The Implementation of a Static Prediction of Heap Space Usage for First-Order Functional Programs

Based upon joint work with Martin Hofmann

Steffen Jost
Institut für Informatik
LMU München

12.January, 2004

The Task:

Determine the memory usage of given functional program prior to runtime. No specific resource annotations present.

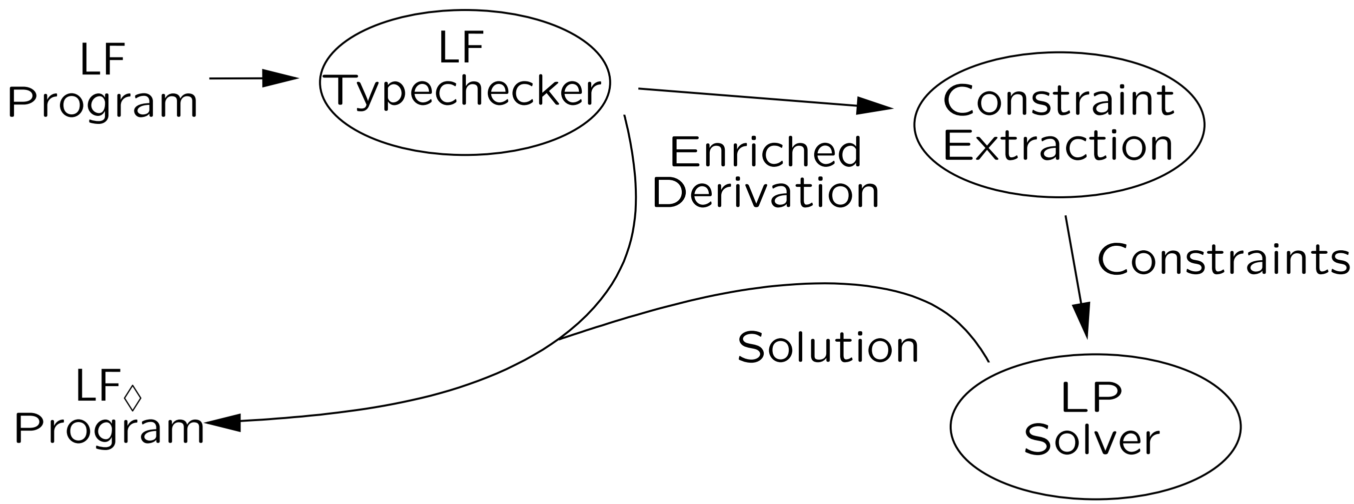
Our Solution:

Derive set of linear inequalities over $\mathbb{Q}$ from typing derivation. Solutions provide bounds on heap space usage as linear function of input size. (Theoretical work presented at POPL'03)

Application:

Proof-carrying code for guarantees on resource consumption
EU project: Mobile Resource Guarantees, Edinburgh-Munich

Overview inference process:



LF is:

- non-linear, first-order, functional, monomorphic, let-normal
- automatically generated from Camelot
  (ML-dialect with memory primitives for deallocation)
- syntactically equal to $LF_\Diamond$, but types yield linear bounds on heap space consumption

Nested resource annotations:

Suppose

```
f: list[list[int,#,1|0],#,2|0],3 -> list[int,#,4|0],5;
```

Evaluating $\mathtt{f}\Big([l_1, \ldots, l_m]\Big)$

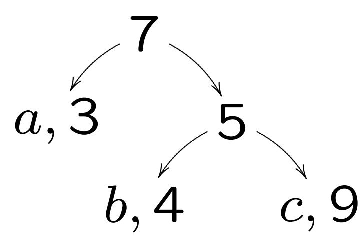
- requires at most $3 + 2m + 1\Sigma|l_i|$ extra heap cells and
- leaves at least $5 + 4|\mathtt{f}(l)|$ unused memory cells

Annotations are merely weight factors:

