Notas de aula - 20/02/2014

Análise Sintática

1. Análise sintática: agrupar os tokens na forma de uma árvore sintática
   (a) Exemplo: foo=42 → AL → [(foo,ID),(=),(42,NUM)] → AS → (atrib (id foo) (num 42))
   (b) Nem todas as sequências de tokens formam programas válidos
   (c) Representação para descrever estruturas válidas em uma linguagem de programação
      i. Característica inerentemente recursiva de linguagens de programação (exp é exp+exp, ou exp*exp...)
      ii. Gramáticas livre de contexto são uma notação natural

2. Gramáticas livre de contexto
   (a) $G = (V, T, P, S)$: V não-terminais, T terminais, produções $P : V \times (V \cup T)^*$, inicial $S \in V$
      i. A recursão está nas produções: um $A \in V$ que gera outros $A_i \in V$
      ii. É comum escrever gramáticas usando apenas as produções, os conjuntos podem ser deduzidos a partir de convenções de escrita
   (b) Relação de derivação ($\Rightarrow$):
      i. Deriva em um passo: $\alpha A \beta \Rightarrow \alpha x_1 ... x_n \beta$ sse $(A, x_1 ... x_n) \in P$
      ii. Fecho transitivo-recursivo ($\Rightarrow^*$): 0 ou mais passos
      iii. Fecho positivo ($\Rightarrow^+$): 1 ou mais passos.
   (c) A linguagem gerada por uma gramática G livre de contexto, $L(G) = \{w \in T^* | S \Rightarrow^+ w\}$, é uma linguagem livre de contexto
   (d) Linguagem definida pela derivação a partir de um símbolo:
      i. $L(a) = \{a\}, a \in T$
      ii. se $w \in L(A)$ existe $(A, x_1 ... x_n) \in P$ tal que $w = w_1 ... w_n$ e $w_i \in L(x_i)$

3. Quiz
   (a) Quais das cadeias a seguir estão na gramática dada?

$$
\begin{array}{lcl}
S & \rightarrow & aXa \\
X & \rightarrow & bY \\
X & \rightarrow & \epsilon \\
Y & \rightarrow & cXc \\
Y & \rightarrow & \epsilon
\end{array}
$$

      i. abcba
      ii. acca
      iii. aba *

iv. abcbcba

4. Árvore sintática concreta
   (a) Terminais nas folhas
   (b) Não-terminais nos nós interiores
   (c) A sequência de folhas ao percorrer a árvore in-order nos dá a cadeia derivada
   (d) A árvore dá a estrutura e a associatividade das operações que a cadeia original não mostra

5. Tipos de derivação
   (a) Qualquer sequência que nos leve de S a w é uma derivação, mas em geral estamos interessados em derivações sistemáticas
   (b) Derivação mais à esquerda: sequência de substituições em que sempre substituimos os não-terminal mais à esquerda
   (c) Derivação mais à direita: sequência de substituições em que sempre substituimos os não-terminal mais à direita
   (d) Diferentes estratégias de análise sintática levam a derivações mais à esquerda ou mais à direita

6. Ambiguidade
   (a) Podemos ter diferentes derivações para w, mas a árvore deve ser única
   (b) Se houver mais de uma árvore possível, o programa tem mais de uma interpretação possível!
   (c) Uma gramática ambígua:
   $$\begin{aligned} E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow (E) \\ E &\rightarrow num \end{aligned}$$
   Duas possibilidades para $num * num + num$ — qual delas queremos?
   (d) Resolvendo:
   $$\begin{aligned} E &\rightarrow A + E | A \\ A &\rightarrow B * A | B \\ B &\rightarrow num \\ B &\rightarrow (E) \end{aligned}$$
   (e) Na prática, um uso cuidadoso de ambiguidade pode manter a apresentação dela mais natural
   (f) Podemos eliminar a ambiguidade não na gramática, mas na implementação do analisador sintático
   (g) Ferramentas de geração de parsers possuem alguma ferramentas pra isso
      i. regras de eliminação de ambiguidade
      ii. diretivas de precedência e associatividade
   (h) Essas ferramentas permitem controlar como a eliminação é feita