List Processing in Real Time on a Serial Computer

Henry G. Baker, Jr.

Presented by Mike Bond

Talk outline

- Problem and solution
- MFYCA Algorithm
- Baker's Algorithm (SRT)
- Discussion

Problem

- Three problems with list processing systems:
  1. Interpretation kills performance
  2. Inefficient use of space
  3. Garbage collection halts execution for time proportional to size of live data
- Compilation fixes first two
- Paper targets third problem

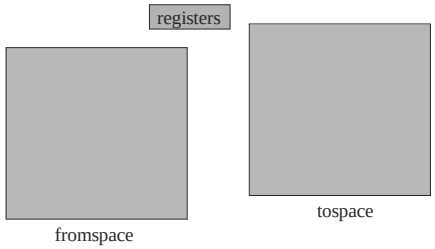
Solution

- Baker's algorithm (SRT)
- Based on MFYCA's algorithm
- Basic idea: Do a little copying during each *cons*, rather than a lot of copying infrequently
- Real-time: all operations in O(1) time
- Pretty good space efficiency

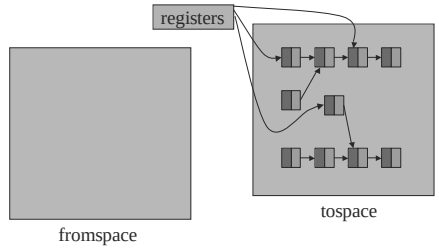
Talk outline

- Problem and solution
- MFYCA Algorithm
- Baker's Algorithm (SRT)
- Discussion

MFYCA algorithm: The setup

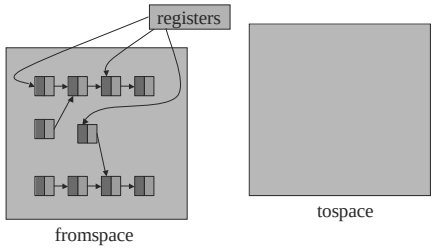


MFYCA algorithm: Between garbage collections



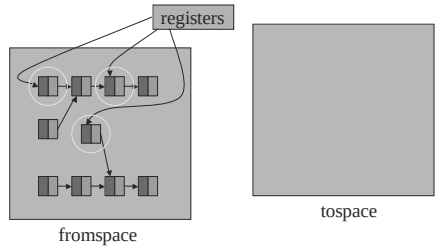
Between garbage collections, everything in *tospace*

MFYCA algorithm: Flip



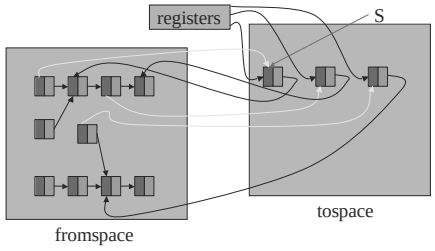
Flip is the first step of garbage collection

MFYCA algorithm: Copying



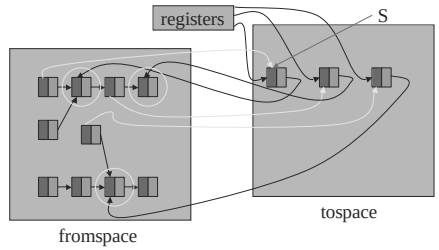
Nodes pointed to by registers are the first copied

MFYCA algorithm: Copying (continued)



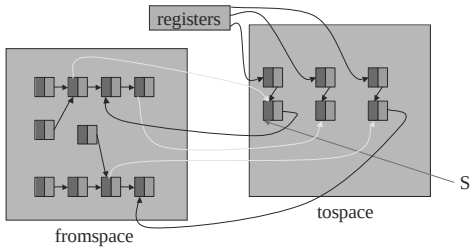
The yellow arrows represent forwarding addresses

MFYCA algorithm: Copying (continued)



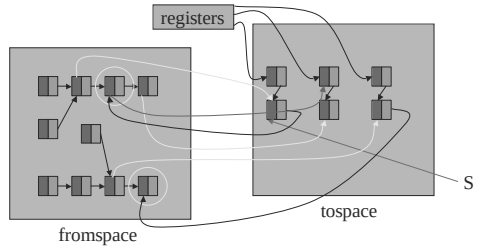
The circled nodes will be copied next

MFYCA algorithm: Copying (continued)



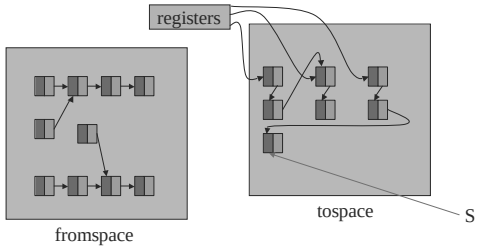
Some arrows between *fromspace* and *tospace* have been omitted

