Liveness-Based Garbage Collection

Rahul Asati, Amitabha Sanyal
Indian Institute of Technology, Bombay
Amey Karkare
Indian Institute of Technology, Kanpur
Alan Mycroft
University of Cambridge

September 8th, 2015

MAIN IDEA

- During execution, there are significant amounts of heap allocated data that are *reachable but not live*.
  - Current GCs will retain such data.
- Our idea:
  - We do a liveness analysis and provide GCs with its result.
  - Modify GCs to mark data for retention *only if it is live*.
- Consequences:
  - Fewer cells marked.

Main Idea

- During execution, there are significant amounts of heap allocated data that are *reachable but not live*.
  - Current GCs will retain such data.
- Our idea:
  - We do a liveness analysis and provide GCs with its result.
  - Modify GCs to mark data for retention *only if it is live*.
- Consequences:
  - Fewer cells marked. More garbage collected per collection.

MAIN IDEA

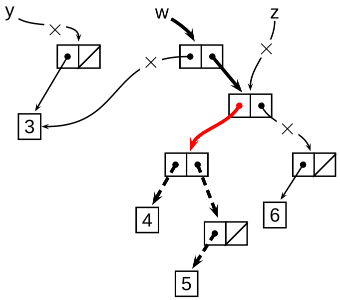
- During execution, there are significant amounts of heap allocated data that are *reachable but not live*.
  - Current GCs will retain such data.
- Our idea:
  - We do a liveness analysis and provide GCs with its result.
  - Modify GCs to mark data for retention *only if it is live*.
- Consequences:
  - Fewer cells marked. More garbage collected per collection. Fewer garbage collections.

MAIN IDEA

- During execution, there are significant amounts of heap allocated data that are *reachable but not live*.
  - Current GCs will retain such data.
- Our idea:
  - We do a liveness analysis and provide GCs with its result.
  - Modify GCs to mark data for retention *only if it is live*.
- Consequences:
  - Fewer cells marked. More garbage collected per collection. Fewer garbage collections.
  - Programs expected to run faster and with smaller heap.

AN EXAMPLE

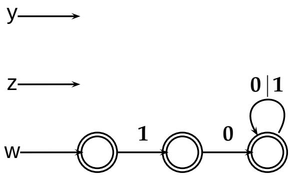
```
(define (append l1 l2)
 (if (null? l1) l2
     (cons (car l1)
           (append (cdr l1) l2))))

(let z ←(cons (cons 4 (cons 5 nil))
              (cons 6 nil)) in
 (let y ← (cons 3 nil) in
     (let w ← (append y z) in
        π:(car (cdr w)))))
```



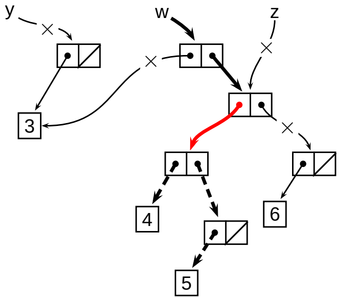
- *Though all cells are reachable at π, a liveness-based GC will retain only the cells pointed by thick arrows.*

Preview of things to come

- Our analysis captures liveness as automata.



Liveness at π



The language analyzed

- First order eager Scheme-like language.
- In Administrative Normal Form (ANF).

$$
\begin{array}{rcl}
p \in \mathit{Prog} & ::= & d_1 \ldots d_n \; e_{\mathsf{main}} \\
d \in \mathit{Fdef} & ::= & (\textbf{define} \; (f \;\; x_1 \;\; \ldots \;\; x_n) \;\; e) \\
e \in \mathit{Expr} & ::= & \left\{ \begin{array}{l} (\textbf{if} \; x \; e_1 \;\; e_2) \\ (\textbf{let} \; x \leftarrow a \; \textbf{in} \; e) \\ (\textbf{return} \; x) \end{array} \right. \\
a \in \mathit{App} & ::= & \left\{ \begin{array}{ll} k & \\ (\textbf{cons} \;\; x_1 \;\; x_2) & \\ (\textbf{car} \;\; x) & (\textbf{cdr} \;\; x) \\ (\textbf{null?} \;\; x) & (+ \;\; x_1 \;\; x_2) \\ (f \;\; x_1 \;\; \ldots \;\; x_n) & \end{array} \right.
\end{array}
$$

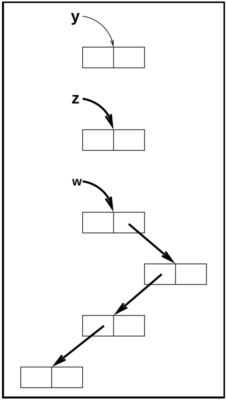
LIVENESS – BASIC CONCEPTS AND NOTATIONS

- *Access paths:* Strings over $\{\mathbf{0}, \mathbf{1}\}$.

