The one new design technique required for the Sudoku solver is template blending, in which we **combine several different kinds of templates** all of which contribute to the backbone of a function (or functions).

Backtracking Search Function

* Solve/say that it can’t be solved function

**Signature**



**Purpose**



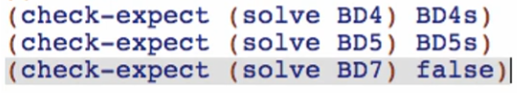
Assume that the board we start with has no invalid entries in it

**Stub**



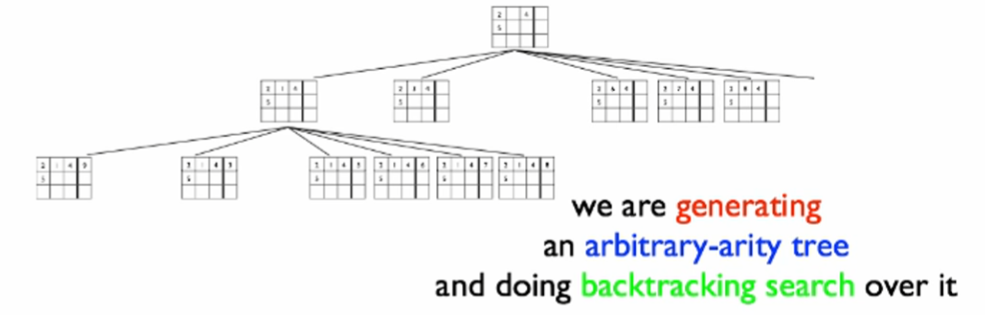
**Examples**

* We really don’t have a specific base case here so we use our examples on the type definition with solutions and one that is unsolvable



**Template**

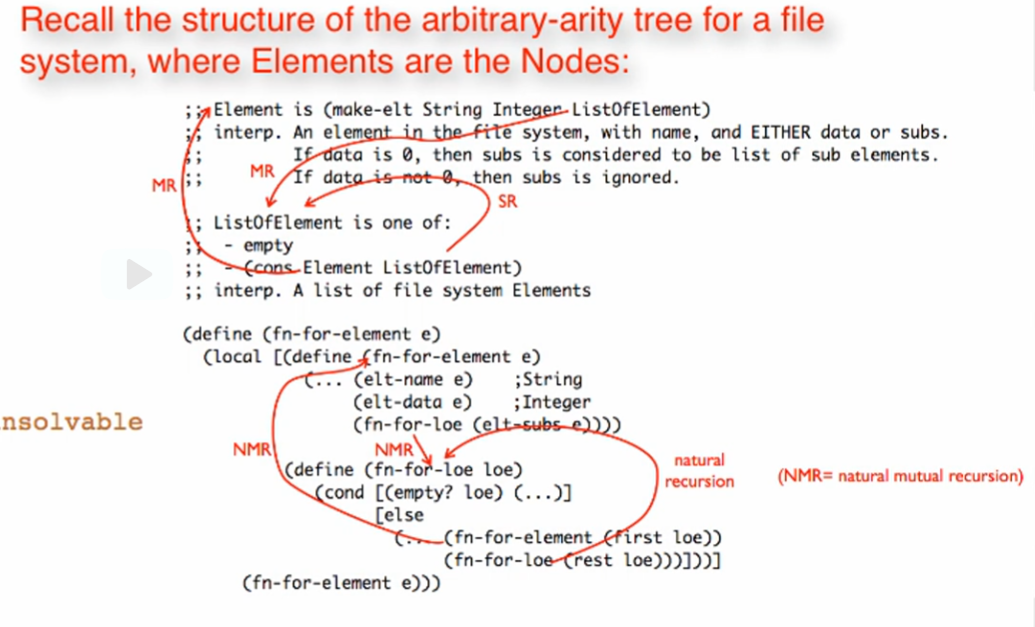
Again last time:



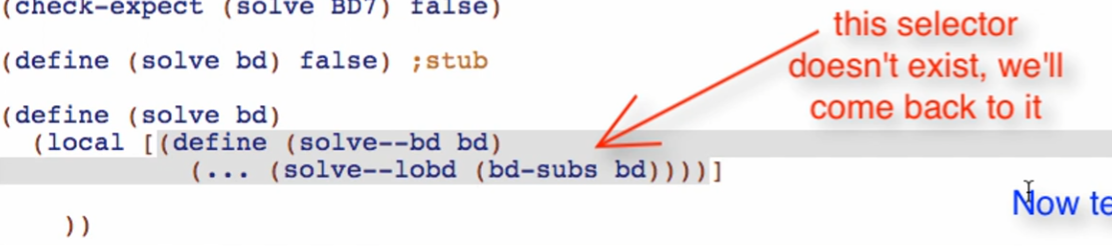
So we are really going to need elements of the template for:

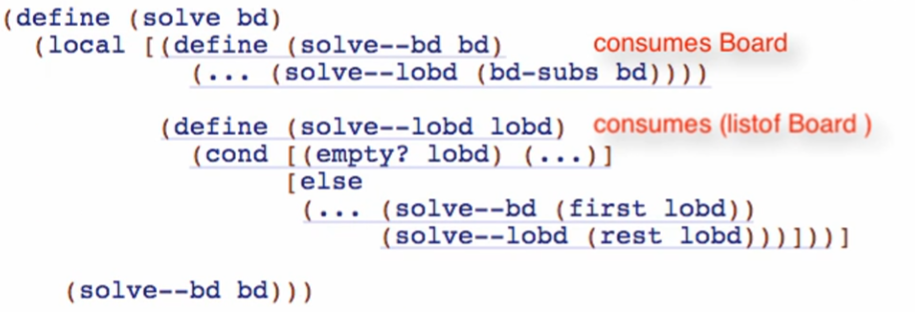
* Generative recursions
* Arbitrary-arity tree
* Backtracking search

