Class 28: Recursion with Helper Procedures

Held: Wednesday, 14 October 2009

Summary: We consider a different form of recursion, one based on the construction of *recursive helpers* that take additional parameters. Along the way, we consider the idea of *tail recursion*. We also explore how careless design of recursive procedures can inadvertently lead to slow execution.

Related Pages:

- EBoard.
- Lab: Recursion with Helper Procedures.
- Reading: Recursion with Helper Procedures.
- Due: Assignment 6: Conditionals.

Notes:

- Yes, we will have a quiz on Friday. It will be on recursion (including, possibly, helper recursion).
- No, you don't get an assignment 7 to do over fall break.
- Reading for Friday: List Recursion, Revisited.

Overview:

- Delayed evaluation in recursive procedures.
- A strategy: Carry along intermediate results.
- Using recursive helpers.
- A term: Tail recursion.
- Designing recursive procedures.

Delayed Evalution in Recursive Procedures

- A number of you have noted that recursion, as written, builds up a bunch of stuff to evaluate.
- For example, if we're summing the list (2 3 5 7 11 13), we end up with
  (+ 2 (+ 3 (+ 5 (+ 7 (+ 11 (+ 13 (sum ()))))))
  before we start doing the addition.
- Similarly, in selecting only the names of dark colors from a list, we might end up with
  (cons "black" (cons "darkblue" (cons "darkgrey" (select-dark ()))))
- Once we get to the base case of the recursion, we can then start to build up the actual result.
- Some people find the delayed evaluation natural, others find it awkward.
- For the latter group, we look for a strategy that helps us evaluate partial results along the way.

Helper Recursion

- The model that we call *helper recursion* (and that many of our colleagues call tail recursion) adds an extra parameter to the recursive procedure
  - That extra parameter carries along partial/intermediate results
- For example, in summing the list (2 3 5 7 11 13), we might have

```
partial-sum           unexplored-elements
0                     (2 3 5 7 11 13)
2                     (3 5 7 11 13)
5                     (5 7 11 13)
8                     (7 11 13)
15                    (11 13)
26                    (13)
39                    ()
```

- When we run out of elements, we can use the intermediate result as our final result
- However, there's a small problem with this strategy: When a client makes the *first* call to the procedure, they won't necessarily understand the purpose of the extra parameter.
  - Hence, we make the modified procedure a helper to the top-level procedure.
  - The top-level procedure is responsible for filling in the extra parameter of the helper.

Tail Recursion

- Note that the two forms of recurison we've seen (direct recursion and helper recursion) have a somewhat different post-recursion step
  - In the first kinds of recursive procedures we wrote, there's still work to do after the recursive call finishes.
  - In the helper-recursion procedures, once we're done with the recursive call, the result is ready; it requires no further processing (at least not within the helper).
- It turns out that there are particularly efficient ways to implement recursive procedures that do not further process recursive results.
- Because of this efficiency, we have a special term for such procedures. We call them *tail-recursive procedures*.
- If any recursive call is a the last operation of a procedure (that is, the tail of the procedure), then we say that the procedure is tail recursive.
  - If some work may be required after one of the recursive calls, then we say that the procedure is not tail recursive.

Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.