# 自动摊还资源分析

一份入门笔记

Jun 14, 2024

**Mepy**

Mepy@stu.pku.edu.cn

## 概览

预设: 咱会点基本代数(知道啥叫幺半群即可, 参考近世代数), 以及描述语言的那套方法(参考TAPL/PFPL)

这篇笔记基本是对 **Types with Potential: Polynomial Resource Bounds via Automatic Amortized Analysis**, the thesis by Jan Hoffmann 的学习, 一些公式我懒得抄, 文法咱也先略过了...

## 预备工作

### 资源幺半群

考虑非负有理数加法幺半群 $(\mathbb{Q}_0^+, +, 0)$, 我们导出一个新的幺半群 $M = \mathbb{Q}_0^+ \times \mathbb{Q}_0^+$, 其中元素形如 $(q, q') \in M$, 表示需要 $q$ 资源启动计算, 计算后剩余 $q'$ 资源, 换言之, 所需资源为 $q - q'$, 但有启动阈值 $q$. 在此意味下, $M$ 上的加法定义如下：

$$(q, q') + (p, p') \coloneqq \begin{cases} (q, q' + p - p') \text{ if } q' > p \text{ i.e 剩余资源足以支付后续计算} \\ (q - q' + p, p') \text{ otherwise} \end{cases}$$

显然有嵌入映射 $i : \mathbb{Q}_0^+ \to M, i(q) \coloneqq (q, 0)$, 我们在元语言层面自动插入此映射. 我们一般令元变量 $p, q \in Q$, 当我们说 $q \in M$ 时, 事实在说 $(q, 0) \in M$. 例如当我们说幺元$0 \in M$时, 实际上说的是 $i(0) = (0, 0) \in M$ 是幺元, 读者自可验证其确实为幺元, 其余像 $q \in M$. 嵌入 $i$ 当然是同态, 即 $i(p) + i(q) = i(p + q)$, 读者展开定义验证即可.

注意, 我们并非强调 $q > q'$ 必然成立, 事实上形如 $(0, q')$ 可以用来表示资源增加, 即上述嵌入映射可以拓展到 $i : \mathbb{Q} \to M$, 若 $q \in \mathbb{Q}_0^+$, 则 $i(-q) \coloneqq (0, q)$.

### 目标语言

简要来说, 我们分析的目标语言是一个 higher order 的函数式语言, 基本上是 Simply Typed Lambda Calculus + 内建的 List $A$ 与 Tree $A$ 类型. 值得注意的是, list 和 tree 存于显式的堆中, 可以理解为有 Reference 拓展.

我们一般用 $V : \mathrm{Var} \to \mathrm{Value}$ 表示当前环境的变量对应的值, 其中变量来自函数参数或者 `let in`; 用 $H : \mathrm{Loc} \to \mathrm{Value}$ 表示堆内存, 用以存储 list 和 tree.

令元变量 $v \in \mathrm{Value}, l \in \mathrm{List}, t \in \mathrm{Tree}, A, B \in \mathrm{Type}, a \in \mathrm{Semantics}$. Value 指的是语言执行时的动态语义的值, Semantics 指的是做资源分析时的静态语义下的值. 原文中也用 $l \in \mathrm{Loc}$, 一般能上下文自明地区分, 本文会刻意指出 $l \in \mathrm{Loc}$.

在 Semantics 中 $l \coloneqq [a_1, ..., a_n]$ 表示一个长度为 $n$ 的 list, $\mathrm{po}(t) \coloneqq [a_1, ..., a_n]$ 表示树 $t$ 的前序遍历.

定义有判词 $H \vDash v \to a : A$, 指的是在堆 $H$ 下, $v \in \mathrm{Value}$ 对应静态语义 $a \in \mathrm{Semantics}$, 并赋有简单类型 $A$, 下面用例子简单解释之: 考虑 $l_0, l_1, l_2 \in \mathrm{Loc}, H = \{l_0 \to [], l_1 \to 514 :: l_0, l_2 \to 114 : l_1\}, v \coloneqq l_2$, 则可以有判断 $H \vDash v \to [114, 514] : \mathrm{List}$. 此判词用于描述堆是良赋型的(well typed).