- No direct reference to length/size of data
  (as compared to sized types [Hughes & Pareto '99,'02])
- Rational values allowed

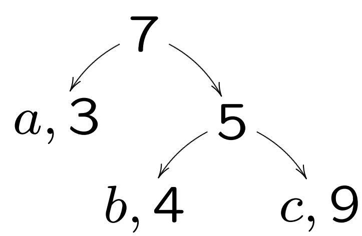
Calculation Examples:

```
type list= Cons(*1*) of int * list | Nil(*0*)
type tree= Leaf(*1*) of char*int |Node(*1*) of int*tree*tree
```

Enriched Type	Instance	Alloc.	Resvd.	Σ
`list[int,#,0|0]`	[1,2,3,4,5]	5	0	5
`list[int,#,1|0]`	[1,2,3,4]	4	4	8
`list[int,#,1|0]`	[1,2,3,4,5]	5	5	10
`list[list[int,#,2|0],#,0|0]`	[[1],[2,3,4]]	6	8	14
`list[list[int,#,2|0],#,3|0]`	[[1],[2,3,4]]	6	14	20
`list[list[int,#,2|1],#,2|0]`	[[1],[2,3,4]]	6	14	20
`tree[char,int,2|int,#,#,3]`		5	12	17

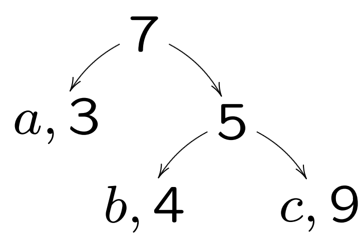
Calculation Examples:

```
type list= Cons(*1*) of int * list | Nil(*0*)
type tree= Leaf(*1*) of char*int |Node(*1*) of int*tree*tree
```

Enriched Type	Instance	Alloc.	Resvd.	Σ
`list[int,#,0|0]`	[1,2,3,4,5]	5	0	5
`list[int,#,1|0]`	[1,2,3,4]	4	4	8
`list[int,#,2|0]`	[1,2,3,4,5]	5	10	15
`list[list[int,#,2|0],#,0|0]`	[[1],[2,3,4]]	6	8	14
`list[list[int,#,2|0],#,3|0]`	[[1],[2,3,4]]	6	14	20
`list[list[int,#,2|1],#,2|0]`	[[1],[2,3,4]]	6	14	20
`tree[char,int,2|int,#,#,3]`		5	12	17

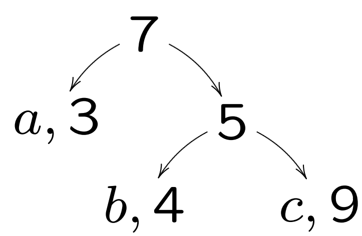
Calculation Examples:

```
type list= Cons(*2*) of int * list | Nil(*0*)
type tree= Leaf(*1*) of char*int |Node(*1*) of int*tree*tree
```

Enriched Type	Instance	Alloc.	Resvd.	Σ
`list[int,#,0|0]`	[1,2,3,4,5]	10	0	10
`list[int,#,1|0]`	[1,2,3,4]	8	4	12
`list[int,#,2|0]`	[1,2,3,4,5]	10	10	20
`list[list[int,#,2|0],#,0|0]`	[[1],[2,3,4]]	12	8	20
`list[list[int,#,2|0],#,3|0]`	[[1],[2,3,4]]	12	14	26
`list[list[int,#,2|1],#,2|0]`	[[1],[2,3,4]]	12	14	26
`tree[char,int,2|int,#,#,3]`		5	12	17

Calculation Examples:

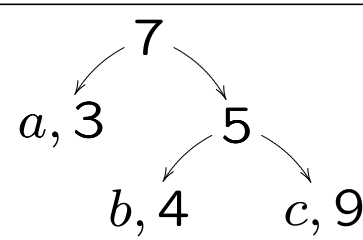
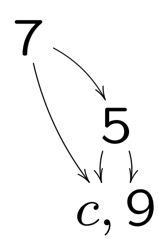
```
type list= Cons(*2*) of int * list | Nil(*0*)
type tree= Leaf(*2*) of char*int |Node(*3*) of int*tree*tree
```

