Linguagens de Domínio Específico

Fabio Mascarenhas – 2017.1

http://www.dcc.ufrj.br/~fabiom/dsl

Recuperação de erros

- Podemos fazer recuperação de erros em parsers determinísticos reiniciando o parser após o erro, com marcas indicando as posições onde os erros aconteceram e qual o conjunto de terminais esperado (basta a cardinalidade)

- Quando um parser de token vai falhar, ele checa se a posição atual é uma posição de recuperação, se for executa a estratégia de recuperação adequada

- A estratégia de recuperação pode ser outro parser que vai simplesmente ficar onde está sem falhar ou pular avançar para o próximo token

- Estratégias de recuperação baseadas em sincronização são mais complicadas, vamos ver elas depois

Predicados sintáticos

- Um *predicado sintático* é um tipo de combinador que ou falha ou não consome nada da entrada
- O combinador `not` recebe um parser `p` e constrói um predicado que falha se `p` não falhar, e se `p` falhar simplesmente não consome nada
- O combinador `and` é o contrário de `not`, e falha caso `p` falhe e não consome nada caso `p` não falhe (qualquer coisa consumida por `p` é simplesmente ignorada)
- Predicados sintáticos permitem construir parsers que atuam diretamente nos caracteres da entrada, misturando análise léxica e sintática

Combinadores *scannerless*

- Uma vez que adicionamos predicados sintáticos, podemos fazer nossos combinadores atuarem diretamente em cima do texto, e não de uma sequência de tokens

- