

Amortised analysis of heap consumption

Olha Shkaravska

Institut of Informatics, LMU
Munich, Germany

Motivation and Structure of this talk

Hofmann-Jost inference system:
inference of **linear** bounds of heap consumption.

What about non-linear bounds?

The talk:

- Amortized analysis for time, incl. Banker's algorithm
- Hofmann-Jost analysis = a Banker's algorithm with constant credits
- A Banker's algorithm for dependent credits

Amortised Time Analysis

Idea: to distribute the worst-case run time of an entire sequence of operations over the operations.

Given: a sequence of n operations.

Let a_i and t_i be the amortized and actual costs of i -th operation.

$$\sum_{i=1}^j a_i \geq \sum_{i=1}^j t_i,$$

where $1 \leq j \leq n$.

Aggregate method

$$a_i = \frac{T(n)}{n}$$

... like we have payed in the African restaurant.

Banker's (Accounting) method

If

$$a_i \geq t_i$$

then $c_i = a_i - t_i$ is viewed as a **credit**.

It can be used late to pay for the operations whose amortised cost is less than their actual cost.

Example.

```
while not StackEmpty(S) and k <> 0
do {
    Pop(S)
    k := k - 1
}
```

Banker's (Accounting) method

```
while not StackEmpty(S) and k <> 0  
do { Pop(S); k := k-1 }
```

The actual costs, t_i -s:

Push 1

Pop 1

Multipop $\min(s, k)$,

where s is a size of the stack S .

The amortized costs, a_i -s:

Push 2

Pop 0

Multipop 0

Physicist's (Potential) method

One can associate all “prepayment” with the data structure as a whole.

Data structures: D_0, \dots, D_n :

- D_0 is an initial one,
- D_i is a result of application of i -th operation on D_{i-1}

Find a **potential function** $\Phi : D_i \mapsto \Phi(D_i)$,
a number.

The amortised cost per op.:

$$a_i = t_i + (\Phi(D_i) - \Phi(D_{i-1}))$$

Physicist's (Potential) method

The amortised cost per op.:

$$a_i = t_i + (\Phi(D_i) - \Phi(D_{i-1}))$$

The total amortized cost is

$$\begin{aligned}\sum_{i=1}^n a_i &= \sum_{i=1}^n t_i + \sum_{i=1}^n (\Phi(D_i) - \Phi(D_{i-1})) \\ &= \sum_{i=1}^n t_i + (\Phi(D_n) - \Phi(D_0))\end{aligned}$$

Hofmann-Jost inference system

We can infer linear heap-consumption bounds:

- Given

$$f : L(\text{Int}) \rightarrow L(\text{Int})$$

- Obtain a notated, with numbers, signature

$$f : L(\text{Int}, k), k_0 \rightarrow L(\text{Int}, k'), k'_0$$

Examples:

$$\text{copy} : L(\text{Int}, 1), 0 \rightarrow L(\text{Int}, 0), 0$$

$$\text{cons} : L(\text{Int}, 0), 1 \rightarrow L(\text{Int}, 0), 0$$

k is a constant credit, $k |l|$ is a potential of the list l .

Type system for dependent credits

$$\mathbb{B} = \{0, 1\}$$

$$T ::= \mathbb{B} | \mathbf{L}_0(T, k) | \dots | \mathbf{L}_m(T, k) | \dots | \mathbf{L}(T, k),$$

where

- $k : \mathbf{Nat} \rightarrow \mathbb{R}^+$,
- $\mathbf{L}_m(T, k)$ is a not. list of length m of type T , s. t. i -th element of the list has a credit $k(i)$,
- $\mathbf{L}(T, k) = \sum_{n=0}^{\infty} \mathbf{L}_n(T, k)$.