Enriched Type	Instance	Alloc.	Resvd.	Σ
list[int,#,0|0]	[1,2,3,4,5]	10	0	10
list[int,#,1|0]	[1,2,3,4]	8	4	12
list[int,#,2|0]	[1,2,3,4,5]	10	10	20
list[list[int,#,2|0],#,0|0]	[[1],[2,3,4]]	12	8	20
list[list[int,#,2|0],#,3|0]	[[1],[2,3,4]]	12	14	26
list[list[int,#,2|1],#,2|0]	[[1],[2,3,4]]	12	14	26
tree[char,int,2|int,#,#,3]		12	12	24

Shared Data

Annotations are linearly distributed in shared data:

```
type tree=
        Leaf(*1*) of char*int | Node(*1*) of int*tree*tree
```

Enriched Type	Instance	Alloc.	Resvd.	Σ
`tree[char,int,2|int,#,#,3]`		5	12	17
`tree[char,int,2|int,#,#,3]`		3	12	15

Contributions:

- Efficient and automatic analysis
  - Inference amounts to solving linear inequalities over $\mathbb{Q}$
  - Yields linear arithmetic expressions for heap usage
  - Modular: only a function and its sub-functions must be examined at once

- Further aspects
  - Shared data structures accounted for
  - Manual intervention possible if desired
  - Slack in solution of LP reveals a computational branch leaking memory

Integer solutions:

- Allow evaluation without memory management support:
  - $\mathrm{LF}_{\Diamond}^{\mathrm{lin},\mathbb{N}}$ translates to malloc-free C via LFPL [MH'00]
  - LFPL: all non size-increasing functions in ETIME = LINSPACE + unbounded Stack [Cook'72]

- Results on complexity of finding solutions in $\mathbb{N}$:
  - Computing optimal solution NP-hard
  - Finding a solution feasible in linearly typed fragment
  - Finding optimal toplevel annotations feasible

Completeness of inference for LFPL requires:

- Prohibit "borrowing" from non-termination:
  - Function return types must not contain surface resources
- Canonical resource placement:
  - $(\mathsf{bool} \otimes \Diamond) \approx (\Diamond \otimes \mathsf{bool}) \rightsquigarrow (\mathsf{bool}, 1)$
  - Only one branch of a sum may contain surface resources
  - Trees with unlabeled leafs only

Implementation

Syntax closely related to Camelot/Caml, except:

- Fully sequentialized (let normal form)
- No polymorphism
- No parameterized types
- Expects well-typed input

Theses issues already addressed by Camelot-Compiler!

Current problems:

- Polymorphism (esp. in resource annotations)
- Higher-order functions
- Non-linear bounds
- Tighter bounds (GC useless here)
- Automation of pattern match mode: destructive/read-only
- Feed detected slack back to source code
- Functional Objects
- Mutual recursive datatypes lead to non-termination
- Not enough examples (e.g. subtyping)

Rational Annotations

Function $\texttt{tos}$ *shall* replace each third element of a list

$$\begin{aligned}
\texttt{tos}\Big([1,2,3,4,5]\Big) &= \texttt{tpo}\Big(\texttt{sec}\Big([1,2,3,4,5]\Big)\Big)\\
\texttt{tos}\Big([1,2,3,4,5]\Big) &= [1,2,1,4,5,4]
\end{aligned}$$

$$\begin{aligned}
\texttt{sec}\Big([1,2,3,4,5]\Big) &= [1,2,4,5]\\
\texttt{tpo}\Big([1,2,4,5]\Big) &= [1,2,1,4,5,4]
\end{aligned}$$