Arbitrary-arity tree template

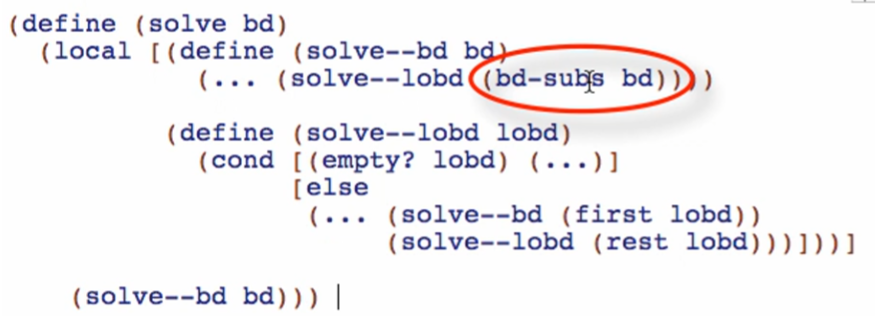


Templating accdg to **Arbitrary-arity tree**

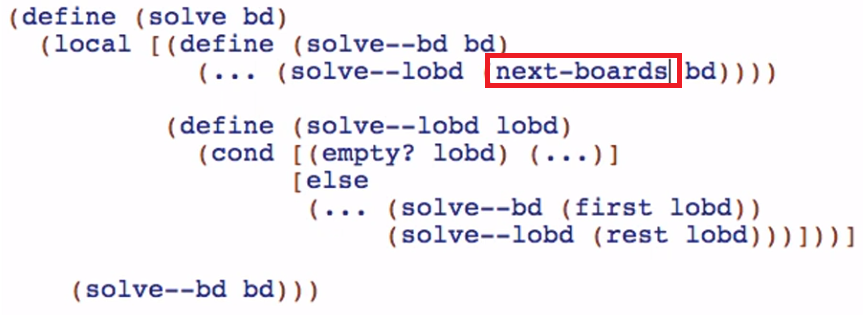




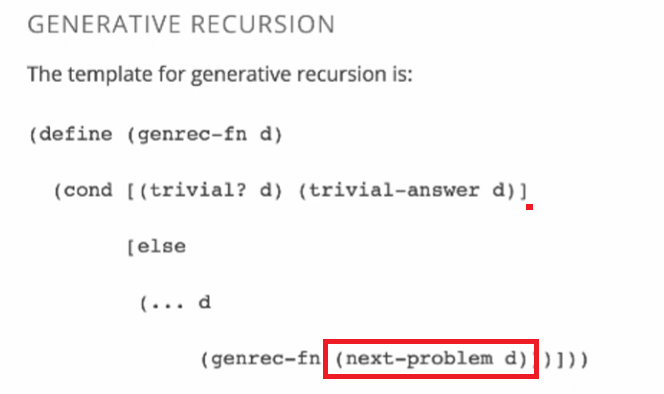
Templating accdg to **Generative Recursion**



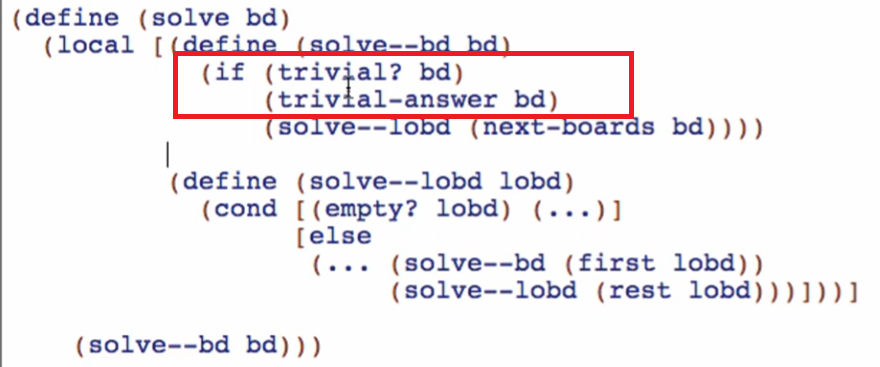
This will have to be GENERATED since we did not use structure definitions. Rename the function to next-boards so we will have distinction that this will be a generative function



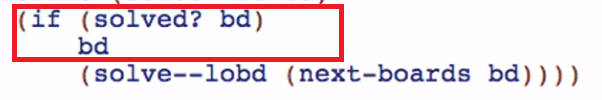
Going over the generative recursion recipe page, this is what we’ve done so far for our generative template



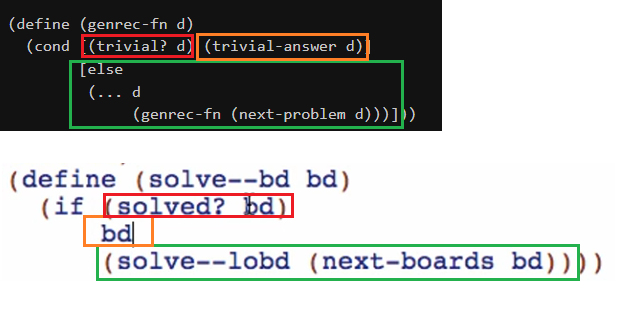
Now we need the trivial test



Then pick better name for the pred trivial? And replace the trivial-answer on what to produce if you satisfy the trivial? Predicate



Overview



Templating accdg to **Backtracking Search**

Go to design recipe of backtracking search

