4.32

The streams in chapter 3 were built so that the last element was, a delayed item, which, when evaluated, produced a ~~sem~~ simple value and another delayed item of the same kind (Typically taking the preceding stream element as input.)

The "lazier" lazy lists described here have a thunk car and a thunk cdr. This allows for more structures than the streams, eg. lazy trees. In addition, the thunks are only evaluated when really needed, rather than in a "fixed order".

As a trivial example, this means the thunks can be used to signal or effect side-effects at the times of use rather than of list generation. Also... Lazy heaps?. Lazy double-linked lists?

"Seamlessly handle infinite data structures."

Since the lazy list elements are not special forms, being part of a general lazy evaluator, they can be used as first-class citizens, giving more flexibility. E.g. lazy integration rather than extra added delay beyond the stream itself.

How can the extra laziness be taken advantage of? Well, I don't rightly know, since I've done fairly little lazy programming. By delayed unification of list elements while allowing "later" elements to unify in the meantime?.