- Length of input list changes in between

- Set $\mathsf{SIZE}\,(\mathsf{int}) = 2$ or use $\mathsf{int} \otimes \mathsf{int}$

Rational Annotations

$$
\begin{aligned}
\mathtt{tos}(l) &= \mathtt{tpo}(\mathtt{sec}(l)) \\
\\
\mathtt{sec}(\mathsf{Nil}) &= \mathsf{Nil} \\
\mathtt{sec}(\mathsf{Cons}(h_1, \mathsf{Nil})) &= \mathsf{Cons}(h_1, \mathsf{Nil}) \\
\mathtt{sec}(\mathsf{Cons}(h_1, \mathsf{Cons}(h_2, \mathsf{Nil}))) &= \mathsf{Cons}(h_1, \mathsf{Cons}(h_2, \mathsf{Nil})) \\
\mathtt{sec}(\mathsf{Cons}(h_1, \mathsf{Cons}(h_2, \mathsf{Cons}(h_3, t)))) &= \mathsf{Cons}(h_1, \mathsf{Cons}(h_2, \mathtt{sec}(t))) \\
\\
\mathtt{tpo}(\mathsf{Nil}) &= \mathsf{Nil} \\
\mathtt{tpo}(\mathsf{Cons}(h_1, \mathsf{Nil})) &= \mathsf{Cons}(h_1, \mathsf{Nil}) \\
\mathtt{tpo}(\mathsf{Cons}(h_1, \mathsf{Cons}(h_2, t))) &= \mathsf{Cons}(h_1, \mathsf{Cons}(h_2, \mathsf{Cons}(h_1, \mathtt{tpo}(t))))
\end{aligned}
$$

Rational Annotations

$$\texttt{tos} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$
$$\texttt{sec} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_2), x_2$$
$$\texttt{tpo} : \mathsf{list}(\mathsf{int}, l_2), x_2 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$

Simplification and Elimination leads to

$$\begin{aligned}
x_1 &\geq x_2\\
x_1 &\geq -(3 + l_1) + (3 + l_2) + x_2\\
x_1 &\geq -2(3 + l_1) + 2(3 + l_2) + x_2\\
x_1 &\geq -3(3 + l_1) + 2(3 + l_2) + x_1 - x_2 + x_2\\
x_2 &\geq x_3\\
x_2 &\geq -(3 + l_2) + (3 + l_3) + x_3\\
x_2 &\geq -2(3 + l_2) + 3(3 + l_3) + x_2 - x_3 + x_3
\end{aligned}$$

plus nonnegativity constraints

Rational Annotations

$$\texttt{tos} : (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0)$$
$$\texttt{sec} : (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0)$$
$$\texttt{tpo} : (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0)$$

allocated + reserved

$[1,2,3,4,5] : \mathsf{list}(\mathsf{int}, 0)$	$5 \cdot 3 + 5 \cdot 0 + 3 = 18$
$[1,2,4,5] : \mathsf{list}(\mathsf{int}, \tfrac{3}{2})$	$4 \cdot 3 + 4 \cdot \tfrac{3}{2} + 0 = 18$
$[1,2,1,4,5,4] : \mathsf{list}(\mathsf{int}, 0)$	$6 \cdot 3 + 6 \cdot 0 + 0 = 18$

Rational Annotations

$$\texttt{tos} : (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0)$$
$$\texttt{sec} : (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0)$$
$$\texttt{tpo} : (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0)$$

allocated + reserved

$$[1,2,3,4] : \mathsf{list}(\mathsf{int}, 0) \qquad 4 \cdot 3 + 4 \cdot 0 + 3 = 15$$
$$[1,2,4] : \mathsf{list}(\mathsf{int}, \tfrac{3}{2}) \qquad 3 \cdot 3 + 3 \cdot \tfrac{3}{2} + 0 = \tfrac{27}{2}$$
$$[1,2,1,4] : \mathsf{list}(\mathsf{int}, 0) \qquad 4 \cdot 3 + 4 \cdot 0 + 0 = 12$$

