# More on Cost Analysis

Wednesday, July 13, 2022 9:28 AM

Static Resource analysis of Program. with cost Decreasing. via finite walk.

Check are they able to handle cost decrease.

① via type system, mainly marco. & effect system. Dagstuhl
graded Hoare Logic. & its Categorical Semantics.

A Type-Theory for Higher-Order (Amortized) Cost analysis

Monadic Refinement for Relational Cost Analysis.

② Via Static Program Analysis [static cost analysis)
1. static cost analysis. for higher-Order Language.
⇒ Mainly Jan Hoffman. Zhong Shao.
⇒ Multivariate Amortized Resource analysis.

⇒ via type system:

① Relational Cost analysis: either { upper Bound - lower Bound. / sum all cost of subexpression. } ↓ same. ⇒

⇒ Extended. Relational Cost analysis for Functional-Imperative Programs.

② Monadic Refinement for Rel-cost.
⇒ higher-order refinements, and cost monads.

one issue: Lambda rule: $e \ominus e' \le t \rightarrow \lambda x.e \ominus \lambda x.e' \le 0$. ☆.

③. Graded Hoare Logic & its Categorical Semantics. → only increase.

④. A Type-Theory for Higher-Order (Amortized) Cost analysis ~ 2021.

⇒ λ-amor. amortized cost analysis.

⇒ potentials: cost have been accounted for, but not yet incurred.

☆: T-tick. only resource consume:

$$\text{T-tick}\ \frac{k : \mathbb{R}.}{\cdots \vdash \uparrow^k : M\ k\ 1}$$

☆ T-release:

$$\frac{\vdash e_1 : [p_1]\tau_1 \qquad \vdash e_2 : M\ (p_1 + p_2)\ \tau_2}{\vdash \text{release}\ x = e_1\ \text{in}\ e_2 : M\ p_2\ \tau_2}$$

release the cost stored in e. ☆.

☆. T-store:

$$\frac{\vdash e : \tau. \qquad \vdash p : \mathbb{R}^+.}{\vdash \text{store}\ e : M\ p\ ([p]\tau).}$$

✓ but not account this in eval.

⑤. Central Moment Analysis for Cost accumulators. in Probabilistic. <> ⑥ Bounded Expectations.
Resource Analysis for Probabilistic Programs.

⇒ bounds on prop. concentration inequality. ⇒ central moments.

⇒ Bounds on variants, central Moment, for ↓.

☆ cost-accumulator: termination time. [...]
rewards. in Markov decision [32]
position information [4. 7. 34].
cash flow [7. 24. >?. 41]. static approach. leverage aggregate infos of cost accumulator.

Amortized: Potential:

④. ⑦*: 2020. Exponential Automatic Amortized Resource analysis. ⇒ Resource Bound exp in size of inputs.

✓⑧ Space Bound Analysis for Functional Programs with Garbage Collection.

☆ ⇒ recovering the resource by restituting the potential to type the subexpression

$$\Rightarrow \frac{\Sigma ; \Gamma \vdash {}^{q}/_{q'}\ e_1 : B \qquad x_h : A,\ x_t : L^p(A) \vdash {}^{q+p-1}/_{q'}\ e_2 : B}{\vdash {}^{q}/_{q'}\ \text{match}\ x\ \{\text{nil} \to e_1 \mid \text{cons}(x_h, x_t) \to e_2\}.}$$ ☆.

⑨ 2011. Multivariate Amortized Resource Analysis.

⑩ 2010. Amortized Resource analysis with Polynomial Potential.

⑪ 2003. potential Based amortized analysis. ⇝ Types & Recurrences for Formal Amortized Analysis
⇒ Steffen Jost. Martin Hofman.
Static Prediction of Heap Space Usage for first-order Function Program.

⑫ 2000, A potential-Based Amortized Analysis of Union-find data Structure.

⑬ 1985: Tarjan. Amortized Computational Complexity
Σ actual cost - net gain of potential incured by data structure invocation.

2007 Extended: $\Gamma, n \vdash_\Sigma e : A, n'$ : n memory resource available, n' resources left over.

☆. Upper Bound decrease: in λ. rule.

$$\Rightarrow \frac{\Sigma(f) = (A_1, \ldots A_p, k) \to (C, k') \qquad n \ge k \qquad n - k + k' \ge n'}{\Gamma \vdash n \vdash f(x_1 \ldots x_p) : C : n' \ \Rightarrow\ n' = n - k + k'}$$

$n = n' + k - k'$ → estimated upper bound decreased by k', this k' is encapsulated as potential into the type of function

⑭. Unifying Type-Theory for Higher-Order (Amortized) Cost Analysis

④. T-release:

$$\frac{\cdots \vdash e_1 : [p_1]\tau_1 \qquad \cdots \vdash e_2 : M\ (p_1 + p_2)\ \tau_2}{\cdots \vdash \text{release}\ x = e_1\ \text{in}\ e_2 : M\ p_2\ \tau_2}$$

the potential for $e_1$. will not be assumed by $e_2$ after release. ⇒ But need explicit Release.
resource (cost) decreased by $p_2$.

↓ T-store

⑮ fix: fix.x.e ? recursive function?