再例, 如下有 $H \vDash l_7 \to [114, 233, 514] : \mathrm{Tree}$

$$\begin{aligned} H = \{ & l_0 \to \bullet, l_1 \to \bullet, l_2 \to \bullet, l_3 \to \bullet, l_4 \to \bullet, \\ & l_5 \to \mathrm{tree}(114, l_0, l_1), l_6 \to \mathrm{tree}(514, l_2, l_3), \\ & l_7 \to \mathrm{tree}(233, l_5, l_6)\} \end{aligned}$$

### 资源动态语义

简单来说, 是语言的(大步)动态语义带上了资源消耗注释. 方便起见, 我们单独扩充一个表达式 $\mathrm{tick}\ (q, q') \in \mathrm{Expr}$作为资源消耗注释, 其中 $(q, q') \in M$, 只有这一表达式消耗资源, 其余表达式并未消耗资源. 考虑 $(q, q') \in M, e \in \mathrm{Expr}, V : \mathrm{Var} \to \mathrm{Value}, H, H' : \mathrm{Loc} \to \mathrm{Value}$, 其中 $H$在计算$e$后变为 $H'$, 有判词 $V, H \vdash e \rightsquigarrow v, H' \mid (q, q')$

$$\frac{}{V, H \vdash \mathrm{tick}\ (q, q') \rightsquigarrow () \mid (q, q')}\ \text{E-Tick}$$

$$\frac{V(x) = [] \qquad V, H \vdash e_1 \rightsquigarrow v, H' \mid (q, q')}{V, H \vdash \text{match } x \text{ with } \mid [] \to e_1 \mid x_h :: x_t \to e_2 \rightsquigarrow v, H' \mid (q, q')}\ \text{E-MatchNil}$$

$$\frac{V(x) = l \quad H(l) = v_h :: v_t \quad V \cup \{x_h \to v_h, x_t \to v_t\}, H \vdash e_2 \rightsquigarrow v, H' \mid (q, q')}{V, H \vdash \text{match } x \text{ with } \mid [] \to e_1 \mid x_h :: x_t \to e_2 \rightsquigarrow v, H' \mid (q, q')}\ \text{E-MatchCons}$$

$$\frac{V(x) = \bullet \qquad V, H \vdash e_1 \rightsquigarrow v, H' \mid (q, q')}{V, H \vdash \text{match } x \text{ with } \mid [] \to e_1 \mid \mathrm{tree}(x_0, x_1, x_2) \to e_2 \rightsquigarrow v, H' \mid (q, q')}\ \text{E-MatchLeaf}$$

$$\frac{\begin{array}{c} V(x) = l \quad H(l) = \mathrm{tree}(v_0, v_1, v_2) \\ V \cup \{x_i \to v_i, i = 0, 1, 2\}, H \vdash e_2 \rightsquigarrow v, H' \mid (q, q') \end{array}}{V, H \vdash \text{match } x \text{ with } \mid [] \to e_1 \mid \mathrm{tree}(x_0, x_1, x_2) \to e_2 \rightsquigarrow v, H' \mid (q, q')}\ \text{E-MatchNode}$$

我有点不明白的是, 为什么一定要显式加个 Heap 呢? 没理解原文 formalization 的动机, 可能是 [JH03] 最开始是在分析堆空间消耗吗?

## 线性资源

本节中, 我们首先定义带有线性资源注释的类型 $A$, 然后定义 $a \in [\![A]\!]$ 所具有的势能, 即势能函数 $\Phi(a : A)$, 然后给出资源相关的子类型关系 $A <: B$, 以及对应的资源语义, 最后给出类型推导规则, 我们忽略了 Soundness 证明, 这只需归纳法便可, 只是稍显复杂.

### 资源类型与势能函数

我们为简单类型 $A \in \mathrm{Type}$ 扩展资源注释, 使其成为资源类型 $A \in \mathrm{ResType}$, 具体来说, 我们只动 $\mathrm{List}\ A$ 和 $\mathrm{Tree}\ A$, 扩充成为 $\mathrm{List}^p A, \mathrm{Tree}^p A$, 其中 $p \in \mathbb{Q}_0^+$.