Rational Annotations

$$\texttt{tos} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$
$$\texttt{sec} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_2), x_2$$
$$\texttt{tpo} : \mathsf{list}(\mathsf{int}, l_2), x_2 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$

Simplification and Elimination leads to

$$x_1 \geq x_2$$
$$x_1 \geq -(3 + l_1) + (3 + l_2) + x_2$$
$$x_1 \geq -2(3 + l_1) + 2(3 + l_2) + x_2$$
$$x_1 \geq -3(3 + l_1) + 2(3 + l_2) + x_1 - x_2 + x_2$$
$$x_2 \geq x_3$$
$$x_2 \geq -(3 + l_2) + (3 + l_3) + x_3$$
$$x_2 \geq -2(3 + l_2) + 3(3 + l_3) + x_2 - x_3 + x_3$$

plus nonnegativity constraints

Rational Annotations

$$\texttt{tos} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$
$$\texttt{sec} : \mathsf{list}(\mathsf{int}, l_1), x_1 \to \mathsf{list}(\mathsf{int}, l_2), x_2$$
$$\texttt{tpo} : \mathsf{list}(\mathsf{int}, l_2), x_2 \to \mathsf{list}(\mathsf{int}, l_3), x_3$$

Simplification and Elimination leads to

$$\begin{array}{llll}
x_1 \geq x_2 & & & \\
x_1 \geq -\ l'_1 & +\ l'_2 & + x_2 & \\
x_1 \geq -2\ l'_1 & +2\ l'_2 & + x_2 & l'_1 \geq 3 \\
x_1 \geq -3\ l'_1 & +2\ l'_2 & + x_1 - x_2 + x_2 & l'_2 \geq 3 \\
x_2 \geq x_3 & & & l'_3 \geq 3 \\
x_2 \geq -\ l'_2 & +\ l'_3 & + x_3 & \\
x_2 \geq -2\ l'_2 & +3\ l'_3 & + x_2 - x_3 + x_3 &
\end{array}$$

plus nonnegativity constraints

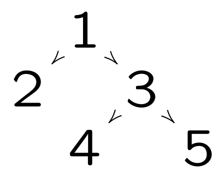
Rational Annotations

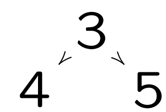
$$
\begin{aligned}
\texttt{tos} &: (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0) \\
\texttt{sec} &: (\mathsf{list}(\mathsf{int}, 0), 3) \longrightarrow (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0) \\
\texttt{tpo} &: (\mathsf{list}(\mathsf{int}, \tfrac{3}{2}), 0) \longrightarrow (\mathsf{list}(\mathsf{int}, 0), 0)
\end{aligned}
$$

versus

$$
\begin{aligned}
\texttt{tos} &: (\mathsf{list}(\mathsf{int}, 3), 6) \longrightarrow (\mathsf{list}(\mathsf{int}, 3), 0) \\
\texttt{sec} &: (\mathsf{list}(\mathsf{int}, 3), 6) \longrightarrow (\mathsf{list}(\mathsf{int}, 6), 0) \\
\texttt{tpo} &: (\mathsf{list}(\mathsf{int}, 6), 0) \longrightarrow (\mathsf{list}(\mathsf{int}, 3), 0)
\end{aligned}
$$

Example Pathlist: Sharing

$\texttt{pathacc}\Bigg($  $, [\,]\Bigg)$

$= \texttt{pathacc}\big(\mathsf{Leaf}(2), [1]\big) +\!\!+ \; \texttt{pathacc}\Big($  $, [1]\Big)$

$$= \big[[2,1]\big] +\!\!+ \; \texttt{pathacc}\big(\mathsf{Leaf}(4), [3,1]\big) +\!\!+ \; \texttt{pathacc}\big(\mathsf{Leaf}(5), [3,1]\big)$$

$$= \big[[2,1], [4,3,1], [5,3,1]\big]$$

Example Pathlist: Sharing

$$
\begin{aligned}
\mathtt{pathlist} &: \mathsf{tree}(\mathsf{A}\quad) \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}\quad)\quad) \\
\mathtt{pathlist}(t) &= \mathtt{pathacc}(t, \mathsf{Nil})
\end{aligned}
$$

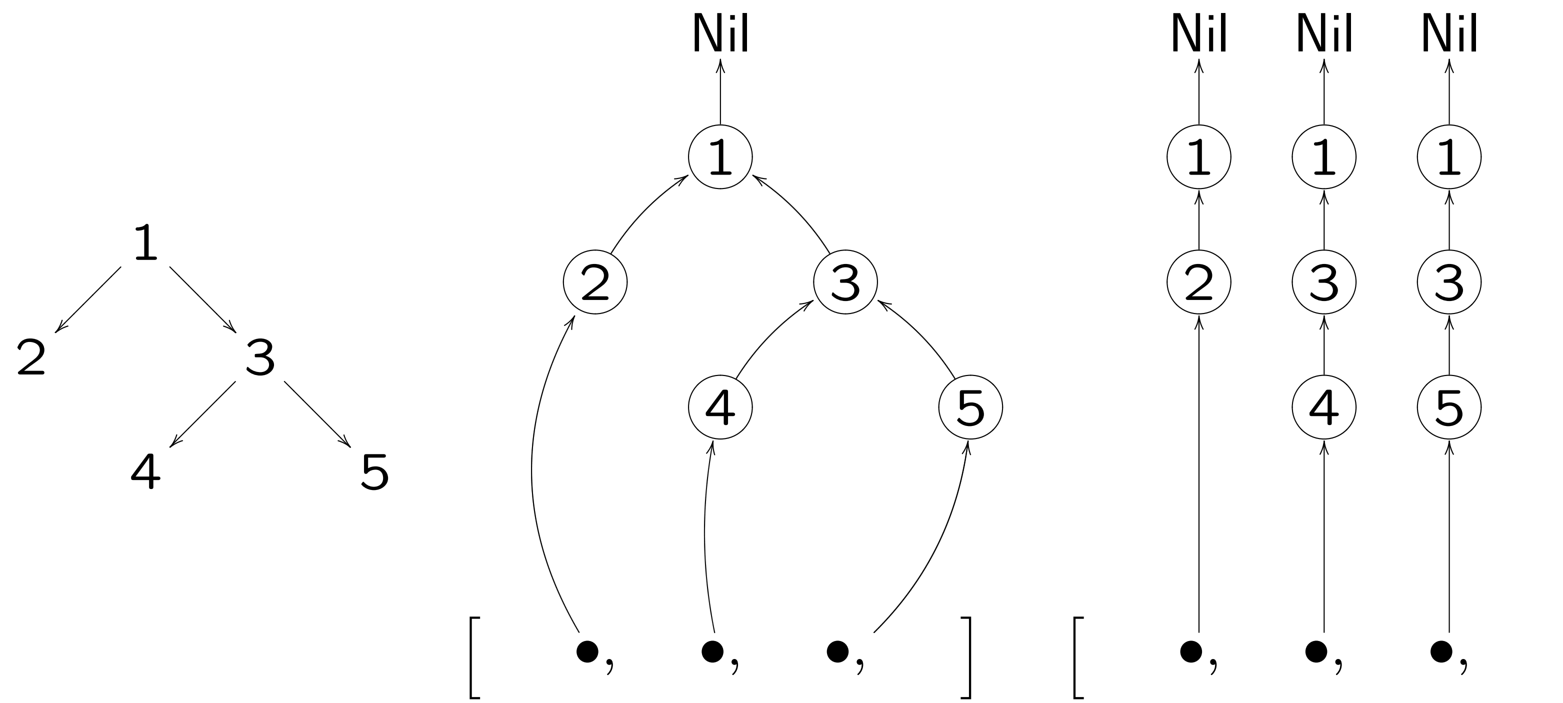
$$
\begin{aligned}
\mathtt{pathacc} &: \mathsf{tree}(\mathsf{A}\quad), \mathsf{list}(\mathsf{A}\quad) \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}\quad)\quad) \\
\mathtt{pathacc}(\mathsf{Leaf}(a), c) &= \mathsf{Cons}\big(\mathsf{Cons}(a, c), \mathsf{Nil}\big) \\
\mathtt{pathacc}(\mathsf{Node}(a, l, r), c) &= \mathsf{let}\ x{=}\mathsf{Cons}(a, c)\ \mathsf{in} \\
&\qquad \mathtt{pathacc}\big(l, x\big) +\!\!+\, \mathtt{pathacc}\big(r, x\big)
\end{aligned}
$$

