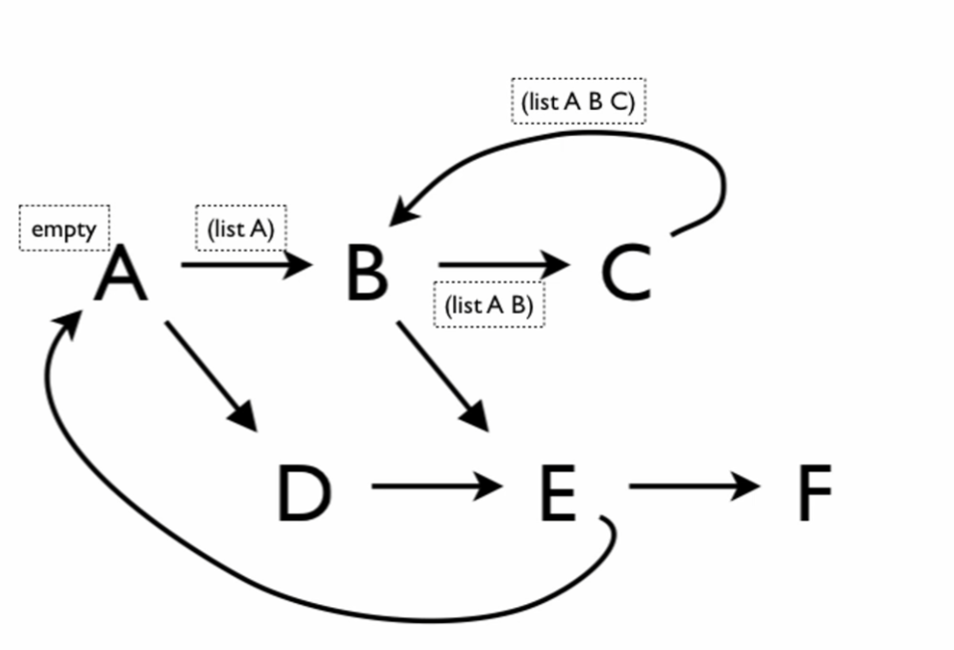
Planning the template





Context preserving accumulator

* So we’ll not go in circles
* This is just like the base case to stop a recursion

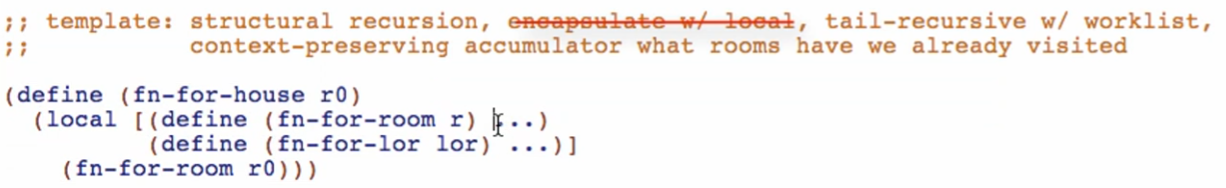


* When it is in the accumulator list, it has been there before therefore it will stop