对于 $a \in \mathrm{Semantics}, A \in \mathrm{ResType}$, 假定 $a \in [\![A]\!]$, 或写作 $a : A$, 可以定义势能函数$\Phi(a : A)$

$$\begin{aligned} \Phi(l : \mathrm{List}^p A) &:= np + \Sigma_{k=1}^n \Phi(a_n : A) \text{ where } l &= [a_1, ..., a_n] \\ \Phi(t : \mathrm{Tree}^p A) &:= np + \Sigma_{k=1}^n \Phi(a_n : A) \text{ where } \mathrm{po}(t) &= [a_1, ..., a_n] \end{aligned}$$

势能函数定义在静态语义上, 可拓展到动态语义, 设 $H \vdash v \to a : A$, 则 $\Phi_H(v : A) := \Phi(a : A)$.

其实有等价定义, 这一定义更加归纳定义:

$$\Phi([] : \mathrm{List}^p A) \coloneqq 0$$
$$\Phi(a :: l : \mathrm{List}^p A) \coloneqq p + \Phi(l : \mathrm{List}^p A) + \Phi(a : A)$$

$$\Phi(\bullet : \mathrm{Tree}^p A) \coloneqq 0$$
$$\Phi(\mathrm{tree}(a, t_1, t_2) : \mathrm{Tree}^p A) \coloneqq p + \Phi(t_1 : \mathrm{Tree}^p A) + \Phi(t_2 : \mathrm{Tree}^p A) + \Phi(a : A)$$

由于引入了资源注释, 因此类型继承了资源上的偏序关系(之对偶), 从文法上, $A <: B$ 如下定义:

$$\mathrm{List}^p A <: \mathrm{List}^q B \coloneqq A <: B \wedge p \geq q$$
$$\mathrm{Tree}^p A <: \mathrm{Tree}^q B \coloneqq A <: B \wedge p \geq q$$

从资源语义上, 有定理: 如果 $A <: B$, 且 $a : A$, 那么 $a : B$ 且 $\Phi(a : A) \geq \Phi(a : B)$.

注意, $A <: B$对应了$\Phi(a : A) \geq \Phi(a : B)$ 是反变的, 因为 $\Phi(a : A)$ 势能较大符合 Liskov 替换原则.

### 类型推导规则

我们直接写算法式的类型推导规则 $\Sigma, \Gamma \vdash e : A \mid (q, q')$, 其中$\Sigma : \mathrm{FnId} \to \mathrm{ResType}$ 是函数签名集合, $\Gamma : \mathrm{Var} \to \mathrm{ResType}$ 是赋型环境(Typing Context):

$$\frac{}{\Sigma, \Gamma \vdash \mathrm{tick}\ (q, q') \rightsquigarrow () \mid (q, q')}\ \text{T-Tick}$$

$$\frac{\Gamma(x) = \mathrm{List}^p A \quad \Sigma, \Gamma \vdash e_1 : B \mid (q_1, q_1') \quad \boxed{q \geq q_1, q_1' \leq q'} \quad \Sigma, \Gamma \cup \{x_h \to A, x_t \to \mathrm{List}^p A\} \vdash e_2 : B \mid (q_2, q_2') \quad \boxed{q + p \geq q_2, q_2' \leq q'}}{\Sigma, \Gamma \vdash \mathrm{match}\ x\ \mathrm{with} \mid [] \to e_1 \mid x_h :: x_t \to e_2 : B \mid (q, q')}\ \text{T-MatchList}$$

$$\frac{\Gamma(x) = \mathrm{Tree}^p A \quad \Sigma, \Gamma \vdash e_1 : B \mid (q_1, q_1') \quad \boxed{q \geq q_1, q_1' \leq q'} \quad \Sigma, \Gamma \cup \{x_0 \to A, x_1 \to \mathrm{Tree}^p A, x_2 \to \mathrm{Tree}^p A\} \vdash e_2 : B \mid (q_2, q_2') \quad \boxed{q + p \geq q_2, q_2' \leq q'}}{\Sigma, \Gamma \vdash \mathrm{match}\ x\ \mathrm{with} \mid \bullet \to e_1 \mid \mathrm{tree}(x_0, x_1, x_2) \to e_2 : B \mid (q, q')}\ \text{T-MatchTree}$$

为 $\boxed{\cdots}$ 圈住的是线性约束, 可被线性规划求解器求解, 目标函数通过启发式给出.