Example Pathlist: Sharing

$$\begin{aligned}
\texttt{pathlist} \quad &: \quad \mathsf{tree}(\mathsf{A}, 1), 2 \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}, 0), 0), 0 \\
\texttt{pathlist}(t) \quad &= \quad \texttt{pathacc}(t, \mathsf{Nil})
\end{aligned}$$

$$\begin{aligned}
\texttt{pathacc} \quad &: \quad \mathsf{tree}(\mathsf{A}, 1), \mathsf{list}(\mathsf{A}, 0), 2 \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}, 0), 0), 0 \\
\texttt{pathacc}(\mathsf{Leaf}(a), c) \quad &= \quad \mathsf{Cons}\Big(\mathsf{Cons}(a, c), \mathsf{Nil}\Big) \\
\texttt{pathacc}(\mathsf{Node}(a, l, r), c) \quad &= \quad \mathsf{let}\ x{=}\mathsf{Cons}(a, c)\ \mathsf{in} \\
&\qquad \texttt{pathacc}\Big(l, x\Big) +\!\!+ \texttt{pathacc}\Big(r, x\Big)
\end{aligned}$$

Example Pathlist: Sharing



cells	$2 \cdot 3 + 3 = 9$	$10 + 6$	$16 + 6$
reserved	$5 + 2$	0	

Pathlist: Read-only versus destructive pattern match

$$\texttt{pathlist} \;:\; \mathsf{tree}(\mathsf{A}, t), x \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}, 0), 0), 0$$

$$\texttt{pathacc} \;:\; \mathsf{tree}(\mathsf{A}, t), \mathsf{list}(\mathsf{A}, 0), x \longrightarrow \mathsf{list}(\mathsf{list}(\mathsf{A}, 0), 0), 0$$

	t	x	Constraints	
Destructive	1	2	$t + x \geq 3$	$t + 1 \geq x$
Read-Only	$2 + a$	1	$t + x \geq 3 + a$	$t + x \geq 2x + a + 1$

with $a := \mathsf{SIZE}\,(\mathsf{A})$

This TEXT is normal. $1 + 2 = \vec{v}$ • • •

This TEXT is red. $1 + 2 = \vec{v}$ • • •

This TEXT is green. $1 + 2 = \vec{v}$ • • •

This TEXT is blue. $1 + 2 = \vec{v}$ • • •

This TEXT is yellow. $1 + 2 = \vec{v}$ • • •

This TEXT is darkred. $1 + 2 = \vec{v}$ • • •

This TEXT is darkgreen. $1 + 2 = \vec{v}$ • • •

This TEXT is darkblue. $1 + 2 = \vec{v}$ • • •

This TEXT is darkyellow. $1 + 2 = \vec{v}$ • • •

This TEXT is grey. $1 + 2 = \vec{v}$ • • •

This TEXT is black. $1 + 2 = \vec{v}$ • • •

This TEXT is normal. $1 + 2 = \vec{v}$ • • •