  $\mathbf{0}$ – access **car** field
  $\mathbf{1}$ – access **cdr** field

- *Liveness environment:* Maps root variables to set of access paths.

$$\mathsf{L}_i \qquad \begin{cases} y \mapsto \emptyset \\ z \mapsto \{\epsilon\} \\ w \mapsto \{\epsilon, 1, 10, 100\} \end{cases}$$



Notation: We write $\mathsf{L}_i(\mathsf{x})$ as $\mathsf{L}_i^{\mathsf{X}}$

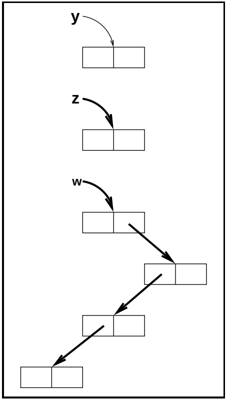
LIVENESS – BASIC CONCEPTS AND NOTATIONS

- *Access paths:* Strings over $\{\mathbf{0}, \mathbf{1}\}$.

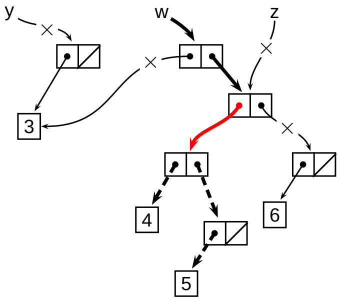
  $\mathbf{0}$ – access **car** field
  $\mathbf{1}$ – access **cdr** field

- *Liveness environment:* Alternate representation.

$$\mathsf{L}_i \quad \begin{cases} \emptyset \ \cup \\ \{z.\epsilon\} \ \cup \\ \{w.\epsilon, w.1, w.10, w.100\} \end{cases}$$



DEMAND



- Demand (notation: σ) is a description of intended use of the result of an expression.
- We assume the demand on the main expresssion to be $(0+1)^*$, which we call σ_{all}.
- The demand on each function body, σ_f, have to be computed.

LIVENESS ANALYSIS – THE BIG PICTURE

```
π_main: (let z ← ... in
          (let y ← ... in
      π9: (let w ← (append y z) in
       π10: (let a ← (cdr w) in
        π11: (let b ← (car a) in
         π12: (return b)))))))
```

```
(define (append l1 l2)
  π1: (let test ← (null? l1) in
   π2: (if test  π3:(return l2)
    π4: (let tl ← (cdr l1) in
     π5: (let rec ← (append tl l2) in
      π6: (let hd ← (car l1) in
       π7: (let ans ← (cons hd rec) in
        π8:  (return ans))))))))
```

Liveness environments:

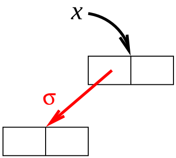
$$L_1 = \ldots$$
$$L_2 = \ldots$$
$$\ldots$$
$$L_9 = \ldots$$
$$L_{10} = \ldots$$

Demand summaries:

$$\sigma_{\mathbf{main}} = \sigma_{all}$$
$$\sigma_{\mathbf{append}} = \ldots$$

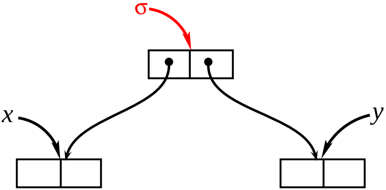
Function summaries:

LIVENESS ANALYSIS OF APPLICATIONS



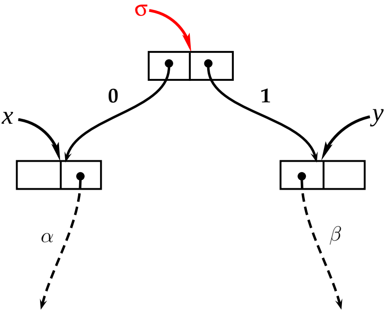
$$\mathcal{L}app((\textbf{car}\ x), \sigma) = \{x.\epsilon,\ x.\mathbf{0}\sigma\}$$

LIVENESS ANALYSIS OF APPLICATIONS



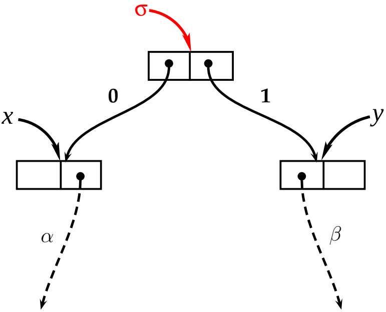
$$\mathcal{L}app((\textbf{cons}\ x\ y), \sigma) = \{x.\alpha \mid \mathbf{0}\alpha \in \sigma\} \cup \{y.\beta \mid \mathbf{1}\beta \in \sigma\}$$

