac{\frac{\nabla'}{e \to_\alpha u} \quad \frac{\nabla''}{e \to_\alpha u}}{\nabla' \simeq \nabla''}$ for all proofs ∇' and ∇''

for every expression e.

Fun $$\frac{e\{e_1/x_1,\ldots e_n/x_n\}\rightarrow_\alpha u}{f(e_1,\ldots e_n)\rightarrow_\alpha u}$$

$(f(x_1,\ldots x_n)=e)$ is a definition in P

Prim $$\frac{e_1\rightarrow_H c_1 \;\ldots\; e_n\rightarrow_H c_n \quad v{=}appl(p,c_1,\ldots c_n)}{p(e_1,\ldots e_n)\rightarrow_\alpha v}$$

p is primitive function

Cond $$\frac{e_1\rightarrow_H true \qquad e_2\rightarrow_\alpha u}{if\ e_1\ then\ e_2\ else\ e_3\rightarrow_\alpha u}$$

$$\frac{e_1\rightarrow_H false \qquad e_3\rightarrow_\alpha u}{if\ e_1\ then\ e_2\ else\ e_3\rightarrow_\alpha u}$$

Case $$\frac{e_1\rightarrow_H nil \qquad e_2\rightarrow_\alpha u}{(case\ e_1\ of\ nil{\Rightarrow}e_2\ x{:}xs{\Rightarrow}e_3)\ \rightarrow_\alpha u}$$

$$\frac{e_1\rightarrow_H e_h{:}e_t \qquad e_3\{e_h/x,e_t/xs\}\rightarrow_\alpha u}{(case\ e_1\ of\ nil{\Rightarrow}e_2\ x{:}xs{\Rightarrow}e_3)\ \rightarrow_\alpha u}$$

Cons $$\frac{e_1\rightarrow_N v_1 \quad e_2\rightarrow_N v_2}{e_1{:}e_2\rightarrow_N v_1{:}v_2}$$

$$\frac{}{e_1{:}e_2\rightarrow_H e_1{:}e_2}$$

Const $$\frac{}{c\rightarrow_\alpha c}$$

First-order case: direct time analysis

Time cost of an expression e is a number of (non-primitive function) calls occurring in successful evaluations of the expression. Due to deterministic assumptions,

- *a unique proof* ∇ *of* $e \rightarrow_\alpha u$ *for some* u *is the unique successful evaluation of the expression* e*;*
- *a number of instances of the rule* **Fun** *in the proof* ∇ *is equal to the time cost of the expression* e*.*

For every proof ∇ let $T(\nabla)$ be a total amount of instances of **Fun** in the ∇. The following equation is straightforward: if $\nabla \equiv \frac{\nabla_1 \ ... \ \nabla_n}{e \to_\alpha u}$, where ∇_1, ... ∇_n are subproofs and $\frac{root(\nabla_1) \ ... \ root(\nabla_n)}{e \to_\alpha u}$ is an instance of a **rule**, then $T(\nabla)$ is

$$\begin{cases} 1 + T(\nabla_1) + ... + T(\nabla_n), \text{ if } \textbf{rule}{=}\textbf{Fun}, \\ T(\nabla_1) + ... + T(\nabla_n) \text{ otherwise.} \end{cases}$$

For every expression e let $\langle e \rangle^N$ and $\langle e \rangle^H$ be time costs of full evaluation and, respectively, of head-normal evaluation of e: $\langle e \rangle^\alpha = T(\nabla)$ *for a proof* ∇ *of* $e \to_\alpha u$ *for some* u*.*

