O'Keefe Chapter 6: Sequences
Part 2

Mike Daniels

8 November 2001

One small wrinkle

- maximum(L) = the maximal element in L
- g(X) = X
- h(X) = X
- f(X, Y) = X if X $\geq$ Y or Y otherwise
- . . . um
- What if e doesn't exist?

Solution

- p([Head|Tail], Result) $\leftarrow$ g(Head, Val), p(Tail, Val, Answer), h(Answer, Result).
- p([], Answer, Answer).
- p([Heal|Tail], SoFar, Answer) $\leftarrow$ g(Head, Val), f(SoFar, Val, Next), p(Tail, Next, Answer).
- (What would list_sum look like under this system?)

Dangerous Temptations

- average(List, Average) $\leftarrow$ list_sum(List, Sum), length(List, Length), Average is Sum/Length.
- What's wrong with this?

The better way

- average([Head|Tail], Average) ← average(Tail, Head, N, 1, D), Average is N/D.
- average([], N, N, D, D).
- average([Head|Tail], N0, N, D0, D) ← N1 is N0 + Head, D1 is D0 + 1, average(Tail, N1, N, D1, D).

The Logarithmic Method

- Covered on page 208.
- Basic idea: To work on a list of length N, first do each half of the list, and then combine the results.
- More stack space needed (non-tail recursive)...
- ...but generally fewer combining steps needed.

Odds and Evens

- At each step, combine the odds and then the evens.
- Obviously, f must be commutative.
- See page 211-212 for code.
- Same number of combining steps, but no cuts needed.
- Needs more space.

Extracting subsequences

- Head
- Tail
- Element
- Prefix
- Suffix (of a finite sequence, naturally)
- Segment

- Note that really none of these notions are compatible with the sequences represented by Enumerators.

Head and Tail

- X_head(Sequence, Head) ← X_next(Head, Sequence, _).
- X_head(Sequence, Tail) ← X_next(_, Sequence, Tail).

Element

- See "converting Generators to Enumerators".

Prefix

- Reduces to append; see page 216–217.

Suffix

- Just the dual concept from prefix, but watch out for the left-to-right nature of lists.

Segment

- A contiguous subsequence: that is, the sequence formed by taking elements X through Y of a given sequence.
- See 220-222.