## 单变量多项式资源

### 资源类型与势能函数

类似地, 我们仍然要拓展资源注释, 首先考虑 $\vec{q} = (q_1, ..., q_n) \in (\mathbb{Q}_0^+)^n$, 一般地对于 $k \geq n + 1$, 令 $q_k = 0$, 即 $\vec{q} \in (\mathbb{Q}_0^+)^\infty$, 后续 0 分量忽略不写, 如是将 List $A$ 和 Tree $A$ 扩充成为 $\mathrm{List}^{\vec{q}} A, \mathrm{Tree}^{\vec{q}} A$.

$\vec{q} \in (\mathbb{Q}_0^+)^\infty$的定义十分方便, 有时我们直接写作 $\vec{q} = (q_k), k \in \mathbb{N}^+$, 定义如下运算, :

$$(p_k) + (q_k) = \qquad (p_k + q_k)$$
$$(p_k) \leq (q_k) \Leftrightarrow \forall k \in \mathbb{N}^+, p_k \leq q_k$$

二者共同替代了类型推导规则中的 $p + q$ 与 $p \leq q$, 而后者导出了子类型关系 $A <: B$:

$$\mathrm{List}^{\vec{p}} A <: \mathrm{List}^{\vec{q}} B \coloneqq A <: B \wedge \vec{p} \geq \vec{q}$$
$$\mathrm{Tree}^{\vec{p}} A <: \mathrm{Tree}^{\vec{q}} B \coloneqq A <: B \wedge \vec{p} \geq \vec{q}$$

以及对应的资源语义定理: 如果 $a : A$ 且 $A <: B$, 那么 $a : B$ 且 $\Phi(a : A) \geq \Phi(a : B)$.

下面首先定义如下的 additive shift 算子 $\lhd$, 再定义势能函数 $\Phi(a : A)$

$$\begin{aligned} \lhd : (\mathbb{Q}_0^+)^n &\quad \to (\mathbb{Q}_0^+)^n \\ \lhd \quad (q_1, ..., q_n) &\ = (q_1 + q_2, ..., q_{n-1} + q_n, q_n) \end{aligned}$$

借由此, 我们可以定义出(归纳定义的)势能函数 $\Phi(a : A)$:

$$\begin{aligned} \Phi\left([] : \mathrm{List}^{\vec{p}} A\right) &:= 0 \\ \Phi\left(a :: l : \mathrm{List}^{\vec{p}} A\right) &:= p_1 + \Phi\left(l : \mathrm{List}^{\lhd(\vec{p})} A\right) + \Phi(a : A) \end{aligned}$$

$$\begin{aligned} \Phi\left(\bullet : \mathrm{Tree}^{\vec{p}} A\right) &:= 0 \\ \Phi\left(\mathrm{tree}(a, t_1, t_2) : \mathrm{Tree}^{\vec{p}} A\right) &:= p_1 + \Phi\left(t_1 : \mathrm{Tree}^{\lhd(\vec{p})} A\right) + \Phi\left(t_2 : \mathrm{Tree}^{\lhd(\vec{p})} A\right) + \Phi(a : A) \end{aligned}$$

类似地, 只是与线性那节时叙述顺序相反, 我们有如下等价定义,
考虑$\vec{p} = (p_1, ..., p_m)$, 记 $h$ 是树 $t$ 的高度, $n_i$ 是树 $t$ 中**深度**为 $i$ 的结点数目(叶子深度为$h$)

$$1.\Phi\left(l : \mathrm{List}^{\vec{p}} A\right) = \Sigma_{k=1}^{m} \binom{n}{k} p_k + \Sigma_{k=1}^{n} \Phi(a_k : A) \qquad \text{where } l = [a_1, ..., a_n]$$

$$2.\Phi\left(t : \mathrm{Tree}^{\vec{p}} A\right) = \Sigma_{i=1}^{h} n_i \Sigma_{k=1}^{i} \binom{i-1}{k-1} p_k + \Sigma_{k=1}^{n} \Phi(a_k : A) \text{ where } \mathrm{po}(t) = [a_1, ..., a_n]$$

在证明前, 我们强调: 由于 $\binom{n}{k}$ 是二项式系数, 因此 $\Sigma_{k=1}^{m} \binom{n}{k} p_k$ 事实上是一个$m$次**多项式**.

