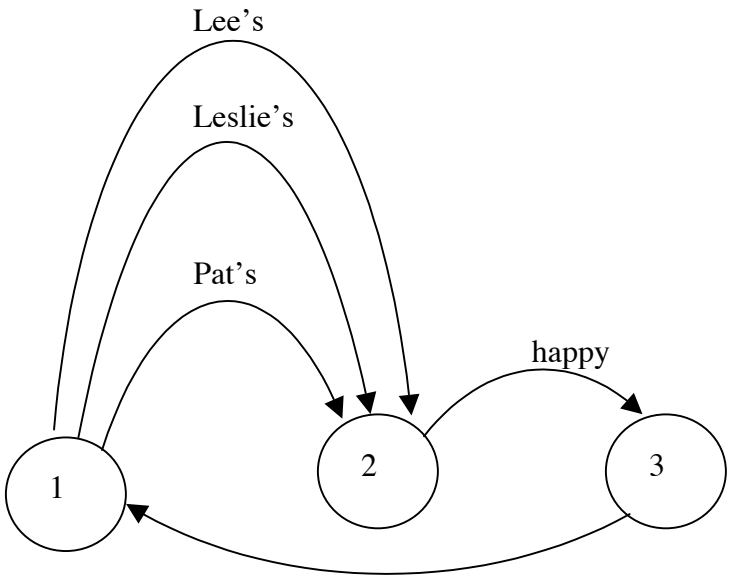
Finite-State Grammar, CFGs, Iteration and Recursion

Linguistics 7420
Fall 2004

The question. Can a finite-state grammar (FSG) represent structures that are modeled by recursive PS rules? The answer is yes and no. If the recursive node is at the left or right edge of a rule, it can be produced by iteration; if it's got terminal nodes on either side, the answer is no. The following is an illustration of a recursive string that can be modeled by iteration in a FSF, using a sentence that we would model with a right-branching recursive PS rule (AP → A S):

(1) Pat's happy Leslie's happy Lee's happy.

The model. To model this sentence, we can use the following **finite-state automaton**. This machine begins in an initial state, runs through a sequence of states (producing a word or string or words with each transition), and ends in a final state. This machine defines a language: the set of sentences that it can produce.



Here, we see that the effect of recursion can be modeled by iteration (returning to the initial state and multiple transition arcs to the 'source' state that produces the recurring adjective. The strings Pat's, Leslie's, etc. are not syntactic constituents, but any string of symbols can be produced by a transition.

Finite-state grammar. Each transition can be represented a rule: the symbol on the left-hand side of the rule is the FROM state, the symbol on the immediate right-hand side of the rule is the symbol that the transition outputs, and the symbol to the right of that is the TO state,

1 → Lee's 2
1 → Leslie's 2
1 → Pat's 2
2 → happy 3
3 → ø 1

The right-hand side of the rule can contain at most one (rightmost) non-final transition.

Some forms of recursion cannot be represented by a finite-state grammar. Chomsky (1957) showed that there are strings in English that are isomorphic to strings that a FSG cannot produce, in particular $a^n b^n$. These sentences involve dependencies between paired words, e.g., *either-or*, as in (2):

(2) Either S or S

Chomsky points out that either of the two S's in (2) can contain the string in (2), resulting in a structure like (3):

(3) Either [either S or S] or S

and so on:

(4) Either [either [either S or S] or S] or S

The problematic strings contain *n* instances of *either* followed by *n* instances of *or*. If we were to try to model such strings in a FSG, we would wind up needing one rule for every *n*. This is impossible if the recursion truly is infinite. A context-free grammar would need only two rules:

S → either S or S
S → S