First-order case: time rules

$$\langle f(e_1, ...e_n)\rangle^\alpha \;=\; 1 + \langle e\{e_1/x_1, ...e_n/x_n\}\rangle^\alpha$$

$$\langle p(e_1, ...e_n)\rangle^\alpha \;=\; \langle e_1\rangle^H + ...\langle e_n\rangle^H$$

$$\langle if\ e_1\ then\ e_2\ else\ e_3\rangle^\alpha \;= \\ = \begin{cases} \langle e_2\rangle^\alpha & \text{if } e_1 \to_H true \\ \langle e_3\rangle^\alpha & \text{if } e_1 \to_H false \end{cases}$$

$$\langle case\ e_1\ of\ nil \to e_2\ x : xs\ e_3\rangle^\alpha \;=$$

$$= \;\langle e_1\rangle^H \;+ \begin{cases} \langle e_2\rangle^\alpha & \text{if } e_1 \to_H nil \\ \langle e_3\{e_h/x, e_t/xs\}\rangle^\alpha & \text{if } e_1 \to_H e_h : e_t \end{cases}$$

$$\langle e_1 : e_2\rangle^N \;=\; \langle e_1\rangle^N + \langle e_2\rangle^N$$

$$\langle e_1 : e_2\rangle^H \;=\; \langle c\rangle^\alpha \;=\; 0$$

First-order case: example

`head(reverse(v))` for
a non-empty list-value $v \equiv v_h : v_t$

$$
\begin{aligned}
&\underline{\langle \texttt{head(reverse(}v\texttt{))} \rangle^N} = \\
&\qquad = 1 + \langle\ \texttt{reverse(}v\texttt{)}\ \rangle^H + \langle e_h \rangle^N = \\
&= 1 + (1 + \langle e_h \rangle^N + \\
&+ \langle\ \texttt{append(reverse(}v_t\texttt{),}\ v_h : nil\texttt{)}\ \rangle^H) = \\
&= 2 + 1 + \langle\ \texttt{reverse(}v_t\texttt{)}\ \rangle^H + \\
&\qquad + \begin{cases} \langle nil \rangle^H, \text{ if } \texttt{reverse(}v_t\texttt{0} \to_H nil \\ \langle e'_h : \texttt{append(}e'_h\texttt{,}v_h : nil \rangle^H, \\ \text{if } \texttt{reverse(}v_t\texttt{)} \to_H e'_h : e'_t \end{cases} \\
&= 3 + \langle\ \texttt{reverse(}v_t\texttt{)}\ \rangle^H = \\
&\qquad\qquad = \underline{2(1 + |v|)}.
\end{aligned}
$$

Higher-order case: syntax, semantics and time rules

Let us extend syntax of expressions $\mathcal{E}$ by *lambda abstraction* $\lambda\mathcal{X}.\mathcal{E}$ and *application* $\mathcal{E}\mathcal{E}$ with lazy semantics:

Lambda $\dfrac{}{\lambda x.e \to_\alpha \lambda x.e}$

Apply $\dfrac{e_1 \to_H \lambda x.e \quad e_2\{e_2/x\} \to_\alpha u}{e_1 e_2 \to_\alpha u}$

Corresponding time rules are:

$$\langle \lambda x.e \rangle^\alpha = 0$$

$$\langle e_1 e_2 \rangle^\alpha = 1 + \langle e_1 \rangle^H + \ldots \langle e\{e_2/x\} \rangle^\alpha, \quad \text{if } e_1 \to_H \lambda x.e$$

Back to first-order case: axiomatization of cost equivalence

Let $\mathtt{I}$ be an identity function `I(x) = x` which represents a single *tick* of time cost. For all e, u, α and $t \in N$ let us write

$$e \xrightarrow{t}_{\alpha} u \text{ iff } e \rightarrow_{\alpha} u \text{ and } \langle e \rangle^{\alpha} = t.$$

Theorem. *For all e, h and t*

- *(completeness)*
  if $e \xrightarrow{t}_{\alpha} h$ then $\models_K e =_{\langle\rangle} \mathtt{I}^t(h)$;
- *(soundness)*
  if $\models_K e =_{\langle\rangle} \mathtt{I}^t(h)$ then $e \xrightarrow{t}_{\alpha} h'$ and $\models_K h' =_{\langle\rangle} h$ for some h'.