Asymptotic Resource Usage Bounds

E.Albert[1],D.Alonso[1], P.Arenas[1], S.Genaim[1] and G.Puebla[2]

[1]DSIC, Complutense University of Madrid, Spain
[2]CLIP, Technical University of Madrid, Spain

December 15^{th} 2009

Contents

1. Motivation: Cost Analysis and Asymptotic Orders.
2. Background: Cost expressions.
3. Orders $\mathcal{O}, \Theta$ for cost expressions.
4. *asymp*: Simplification of cost expressions.
5. Scalability effects.

Motivation

- Execution Cost is a fundamental characteristic of programs.
- There exists many non-asymptotic cost-analyzers.
- Cost is typically described in **asymptotic** terms:
  - Focus on the scalability over the input size.
  - Implementation independence: Ignoring constant proportions.
  - Readability: expressions are more compact and manageable.
- Asymptotic Cost Analysis: from a program, obtain a cost function f_a that describes its asymptotic cost.
  - Better performance for obtaining closed-form results, without recurrences and indeterminacy.
  - Better scalability: Non-asymptotic functions can become too large and complex, whereas asymptotic functions are smaller and simpler.

Contributions

1. Adaptation of multivariable $\mathcal{O}$ and Θ to cost expressions.
   1. Cost expressions can model the cost of realistic programs.
   2. Asymptotic behaviour determined by its loops.
2. Definition of asymptotic transformation on cost expressions.
   1. Simplifies cost expressions to normal form.
   2. Removes subsumed operands using context information.
   3. Transforms to asymptotic form any cost output from any non-asymptotic analyzer.
3. Implementation of an asymptotic cost analyzer.

- **Cost Expressions** can be used for descrbing execution costs.
- Their syntax follows this grammar:

$$
\begin{array}{lllllllll}
exp & ::= & r & \mid & log_a(1+nat(l)) & \mid & exp+exp & \mid & max(\{exp\}) \\
 & \mid & nat(l) & \mid & a^{nat(l)} & \mid & exp*exp & &
\end{array}
$$

where $a \in \mathbb{N}$ and $a \geq 2$, $r \in \mathbb{Q}^+$ and l is a linear expression.

Operand *nat* avoids negative values of cost $nat(l) = max(l, 0)$

- This grammar roughly maps to programming constructs:

$r \in \mathbb{Q}^+$	basic operations
$nat(l)$	iterations of a loop
$a^{nat(l)}$	multiple recursion
$log_a(nat(l)+1)$	divide and conquer recursion
$exp + exp$	sequences
$exp * exp$	loops
$max(exp, exp)$	indeterminism

- They can describe any kind of estimate (upper or lower bounds).

```
class List{
    boolean data;  List next;
    static m(List x, int i, int n){
        while (i<n)
            if (x.data){ g(i,n); i++;}
            else       { g(0,i); n--;}
            x=x.next;
        }
    }
}
```

- An upper bound of the number of executed bytecode instructions:

$$C_g^+(a,b) = 4 + \mathrm{nat}(b-a)$$

$$C_m^+(n,i) = 6 + \underbrace{nat(n-i)}_{iters} * \overbrace{max(\{\underbrace{19 + 5 * nat(n-i)}_{then}, \underbrace{21 + 5 * nat(n-1)}_{else}\})}^{if}$$

- The asymptotic form: $C_m^{as^+}(n,i) = \underbrace{nat(n-i)}_{iters} * \underbrace{nat(n)}_{if}$

Definition (Multivariable Asymptotic Orders)

Let $f, g : \mathbb{N}^m \mapsto \mathbb{R}^+$ be two functions. We say that $f \in \mathcal{O}(g)$ **if** $\exists n \in \mathbb{N}$ and $c \in \mathbb{R}^+$with $c > 0$ s.t.

$$\forall \bar{v} \in \mathbb{N}^m : (\forall_{i=1}^m v_i \geq m) \rightarrow f(\bar{v}) \leq c * g(\bar{v})$$

and similarly, $f \in \Theta(g)$, **if** $\exists n \in \mathbb{N}$ and $c_1, c_2 \in \mathbb{R}^+$with $c_i > 0$ s.t.
$\forall \bar{v} \in \mathbb{N}^m : (\forall_{i=1}^m v_i \geq m) \rightarrow c_1 * g(\bar{v}) \leq f(\bar{v}) \leq c_2 * g(\bar{v})$

But we can't use the variables of cost expressions as inputs for the definition: they appear in a linear combination inside a nat expression.

Example (Asymptotic value of *nat*)

The asymptotic value of $nat(n - i)$ can be ∞, if n tends to ∞ and i remains constant, or 0 if $n \leq i$.

Solution: instead of e we use the *nat*-**free** form $\tilde{e}$, where each *nat* is replaced by an atomic *nat*-variable $V \in \mathbb{Q}^+$.

Example (*nat*-free forms)

with $A = x + 2y$ and $B = x - 2y$

$$\begin{array}{rcl} nat(2x+4y+1) * 2^{nat(2x-4z+1)} & \rightarrow & A * 2^{2*B} \\ log_2(nat(x-2z)+1) * nat(x+2y) & \rightarrow & log(B) * A \end{array}$$

Definition (Asymptotic Notations of Cost Expressions)

Let e_1, e_2 be two cost expressions. Then

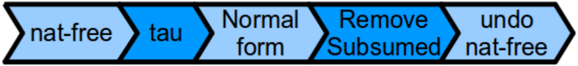
$$e_1 \in \mathcal{O}(e_2) \Leftarrow \tilde{e}_1 \in \mathcal{O}(\tilde{e}_2) \qquad e_1 \in \Theta(e_2) \Leftarrow \tilde{e}_1 \in \Theta(\tilde{e}_2)$$

Intuitively, a program's asymptotic behaviour depends mostly on the number of iterations of its loops, which are captured by *nat* expressions.