LIVENESS ANALYSIS OF APPLICATIONS



$$\begin{aligned}\mathcal{L}app((\textbf{cons}\ x\ y), \sigma) \quad = \quad & \{x.\alpha \mid \mathbf{0}\alpha \in \sigma\} \ \cup \\ & \{y.\beta \mid \mathbf{1}\beta \in \sigma\}\end{aligned}$$

Liveness analysis of Applications

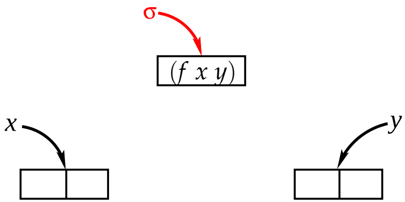


$$\mathcal{L}app((\textbf{cons}\ x\ y), \sigma) = \{x.\alpha \mid \mathbf{0}\alpha \in \sigma\} \cup \{y.\beta \mid \mathbf{1}\beta \in \sigma\}$$

- $\bar{\mathbf{0}}$ – Removal of a leading $\mathbf{0}$
  $\bar{\mathbf{1}}$ – Removal of a leading $\mathbf{1}$

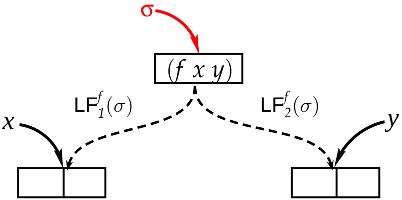
$$\mathcal{L}app((\textbf{cons}\ x\ y), \sigma) = x.\bar{\mathbf{0}}\sigma \cup y.\bar{\mathbf{1}}\sigma$$

Liveness Analysis of applications



$\mathcal{L}app((f\ x\ y), \sigma) =$

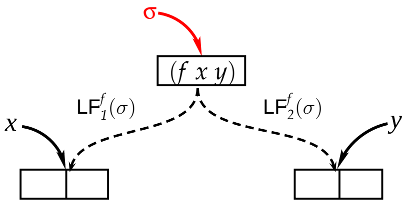
LIVENESS ANALYSIS OF APPLICATIONS



$$\mathcal{L}app((f\ x\ y), \sigma) = \{x.\mathsf{LF}^1_f(\sigma),\ y.\mathsf{LF}^2_f(\sigma)\}$$

- LF_f is a context independent summary of the function f.

Liveness Analysis of applications



$$\mathcal{L}app((f\ x\ y), \sigma) = \{x.\mathsf{LF}^1_f(\sigma),\ y.\mathsf{LF}^2_f(\sigma)\}$$

- LF_f is a context independent summary of the function f.
- To find $\mathsf{LF}^i_f(\sigma)$:
  - Assume a symbolic demand σ.
  - Let e_f be the body of f..
  - Set $\mathsf{LF}^i_f(\sigma)$ to $\mathcal{L}exp(e_f, \sigma)(x_i)$.

LIVENESS ANALYSIS OF EXPRESSIONS

$$\mathcal{L}exp((return\ x), \sigma) = \{x.\sigma\}$$

$$\mathcal{L}exp((\mathbf{if}\ x\ \ e_1\ \ e_2), \sigma) = \{x.\epsilon\} \cup \mathcal{L}exp(e_1, \sigma) \cup \mathcal{L}exp(e_2, \sigma)\}$$

$$\mathcal{L}exp((\mathbf{let}\ x\ \leftarrow\ s\ \mathbf{in}\ e), \sigma) = \mathsf{L} \setminus \{x.*\} \cup \mathcal{L}app(s, \mathsf{L}(x))$$

$$\text{where } \mathsf{L} = \mathcal{L}exp(e, \sigma)$$

LIVENESS ANALYSIS – THE BIG PICTURE

```
πmain: (let z ← ... in
        (let y ← ... in
     π9: (let w ← (append y z) in
      π10: (let a ← (cdr w) in
       π11: (let b ← (car a) in
        π12: (return b)))))))
```

```
(define (append l1 l2)
 π1: (let test ← (null? l1) in
  π2: (if test  π3:(return l2)
   π4: (let tl ← (cdr l1) in
    π5: (let rec ← (append tl l2) in
     π6: (let hd ← (car l1) in
      π7: (let ans ← (cons hd rec) in
       π8: (return ans))))))))
```

Liveness environments:

$$\mathsf{L}_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

$$\mathsf{L}_1^{\mathtt{l2}} = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