MFYCA algorithm: Copying (continued)



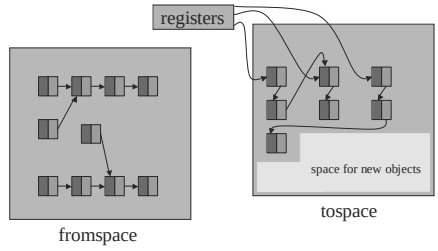
The circled nodes will be copied next

MFYCA algorithm: Copying (continued)



Arrows between *fromspace* and *tospace* have been omitted

MFYCA algorithm: Garbage collection finished



Everything in *fromspace* is garbage

Talk outline

- Problem and solution
- MFYCA Algorithm
- Baker's Algorithm (SRT)
- Discussion

Baker's algorithm: How it's different

- When *tospace* fills up, do a flip and copy only roots
- Every *cons* does a few iterations of garbage collection
- Each *car* and *cdr* checks for a forwarding address – updates pointer if found

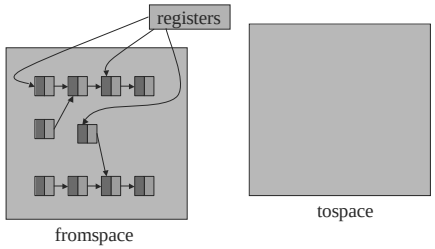
Baker's algorithm: Proven guarantees

- Serial Real-Time system
  - one thread
  - all operations O(1) time
- Won't run out of *tospace* (unless we really run out of space)
  - (1 + 1/k) times as much storage needed as for MFYCA
  - Tradeoff between time and space

Baker's algorithm: Unsupported cases

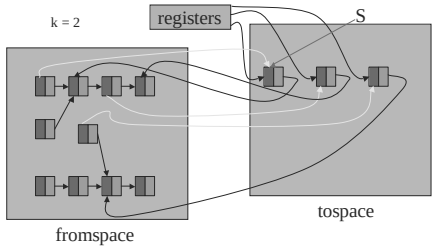
- Operations not constant time…
  - … on virtual memory machines (?)
  - … for arrays
  - … during memory extensions
- But no one uses those anymore, right?

Baker's algorithm: Right after flip



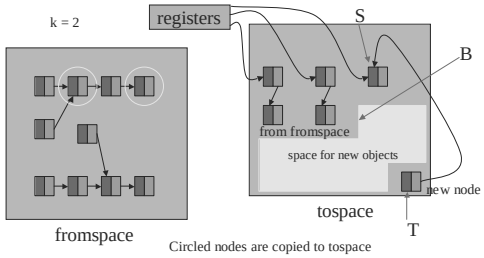
Flip is the first step of garbage collection

Baker's algorithm: Copy the roots



Only roots are copied after a flip

Baker's algorithm: Result of a *cons*



Circled nodes are copied to tospace

Arrows between fromspace and tospace are not shown

Talk outline

- Problem and solution
- MFYCA Algorithm
- Baker's Algorithm (SRT)
- Discussion

Discussion: Space

- Paper very concerned with space: with large *k*, Baker and MFYCA use essentially same space
- Space still so important in 2003? Or are concerns somewhat different?
- Small working set size more important today
- How do Baker's modifications affect working set size?

Discussion: Breadth-first order

- The “graph” of objects is traversed in breadth-first order
  - True for both MFYCA and Baker
- Why?
- Is this beneficial? Consider locality.

Discussion: T pointer

- Baker’s algorithm adds T pointer to MFYCA
- New objects allocated at the end of the free space; garbage-collected accessible objects copied to beginning of free space

- Why do this?
- Does this have other effects? Could MFYCA have used a T pointer?

Discussion: Read barrier overhead

- The (potential) copying done by each *car* and *cdr* has to be done by MFYCA, too
- However, each *car* and *cdr* checks if the node is in *fromspace* vs. *tospace*
- Is this bad?
- How bad?
- Consider database application (from paper)

Discussion: More operations

- How well does Baker’s algorithm apply to a larger set of operations than *cons*, *car*, *cdr*, *rplaca*, *rplacd*, *eq*, and *atom*?
- Consider imperative languages

Conclusion

- Baker’s algorithm: Modification to MFYCA
- Contributions:
  - Real-time: all operations constant time
  - Space efficiency and flexibility: can choose *k* for space-time tradeoff
  - Proof: Correct and doesn’t run out of space when it shouldn’t