**Formal Languages Cont’d**

* What features of C or Scheme programs cannot be verified by a DFA?
  + Consider ∑ = {(, )}
  + L = {w ∈ L\* | w is a string of balanced parens}
  + E.g. ε ∈ L, () ∈ L, (()()) ∈ L etc.
  + Each new state allows one more level of nesting, but no finite # of states allows all levels of nesting – cannot be modelled by DFA
* **Context-free languages**
  + Languages that can be described by a context-free grammar
  + E.g. balanced parens problem:
    - S → ε a word in the language is either empty,
    - S → (S) surrounded by parens, or
    - S → SS the concatenation of 2 words in the language
    - Shorthand: S → ε | (S) | SS
  + E.g. S ⇒ SS ⇒ (S)S ⇒ (S)(S) ⇒ ((S))(S) ⇒ (())(S) ⇒ (())()
    - ⇒ means “derives” – second string can be obtained from first string by one application of a grammar rule
  + A context-free grammar consists of:
    - An alphabet ∑ of terminal symbols
    - A finite, non-empty set N of non-terminal symbols
    - N ∩ ∑ = φ (they have no intersection)
    - V = N ∪ ∑ (“vocabulary” denotes union)
    - A finite set P of productions
      * Production has the form A → B, A ∈ N, B → V\*
    - An element S ∈ N – start symbol
  + Conventions:
    - a, b, c … − elements of ∑ (characters)
    - w, x, y … − elements of ∑\* (words)
    - A, B, C … S … − elements of N (non-terminals)
    - S – start symbol
    - α, β … − elements of V\*