...

$$\mathsf{L}_9^{\mathtt{y}} = \mathsf{LF}^1_{\mathbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$$

Demand summaries:

Function summaries:

$$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\mathbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

LIVENESS ANALYSIS – DEMAND SUMMARY

$\sigma_{\mathbf{main}} = \sigma_{all}$

```
πmain: (let z ← ... in
        (let y ← ... in
   π9: (let w ← (append y z) in
   π10: (let a ← (cdr w) in
   π11: (let b ← (car a) in
   π12: (return b)))))))
```

```
(define (append l1 l2)
  π1: (let test ← (null? l1) in
   π2: (if test  π3:(return l2)
    π4: (let tl ← (cdr l1) in
     π5: (let rec ← (append tl l2) in
      π6: (let hd ← (car l1) in
       π7: (let ans ← (cons hd rec) in
        π8: (return ans))))))))
```

Liveness environments: **Demand summaries:** **Function summaries:**

$$\mathsf{L}_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

$$\mathsf{L}_1^{\mathtt{l2}} = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

...

$$\mathsf{L}_9^{\mathtt{y}} = \mathsf{LF}^1_{\mathbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$$

$$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\mathbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

LIVENESS ANALYSIS – DEMAND SUMMARY

$\sigma_{\mathbf{main}} = \sigma_{all}$

```
πmain: (let z ← ... in
        (let y ← ... in           σ1
  π9:  (let w ← (append y z) in
   π10: (let a ← (cdr w) in
    π11: (let b ← (car a) in
     π12: (return b))))))
```

```
(define (append l1 l2)
 π1: (let test ← (null? l1) in
  π2: (if test  π3:(return l2)
   π4: (let tl ← (cdr l1) in
    π5: (let rec ← (append tl l2) in
     π6: (let hd ← (car l1) in
      π7: (let ans ← (cons hd rec) in
       π8: (return ans)))))))
```

Liveness environments:

$\mathsf{L}_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup$

$\quad \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$

$\mathsf{L}_1^{\mathtt{l2}} = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$

...

$\mathsf{L}_9^{\mathtt{y}} = \mathsf{LF}^1_{\mathbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$

Demand summaries:

Function summaries:

$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup$

$\quad \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$

$\mathsf{LF}^2_{\mathbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$

LIVENESS ANALYSIS – DEMAND SUMMARY

$\sigma_{\textbf{main}} = \sigma_{all}$ $\qquad$ $\sigma_{\textbf{append}} = \sigma_1 \cup \ldots$

```
πmain: (let z ← ... in
        (let y ← ... in                σ1
  π9: (let w ← (append y z) in
   π10: (let a ← (cdr w) in
    π11: (let b ← (car a) in
     π12: (return b))))))
```

```
(define (append l1 l2)
 π1: (let test ← (null? l1) in
  π2: (if test  π3:(return l2)
   π4: (let tl ← (cdr l1) in
    π5: (let rec ← (append tl l2) in
     π6: (let hd ← (car l1) in
      π7: (let ans ← (cons hd rec) in
       π8: (return ans))))))))
```

Liveness environments: $\qquad$ **Demand summaries:** $\qquad$ **Function summaries:**

$$\mathsf{L}_1^{\texttt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

$$\mathsf{L}_1^{\texttt{l2}} = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

...

$$\mathsf{L}_9^{\texttt{y}} = \mathsf{LF}^1_{\textbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{1}\mathbf{0}\sigma_{all})$$

$$\mathsf{LF}^1_{\textbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\textbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

LIVENESS ANALYSIS – DEMAND SUMMARY

$\sigma_{\textbf{main}} = \sigma_{all}$

```
πmain: (let z ← ... in
          (let y ← ... in            σ1
    π9:    (let w ← (append y z) in
    π10:    (let a ← (cdr w) in
    π11:     (let b ← (car a) in
    π12:      (return b))))))
```

$\sigma_{\textbf{append}} = \sigma_1 \cup \ldots$

```
(define (append l1 l2)
  π1: (let test ← (null? l1) in
   π2: (if test  π3: (return l2)
    π4: (let tl ← (cdr l1) in        σ2
     π5: (let rec ← (append tl l2) in
      π6: (let hd ← (car l1) in
       π7: (let ans ← (cons hd rec) in
        π8: (return ans)))))))
```

Liveness environments:

$$\mathsf{L}_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

$$\mathsf{L}_1^{\mathtt{l2}} = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

...

$$\mathsf{L}_9^{\mathtt{y}} = \mathsf{LF}^1_{\textbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$$

Demand summaries:

Function summaries:

$$\mathsf{LF}^1_{\textbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\textbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

LIVENESS ANALYSIS – DEMAND SUMMARY

$\sigma_{\textbf{main}} = \sigma_{all}$

π_{main}: (let z ← ... in
(let y ← ... in σ_1
π_9: (let w ← **(append y z)** in
π_{10}: (let a ← (cdr w) in
π_{11}: (let b ← (car a) in
π_{12}: (return b))))))

$\sigma_{\textbf{append}} = \sigma_1 \cup \sigma_2$

(define (append l1 l2)
π_1: (let test ← (null? l1) in
π_2: (if test π_3: (return l2)
π_4: (let tl ← (cdr l1) in σ_2
π_5: (let rec ← **(append tl l2)** in
π_6: (let hd ← (car l1) in
π_7: (let ans ← (cons hd rec) in
π_8: (return ans))))))))

Liveness environments:

$$\mathsf{L}_1^{\texttt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

$$\mathsf{L}_1^{\texttt{l2}} = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma_{\textbf{append}})$$

...

$$\mathsf{L}_9^{\texttt{y}} = \mathsf{LF}^1_{\textbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$$

Demand summaries:

Function summaries:

$$\mathsf{LF}^1_{\textbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\textbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\textbf{append}}(\bar{\mathbf{1}}\sigma)$$

LIVENESS ANALYSIS – DEMAND SUMMARY

```
π_main: (let z ← ... in
          (let y ← ... in
      π9: (let w ← (append y z) in
       π10: (let a ← (cdr w) in
        π11: (let b ← (car a) in
         π12: (return b)))))))
```

```
(define (append l1 l2)
 π1: (let test ← (null? l1) in
  π2: (if test  π3:(return l2)
   π4: (let tl ← (cdr l1) in
    π5: (let rec ← (append tl l2) in
     π6: (let hd ← (car l1) in
      π7: (let ans ← (cons hd rec) in
       π8: (return ans))))))))
```

Liveness environments:

$$\mathsf{L}_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

$$\mathsf{L}_1^{\mathtt{l2}} = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

...

$$\mathsf{L}_9^{\mathtt{y}} = \mathsf{LF}^1_{\mathbf{append}}(\{\epsilon, \mathbf{1}\} \cup \mathbf{10}\sigma_{all})$$

Demand summaries:

$$\sigma_{\mathbf{main}} = \sigma_{all}$$

$$\sigma_{\mathbf{append}} = \{\epsilon,\ \mathbf{1}\} \cup \mathbf{10}\sigma_{all} \cup \bar{\mathbf{1}}\sigma_{\mathbf{append}}$$

Function summaries:

$$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

$$\mathsf{LF}^2_{\mathbf{append}}(\sigma) = \sigma \cup \mathsf{LF}^2_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

OBTAINING A CLOSED FORM SOLUTION FOR LF

- Function summaries will always have the form:

$$\mathsf{LF}_f^i(\sigma) = \mathsf{I}_f^i \cup \mathsf{D}_f^i \sigma$$

OBTAINING A CLOSED FORM SOLUTION FOR LF

- Function summaries will always have the form:

$$\mathsf{LF}_f^i(\sigma) = \mathsf{I}_f^i \cup \mathsf{D}_f^i \sigma$$

- Consider the equation for $\mathsf{LF}^1_{\mathbf{append}}$

$$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \ \cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

OBTAINING A CLOSED FORM SOLUTION FOR LF

- Function summaries will always have the form:

$$\mathsf{LF}_f^i(\sigma) = \mathsf{I}_f^i \cup \mathsf{D}_f^i\sigma$$

- Consider the equation for $\mathsf{LF}^1_{\mathbf{append}}$

$$\mathsf{LF}^1_{\mathbf{append}}(\sigma) = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \;\cup \mathbf{1}\mathsf{LF}^1_{\mathbf{append}}(\bar{\mathbf{1}}\sigma)$$

- Substitute the assumed form in the equation:

$$\mathsf{I}^1_{\mathbf{append}} \cup \mathsf{D}^1_{\mathbf{append}}\sigma = \{\epsilon\} \;\cup\; \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}(\mathsf{I}^1_{\mathbf{append}} \cup \mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}\sigma)$$

- Equating the terms without and with σ, we get:

$$\begin{aligned}\mathsf{I}^1_{\mathbf{append}} &= \{\epsilon\} \;\cup\; \mathbf{0}\bar{\mathbf{0}} \cup \mathbf{1}\mathsf{I}^1_{\mathbf{append}}\\ \mathsf{D}^1_{\mathbf{append}} &= \mathbf{0}\bar{\mathbf{0}} \cup \mathbf{1}\mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}\end{aligned}$$

SUMMARY OF ANALYSIS RESULTS

Liveness at program points:

$$L_1^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma \cup \mathbf{1}(\mathsf{l}^1_{\mathbf{append}} \cup \mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}\sigma_{\mathbf{append}})$$

$$L_1^{\mathtt{l2}} = \{\epsilon\} \cup \mathsf{l}^2_{\mathbf{append}} \cup \mathsf{D}^2_{\mathbf{append}}\bar{\mathbf{1}}\sigma_{\mathbf{append}}$$

$$L_5^{\mathtt{l1}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}}\sigma_{\mathbf{append}}$$

$$L_5^{\mathtt{tl}} = \mathsf{l}^1_{\mathbf{append}} \cup \mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}\sigma_{\mathbf{append}}$$

$$L_5^{\mathtt{l2}} = \mathsf{l}^2_{\mathbf{append}} \cup \mathsf{D}^2_{\mathbf{append}}\bar{\mathbf{1}}\sigma_{\mathbf{append}}$$

$\ldots$

Demand summaries:

$$\sigma_{\mathbf{append}} = \{\epsilon,\ \mathbf{1}\} \cup \bar{\mathbf{1}}\sigma_{\mathbf{append}} \cup \mathbf{10}\sigma_{all}$$

Function summaries:

$$\mathsf{l}^1_{\mathbf{append}} = \{\epsilon\} \cup \mathbf{0}\bar{\mathbf{0}} \cup \mathbf{1}\mathsf{l}^1_{\mathbf{append}}$$

$$\mathsf{D}^1_{\mathbf{append}} = \mathbf{0}\bar{\mathbf{0}} \cup \mathbf{1}\mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}$$

$$\mathsf{l}^2_{\mathbf{append}} = \mathsf{l}^2_{\mathbf{append}}$$

$$\mathsf{D}^2_{\mathbf{append}} = \{\epsilon\} \cup \mathsf{D}^2_{\mathbf{append}}\bar{\mathbf{0}}$$

SOLUTION OF THE EQUATIONS

View the equations as grammar rules:

$$
\begin{aligned}
\mathsf{L}_1^{\mathtt{l1}} &\rightarrow \epsilon \mid \mathbf{0}\bar{\mathbf{0}}\sigma \mid \mathbf{1}(\mathsf{I}^1_{\mathbf{append}} \mid \mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}\sigma_{\mathbf{append}}) \\
\mathsf{I}^1_{\mathbf{append}} &\rightarrow \epsilon \mid \mathbf{1}\mathsf{I}^1_{\mathbf{append}} \\
\mathsf{D}^1_{\mathbf{append}} &\rightarrow \mathbf{0}\bar{\mathbf{0}} \mid \mathbf{1}\mathsf{D}^1_{\mathbf{append}}\bar{\mathbf{1}}
\end{aligned}
$$

The solution of $\mathsf{L}_1^{\mathtt{l1}}$ is the language $\mathscr{L}(\mathsf{L}_1^{\mathtt{l1}})$ generated by it.

$\mathbf{0/1}$ REMOVAL

- The CFG has access paths marked for **0/1** removal arising from the **cons** rule.

$\mathbf{0}/\mathbf{1}$ REMOVAL

- The CFG has access paths marked for **0/1** removal arising from the **cons** rule.
- **0** removal from a set of access paths:

$$\alpha_1\bar{\mathbf{0}}\mathbf{0}\alpha_2 \Rightarrow \alpha_1\alpha_2$$
$$\alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \Rightarrow \text{ drop } \alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \text{ from the set}$$

$\mathbf{0/1}$ REMOVAL

- The CFG has access paths marked for $\mathbf{0/1}$ removal arising from the **cons** rule.
- $\mathbf{0}$ removal from a set of access paths:

$$\alpha_1\bar{\mathbf{0}}\mathbf{0}\alpha_2 \Rightarrow \alpha_1\alpha_2$$
$$\alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \Rightarrow \text{ drop } \alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \text{ from the set}$$

- The simplification is easier to do on a finite state automaton.

$\mathbf{0}/\mathbf{1}$ REMOVAL

- The CFG has access paths marked for **0/1** removal arising from the **cons** rule.
- **0** removal from a set of access paths:

$$\alpha_1\bar{\mathbf{0}}\mathbf{0}\alpha_2 \Rightarrow \alpha_1\alpha_2$$
$$\alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \Rightarrow \text{ drop } \alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \text{ from the set}$$

- The simplification is easier to do on a finite state automaton.
- Approximate the CFG by a strongly regular language (Mohri-Nederhoff transformation).