Inference for dependent credits

Typing judgment is almost the same as for HJ typing:
 $\Gamma, n \vdash e : T, n'$

The context is mixed: with non-sized and sized types.

$$\frac{n \geq n' + 1 + k(m + 1)}{h : T, t : \mathbf{L}_m(T, k), n \vdash} \text{Cons}$$
$$\text{cons}(h, t) : \mathbf{L}_{m+1}(T, k), n'$$

Inference for dependent credits

$$\frac{\begin{array}{c} \Gamma, n \vdash e_1 : A, n' \\ \Gamma, h : T, \\ t : \mathbf{L}_{m-1}(T, k), n + 1 + k(m) \vdash e_2 : A, n' \end{array}}{\Gamma, l : \mathbf{L}_m(T, k), n \vdash} \text{DM}$$

match l with

$$\begin{array}{l} \text{Nil} \Rightarrow e_1 : \quad A, n' \\ | \text{Cons}@(h, t) \Rightarrow e_2 : \end{array}$$

Inference for dependent credits

The rule

$$\frac{\begin{array}{l} \Gamma, n \vdash e_1 : A, n' \\ \Gamma, n \vdash e_2 : A, n' \end{array}}{\Gamma, x : B, n \vdash \text{if } x \text{ then } e_1 \text{ else } e_2 : A, n'} \text{If}$$

perhaps, is not that restrictive if we have

$$\frac{\Gamma, n \vdash e : L_m(T, k), n'}{\Gamma, n \vdash e : L(T, k), n'} \text{Sum}$$

Inference for dependent credits

$$\begin{array}{c} \Sigma(P) = \mathbb{L}(T, k), k_0 \xrightarrow{p} \\ \mathbb{L}(T', k'), k'_0 \\ n \geq k_0 \\ n - k_0 \geq n' - k'_0 \\ \hline \Gamma, l : \mathbb{L}_m(T, k), n \vdash \\ P(l) : \mathbb{L}_{p(m)}(T', k'), n' \end{array} \text{Fun}$$

Inference for dependent credits

$$\frac{\Sigma(P) = \mathbb{L}(T, k), k_0 \xrightarrow{p} \mathbb{L}(T', k'), k'_0}{l : \mathbb{L}_m(T, k), k_0 \vdash e_P : \mathbb{L}_{p(m)}(T', k'), k'_0} \text{Spec}$$

Checking heap bounds

Given a program of $\mathbb{L}(T) \rightarrow \mathbb{L}(T')$.

How to check, if its heap consumption does not exceed $O(f(x))$, where x is a length of an input list, and $f(x)$ is smooth?

Notate the signature

with functions of $k, k' : \text{Nat} \rightarrow \mathbb{R}^+$ and nat. numbers k_0, k'_0 :

$\mathbb{L}(T, k), k_0 \rightarrow \mathbb{L}(T', k'), k'_0$

Take $k = f'$

Checking heap bounds $f(x)$

Take $k = f'$ in $\mathbb{L}(T, k)$, $k_0 \rightarrow \mathbb{L}(T', k')$, k'_0

If type-checking for this k and some nonnegative k_0 , k' , k'_0 works (a bit of type-inference for k_0, k', k'_0), then the program consumes up to $O(f(x))$ heap units.

Why?

- $f(x) = \int_0^x f'(v) dv + f(0)$
- $\sum_{v=1}^x k(v)$ is a total amount of free heap units associated with an input list of length x
- approximate the integral by the sum $\sum_{v=1}^x k(v)$,
rectang. approx. of the square:
 $|\sum_{v=1}^x k(v) - f(x)| \leq C.$

Examples

- $f(x) = x$ for copy: we have
 $k(x) = f'(x) \equiv 1$
- $f(x) = a \log(x + b) + c$ and $k(x) = \frac{a}{\ln 2} \frac{1}{x+b}$
for binary

```
binary l =  
  match l with  
  Nil => Nil  
  | Cons(h, t) => let y = binary t  
                  in binInc y
```

where

To generalise type inference

```
binInc l =  
  match l with  
  Nil => Cons(1, Nil)  
  | Cons@(h, t) => if h=0 then  
                    Cons(1, t)  
                    else  
                      Cons(0, binInc t)
```

If $\|l\| = 2^s - 1$, for some natural number s ,
binInc consumes exactly one heap unit,
otherwise there is no consumption.