证明:

1. 记 $\varphi(n, \vec{p}) := \Sigma_{k=1}^{m} \binom{n}{k} p_k$, 关于 $\Phi\left(l : \mathrm{List}^{\vec{p}} A\right)$, $l = []$ 显然; 考虑 $n > 0$ 的情形, 展开定义 :

$$\begin{aligned} \Phi\left(a_1 :: l : \mathrm{List}^{\vec{p}} A\right) &:= p_1 + \Phi\left(l : \mathrm{List}^{\lhd(\vec{p})} A\right) + \Phi(a_1 : A) \\ &= p_1 + \varphi(n-1, \lhd(\vec{p})) + \Sigma_{k=1}^{n-1} \Phi(a_{k+1} : A) + \Phi(a_1 : A) \\ &= p_1 + \varphi(n-1, \lhd(\vec{p})) + \Sigma_{k=1}^{n} \Phi(a_k : A) \end{aligned}$$

往证 $p_1 + \varphi(n-1, \lhd(\vec{p})) = \varphi(n, \vec{p})$, 这由如下给出:

$$\begin{aligned} p_1 + \varphi(n-1, \lhd(\vec{p})) &= p_1 + \Sigma_{k=1}^{m-1} \binom{n-1}{k} (p_k + p_{k+1}) + \binom{n-1}{m} p_m \\ &= p_1 + \Sigma_{k=1}^{m-1} \binom{n-1}{k} p_k + \Sigma_{k=1}^{m-1} \binom{n-1}{k} p_{k+1} + \binom{n-1}{m} p_m \\ &= p_1 + \binom{n-1}{1} p_1 + \Sigma_{k=2}^{m-1} \binom{n-1}{k} p_k + \binom{n-1}{m} p_m + \Sigma_{k=1}^{m-1} \binom{n-1}{k} p_{k+1} \\ &= n p_1 + \Sigma_{k=2}^{m} \binom{n-1}{k} p_k + \Sigma_{k=1}^{m-1} \binom{n-1}{k} p_{k+1} \\ &= n p_1 + \Sigma_{k=1}^{m-1} \binom{n-1}{k+1} p_{k+1} + \Sigma_{k=1}^{m-1} \binom{n-1}{k} p_{k+1} \end{aligned}$$

$$\because \binom{n}{k} = \binom{n-1}{k+1} + \binom{n-1}{k}$$

$$\begin{aligned} &= \binom{n}{1} p_1 + \Sigma_{k=1}^{m-1} \binom{n}{k+1} p_{k+1} = \binom{n}{1} p_1 + \Sigma_{k=2}^{m} \binom{n}{k} p_k \\ &= \Sigma_{k=1}^{m} \binom{n}{k} p_k = \varphi(n, \vec{p}) \end{aligned}$$

2. 记$\psi(t, \vec{p}) := \Sigma_{i=1}^{h} n_i \Sigma_{k=1}^{i} \binom{i-1}{k-1} p_k$, 对高度 $h$ 归纳, $h = 0$ 时显然; $h > 0$ 展开定义 :

$$\begin{aligned}\Phi(\mathrm{tree}(a,t_1,t_2)) &\coloneqq p_1+\Phi\big(t_1:\mathrm{Tree}^{\lhd(\vec{p})}A\big)+\Phi\big(t_2:\mathrm{Tree}^{\lhd(\vec{p})}A\big)+\Phi(a:A)\\ &= p_1+\psi(t_1,\lhd(\vec{p}))+\psi(t_2,\lhd(\vec{p}))+\Sigma_{k=1}^{n}\Phi(a_n:A)\end{aligned}$$