$\mathbf{0}/\mathbf{1}$ REMOVAL

- The CFG has access paths marked for **0/1** removal arising from the **cons** rule.
- **0** removal from a set of access paths:

$$\alpha_1\bar{\mathbf{0}}\mathbf{0}\alpha_2 \Rightarrow \alpha_1\alpha_2$$
$$\alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \Rightarrow \text{ drop } \alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \text{ from the set}$$

- The simplification is easier to do on a finite state automaton.
- Approximate the CFG by a strongly regular language (Mohri-Nederhoff transformation).
- Convert the strongly regular language to an automaton.

$\mathbf{0}/\mathbf{1}$ REMOVAL

- The CFG has access paths marked for **0/1** removal arising from the **cons** rule.
- **0** removal from a set of access paths:

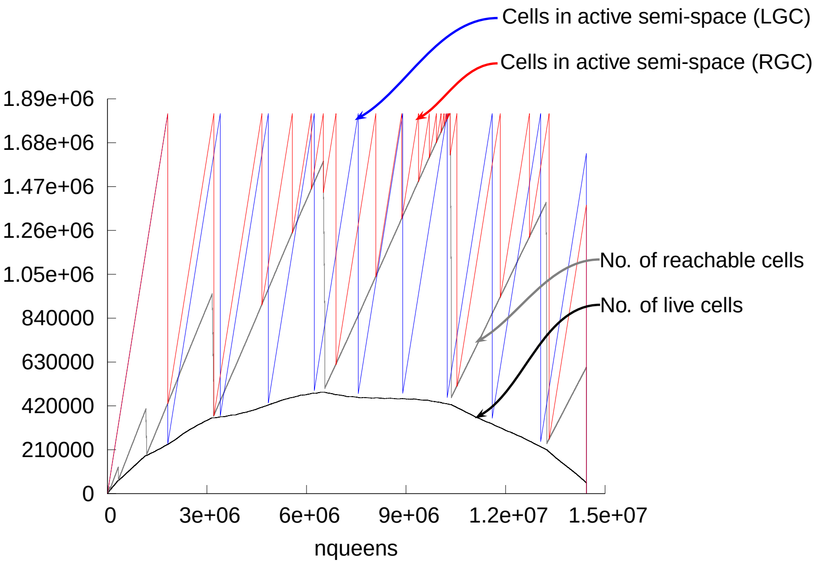
$$\alpha_1\bar{\mathbf{0}}\mathbf{0}\alpha_2 \Rightarrow \alpha_1\alpha_2$$
$$\alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \Rightarrow \text{ drop } \alpha_1\bar{\mathbf{0}}\mathbf{1}\alpha_2 \text{ from the set}$$

- The simplification is easier to do on a finite state automaton.
- Approximate the CFG by a strongly regular language (Mohri-Nederhoff transformation).
- Convert the strongly regular language to an automaton.
- Perform removal on the automaton.

EXPERIMENTAL SETUP

- Built a prototype consisting of:
  - An ANF-scheme interpreter
  - Liveness analyzer
  - A single-generation copying collector.
- The collector optionally uses liveness
  - Marks a link during GC only if it is live.
- Benchmark programs are mostly from the no-fib suite.

GC BEHAVIOR AS A GRAPH



RESULTS AS TABLES

Analysis Performance:

Program	sudoku	lcss	gc_bench	knightstour	treejoin	nqueens	lambda
Time (msec)	120.95	2.19	0.32	3.05	2.61	0.71	20.51
DFA size	4251	726	258	922	737	241	732
Precision(%)	**87.5**	**98.8**	**99.9**	**94.3**	**99.6**	**98.8**	**83.8**

- Our liveness analysis is precise.

RESULTS AS TABLES

Garbage collection performance

Program	# Collected cells per GC		#GCs		MinHeap (#cells)		GC time (sec)	
	RGC	LGC	RGC	LGC	RGC	LGC	RGC	LGC
sudoku	**490**	**1306**	22	9	1704	589	.028	.122
lcss	**46522**	**51101**	8	7	52301	1701	.045	.144
gc_bench	**129179**	**131067**	9	9	131071	6	.086	.075
nperm	**47586**	**174478**	14	4	202597	37507	1.406	.9
fibheap	**249502**	**251525**	1	1	254520	13558	.006	.014
knightstour	**2593**	**314564**	1161	10	508225	307092	464.902	14.124
treejoin	**288666**	**519943**	2	1	525488	7150	.356	.217
nqueens	**283822**	**1423226**	46	9	1819579	501093	70.314	24.811
lambda	**205**	**556**	23	8	966	721	.093	2.49

- LGC collects more garbage than RGC.