Consider another measure $\mu = \|\cdot\|$.

Consumption on measure

$$\begin{aligned}\text{consume}(l) &= f(\mu(l)) \\ &= \int_0^{\mu(l)} f'_\mu(v) dv + f(0) \\ &\approx \sum_{v=1}^{\mu(l)} f'_\mu(v) + C\end{aligned}$$

Generally: a credit in heap units is payed pro 1 unit of growth of measure.

Consumption on measure: the example to do

binInc

- $\mu = \|\cdot\|$,
- $f(x) = \lceil \log_2(x+2) \rceil - \lceil \log_2(x+1) \rceil$,
- $k(x) = \frac{a}{\ln 2} \frac{1}{x+2} - \frac{b}{\ln 2} \frac{1}{x+1}$
- $\mathbf{L}_d(\mathbf{B}, k), 0 \rightarrow \mathbf{L}_{d+1}(\mathbf{B}, 0), 0$,
where d is a measure of an input.

Functions of 2 arguments (back to length)

$$k_1, k_2 : \text{Nat} \rightarrow \text{Nat} \rightarrow \mathbb{R}^+$$

k_i : length of the partner \rightarrow position of the element \rightarrow credit

$$\Sigma(P) = \text{L}(T_1, k_1), \text{L}(T_1, k_2), k_0 \xrightarrow{p} \\ \text{L}(T', k'), k'_0 \\ n \geq k_0 \\ n - k_0 \geq n' - k'_0$$

$$\Gamma, l_1 : \text{L}_{m_1}(T, k_1(m_2)), l_2 : \text{L}_{m_2}(T, k_2(m_1)), n \vdash \\ P(l_1, l_2) : \text{L}_{p(m_1, m_2)}(T', k'), n'$$

Functions of 2 arguments

$$\begin{aligned} f(x, y) &= \int_0^y f'_y(x, u) du + f(x, 0) \\ &= \int_0^y f'_y(x, u) du + \int_0^x f'_{0_x}(v) dv \\ &\quad + f_0(0) \\ &\approx \sum_{u=1}^y f'_y(x, u) + \sum_{v=1}^x f'_{0_x}(v) \end{aligned}$$

where $f_0 := f(x, 0)$

Functions of 2 arguments

$$f(x, y) \approx \sum_{u=1}^y f'_y(x, u) + \sum_{v=1}^x f'_{0x}(v)$$

with $f_0 := f(x, 0)$

Let

$$k_1(y) = f'_{0x}$$
$$k_2(x) = f'_y(x, y)$$

Example of bounds

$$f(x, y) = axy + bx + cy + d ?$$

How to answer this question?

$$k_1(y) = f'_{0x} = b$$

$$k_2(x) = f'_y(x, y) = ax + c$$

Find such a, b, c, d that for some k_0, k', k'_0 type-checking works...

Let for simplicity $k' \equiv 0, k'_0 = 0$.

Example: Multiplication

```
mult(l1, l2) =  
match l2 with  
  Nil => Nil  
  | Cons(h, t) => let x=mult(l1, t)  
                  in  
                  let l=copy l1  
                  in  
                  cons(l, x)
```

Typechecking works with
 $a = 1, b = 0, c = 1, k_0 = 0$

$$L(\mathbb{B}, 0), L(\mathbb{B}, x + 1), 0 \longrightarrow L((\mathbb{B}, 0), 0), 0$$

To Do

- Design an Inference system parametric w.r.t. measures
- The Examples: type-checking revisited
- Soundness of the inference system w.r.t. op.sem of Hofmann-Jost