由于$n_1=1$, 且在子树 $t_1, t_2$ 中, $n_{1,i}+n_{2,i}=n_{i+1}, n_{1,h}=n_{2,h}=0$:

$$\begin{aligned}p_1+\psi(t_1,\lhd(\vec{p}))+\psi(t_2,\lhd(\vec{p})) &= p_1+\Sigma_{i=1}^{h-1}n_{i+1}\Sigma_{k=1}^{i}\binom{i-1}{k-1}(p_k+p_{k+1})\\ &= p_1+\Sigma_{i=2}^{h}n_i\Sigma_{k=1}^{i-1}\binom{i-2}{k-1}(p_k+p_{k+1})\\ &= p_1+\Sigma_{i=2}^{h}n_i\left(p_1+p_i+\Sigma_{k=2}^{i-1}p_k\left(\binom{i-2}{k-2}+\binom{i-2}{k-1}\right)\right)\\ &= p_1+\Sigma_{i=2}^{h}n_i\left(p_1+p_i+\Sigma_{k=2}^{i-1}p_k\binom{i-1}{k-1}\right)\\ &= p_1+\Sigma_{i=2}^{h}n_i\Sigma_{k=1}^{i}p_k\binom{i-1}{k-1}\\ &= \Sigma_{i=1}^{h}n_i\Sigma_{k=1}^{i}\binom{i-1}{k-1}p_k=\psi(t,\vec{p})\end{aligned}$$

□

**定理 0.1**: 设树 $t$ 满足 $\mathrm{po}(t)=[a_1,...,a_n]$, 高度为 $h$, $\vec{p}=(p_1,...,p_m)$:
1. $\Phi\big(t:\mathrm{Tree}^{\vec{p}}A\big)\le\varphi(n,\vec{p})+\Sigma_{k=1}^{n}\Phi(a_k:A)$;
2. $\Phi\big(t:\mathrm{Tree}^{\vec{p}}A\big)\le\Sigma_{k=1}^{m}p_k n(h-1)^{k-1}+\Sigma_{k=1}^{n}\Phi(a_k:A)$;

证明:
1. 对 $n$ 归纳,

□

## 类型推导规则

类型推到规则与上一节十分类似, 只是子结构的类型应当进行 $\lhd$ 操作, $\boxed{\ldots}$ 强调变化部分:

$$\frac{\begin{array}{c}\Gamma(x)=\mathrm{List}^{\boxed{\vec{p}}}A \qquad \Sigma,\Gamma\vdash e_1:B\mid(q_1,q_1') \qquad q\ge q_1, q_1'\le q'\\ \Sigma,\Gamma\cup\left\{x_h\to A, x_t\to\mathrm{List}^{\boxed{\lhd(\vec{p})}}A\right\}\vdash e_2:B\mid(q_2,q_2') \qquad q+\boxed{p_1}\ge q_2, q_2'\le q'\end{array}}{\Sigma,\Gamma\vdash\mathrm{match}\ x\ \mathrm{with}\mid[]\to e_1\mid x_h::x_t\to e_2:B\mid(q,q')}\text{T-MatchList}$$

$$\frac{\begin{array}{c}\Gamma(x)=\mathrm{Tree}^{\boxed{\vec{p}}}A \qquad \Sigma,\Gamma\vdash e_1:B\mid(q_1,q_1') \qquad q\ge q_1, q_1'\le q'\\ \Sigma,\Gamma\cup\left\{x_0\to A, x_1\to\mathrm{Tree}^{\boxed{\lhd(\vec{p})}}A, x_2\to\mathrm{Tree}^{\boxed{\lhd(\vec{p})}}A\right\}\vdash e_2:B\mid(q_2,q_2')\\ q+\boxed{p_1}\ge q_2, q_2'\le q'\end{array}}{\Sigma,\Gamma\vdash\mathrm{match}\ x\ \mathrm{with}\mid\bullet\to e_1\mid\mathrm{tree}(x_0,x_1,x_2)\to e_2:B\mid(q,q')}\text{T-MatchTree}$$

# 多变量多项式资源

TODO...

# 一般数据类型

TODO...