RESULTS AS TABLES

Garbage collection performance

Program	# Collected cells per GC		#GCs		MinHeap (#cells)		GC time (sec)	
	RGC	LGC	RGC	LGC	RGC	LGC	RGC	LGC
sudoku	490	1306	**22**	**9**	1704	589	.028	.122
lcss	46522	51101	**8**	**7**	52301	1701	.045	.144
gc_bench	129179	131067	**9**	**9**	131071	6	.086	.075
nperm	47586	174478	**14**	**4**	202597	37507	1.406	.9
fibheap	249502	251525	**1**	**1**	254520	13558	.006	.014
knightstour	2593	314564	**1161**	**10**	508225	307092	464.902	14.124
treejoin	288666	519943	**2**	**1**	525488	7150	.356	.217
nqueens	283822	1423226	**46**	**9**	1819579	501093	70.314	24.811
lambda	205	556	**23**	**8**	966	721	.093	2.49

- No of collections of LGC no higher than RGC. Very often, smaller.

RESULTS AS TABLES

Garbage collection performance

Program	# Collected cells per GC		#GCs		MinHeap (#cells)		GC time (sec)	
	RGC	LGC	RGC	LGC	RGC	LGC	RGC	LGC
sudoku	490	1306	22	9	**1704**	**589**	.028	.122
lcss	46522	51101	8	7	**52301**	**1701**	.045	.144
gc_bench	129179	131067	9	9	**131071**	**6**	.086	.075
nperm	47586	174478	14	4	**202597**	**37507**	1.406	.9
fibheap	249502	251525	1	1	**254520**	**13558**	.006	.014
knightstour	2593	314564	1161	10	**508225**	**307092**	464.902	14.124
treejoin	288666	519943	2	1	**525488**	**7150**	.356	.217
nqueens	283822	1423226	46	9	**1819579**	**501093**	70.314	24.811
lambda	205	556	23	8	**966**	**721**	.093	2.49

- Programs require smaller heaps to execute when run with LGC.

Results as Tables

Garbage collection performance

Program	# Collected cells per GC RGC	# Collected cells per GC LGC	#GCs RGC	#GCs LGC	MinHeap (#cells) RGC	MinHeap (#cells) LGC	GC time (sec) RGC	GC time (sec) LGC
sudoku	490	1306	22	9	1704	589	.028	.122
lcss	46522	51101	8	7	52301	1701	.045	.144
gc_bench	129179	131067	9	9	131071	6	**.086**	**.075**
nperm	47586	174478	14	4	202597	37507	**1.406**	**.9**
fibheap	249502	251525	1	1	254520	13558	.006	.014
knightstour	2593	314564	1161	10	508225	307092	**464.902**	**14.124**
treejoin	288666	519943	2	1	525488	7150	**.356**	**.217**
nqueens	283822	1423226	46	9	1819579	501093	**70.314**	**24.811**
lambda	205	556	23	8	966	721	.093	2.49

- GC time is smaller for LGC in some cases...

RESULTS AS TABLES

Garbage collection performance

Program	# Collected cells per GC		#GCs		MinHeap (#cells)		GC time (sec)	
	RGC	LGC	RGC	LGC	RGC	LGC	RGC	LGC
sudoku	490	1306	22	9	1704	589	**.028**	**.122**
lcss	46522	51101	8	7	52301	1701	**.045**	**.144**
gc_bench	129179	131067	9	9	131071	6	.086	.075
nperm	47586	174478	14	4	202597	37507	1.406	.9
fibheap	249502	251525	1	1	254520	13558	**.006**	**.014**
knightstour	2593	314564	1161	10	508225	307092	464.902	14.124
treejoin	288666	519943	2	1	525488	7150	.356	.217
nqueens	283822	1423226	46	9	1819579	501093	70.314	24.811
lambda	205	556	23	8	966	721	**.093**	**2.49**

- ...and larger in some.

SCOPE FOR FUTURE WORK

- Reducing GC-time.
  - Reducing re-visits to heap nodes.
  - Basing the implementation on full Scheme, not ANF-Scheme
- Increasing the scope of the method.
  - Higher order functions.
  - Lazy languages.
- Orthogonal: Using the notion of demand for other analysis.

CONCLUSIONS

- Proposed a liveness-based GC scheme. Proved theoretically:
    - The soundness of liveness analysis.
    - That it can never do more collections than reachability based GC.
- Implemented a prototype that:
    - Demonstrated the precision of the analysis.
    - Demonstrated reduced heap requirement.
    - Reduced GC time for a majority of programs.
- Unfinished agenda:
    - Improving GC time for a larger fraction of programs.
    - Improving scope of the method.