Asymptotic Simplification:*asymp*

Syntactic Simplifications on Cost Expressions

For a cost expression e, $asymp(e)$ is a expression e' obtained by:



- τ: remove constants and replace $max(e_1, e_2)$ by $\tilde{e}_1 + \tilde{e}_2$.
- Normalize e as $\sum_i \prod_j b_{ij}$ where b_{ij} is basic $nat{-}free$ expression.
- Remove subsumed addends: $B + B^2 \in \Theta(B^2)$. This step uses as input a **context constraint**, a system of inequalities $I \geq 0$ between the variables in e and the *nat*-variables in $\tilde{e}$.

Theorem (Soundness)

For any cost expression e, $asymp(e) \in \Theta(e)$.

Basic *nat−free* **expressions** are those with the form

- **Exponential:** 2^{r*A} where $r \in \mathbb{R}$ and $r > 0$.
- **Polynomial:** A^r where $r \in \mathbb{R}$ and $r > 0$.
- **Logarithmic:** $log_2 A$.

Basic cost expressions only contain one *nat*-variable.

Definition (*pow*, *deg*)

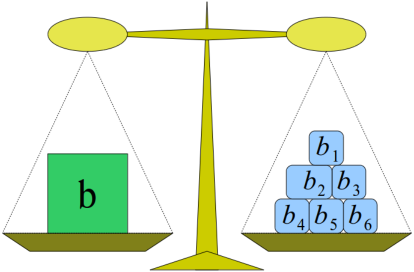
For any basic $nat-$free expression b. we define $pow(b), deg(b)$ as:

$$
\begin{array}{rr}
pow(2^{r*A}) = r & deg(2^{r*A}) = \infty \\
pow(A^r) = 0 & deg(A^r) = r \\
pow(log_2 A) = 0 & deg(log_2 A) = 0
\end{array}
$$

Asymptotic Simplification: *asymp*

Expression-Product Subsumption

For a basic *nat*−free cost expression b and a product M of basic cost expressions, we want to infer that $M \in \mathcal{O}(b)$.



- b is a basic *nat*−free cost expression of the *nat*−variable A.
- $M = b_1 * \cdots * b_m$, where each b_i is a basic expression of variable A_i
- We need a context constraint φ such that $\varphi \models A \geq A_i$.

Asymptotic Simplification: *asymp*

Expression-Product Subsumption

It holds that $\varphi \models M \in \mathcal{O}(b)$ **if**:

1. $b = 2^{r*A}$ and for $s = \sum_{i=1}^{m} pow(b_i)$,
   1. $r > s$ **or**
   2. $r = s$ and M has no polynomial nor logarithm.
2. $b = A^r$ and M has no exponential and for $s = \sum_{i=1}^{m} deg(b_i)$,
   1. $r > s$ **or**
   2. $r = s$ and M has no logarithm.
3. $b = log(A)$ and $M = log(A_i)$.

Example

$$2^A * A^n * \ldots \in \mathcal{O}(3^A) \qquad 2^A * A^2 \notin \mathcal{O}(2^A)$$

$$A * log(A) * \ldots \in \mathcal{O}(A^2) \qquad A^2 * log(A) \notin \mathcal{O}(A^2)$$

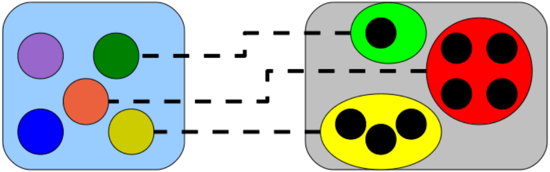
$$log(A) \in \mathcal{O}(log(A)) \qquad log(A) \notin \mathcal{O}(1)$$

Asymptotic Simplification: *asymp*

Product-Product Subsumption

A product M_1 **subsumes** a product M_2 **If**

- M_2 can be factorized in k subproducts $S_1, \ldots, S_k$
- and there are k distinct factors b_i of M_1
- and for every S_i, it holds that $S_i \in \mathcal{O}(b_i)$.

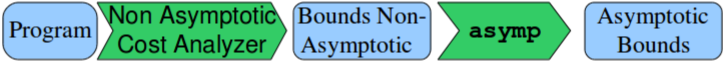


Example

Let $M_1 = 3^B A^3$, $M_2 = log(A)log(B)2^B C$ and $\varphi \equiv \{A \geq B, A \geq C\}$. If we factorize M_2 into $P_1 = 2^B$ and $P_2 = C * log(A) * log(B)$, we have that $P_1 \in \mathcal{O}(3^B)$ and $P_2 \in \mathcal{O}(A^3)$. Therefore, $M_2 \in \mathcal{O}(M_1)$.

Generation of Asymptotic Upper Bounds

- *asymp* can be used as a back-end of any non-asymptotic analyzer.



- **Asymptotic analyzer**: Integrates *asymp* in the solving process.

- We have integrated the *asymp* transformation in COSTA, and we have achieved the desired improvements in its performance.

Conclusions and Future Work

- Generic, automatic approach to asymptotic cost analysis
  - Traditionally done manually.
  - Real-life applications require mechanical techniques.
- Open Challenges:
  - Lower-bounds.
  - Certification of Resource Usage.
  - Modular Cost Analysis
  - Improve analysis accuracy.

Asymptotic Resource Usage Bounds

E.Albert[1],D.Alonso[1], P.Arenas[1], S.Genaim[1] and G.Puebla[2]

[1]DSIC, Complutense University of Madrid, Spain
[2]CLIP, Technical University of Madrid, Spain

December 15th 2009