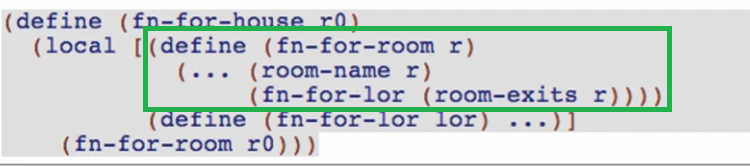
Working through the template

**Encapsulate with local**



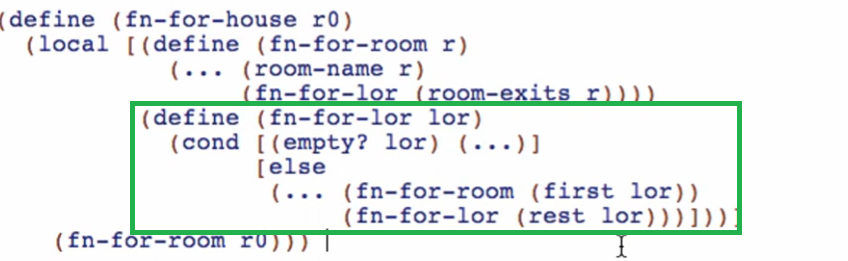
**Structural recursion**

fn-for-room



* Compound data
* Reference rule

fn-for-lor



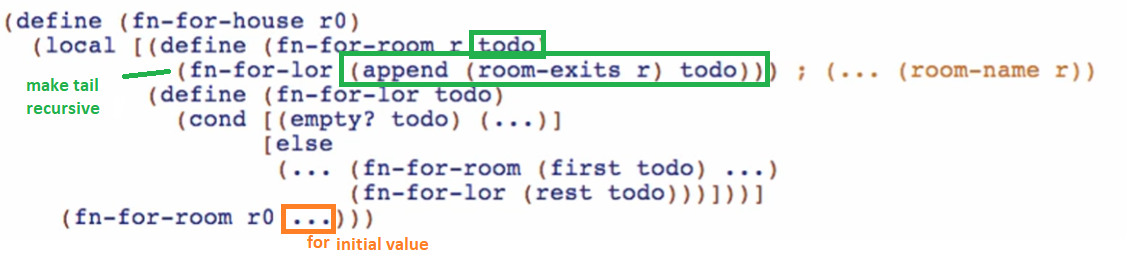
* Reference rule: fn-for-room
* Self-reference rule: fn-for-lor



Run and check if well-formed

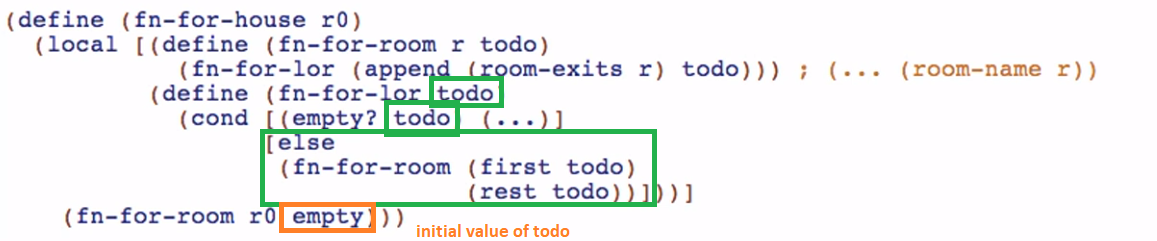
**Tail-recursive w/ worklist**

fn-for-room



* Add todo parameter
* Make tail recursive
* Append list of exits and todo

fn-for-lor



* Rename lor to todo
* Make tail recursive

Type and interp for todo accumulator





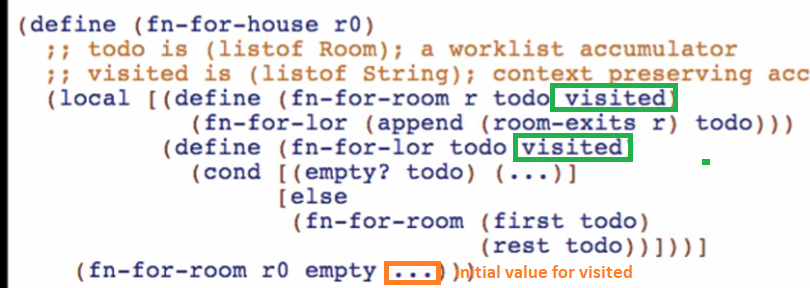
**Context-preserving accumulator**

Type and interp for visited accumulator

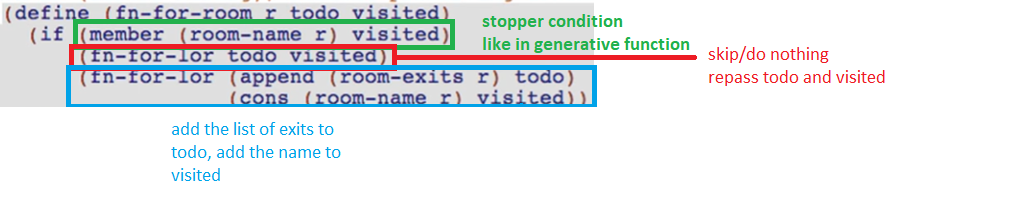




Add the accumulator

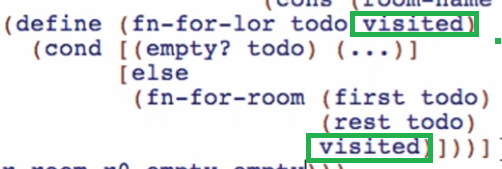


fn-for-room



* Check if the room is visited (for stopping the cycle)
  + If room-name is in visited, skip the room, do not add to todo (since it is now done) & visited (since it is already in the list
  + Else add the room-exits on the todo list, add the room-name on visited

fn-for-lor



* Pass visited to fn-for-room reference
  + No modification for this in natural recursions!

initial value



Overview:

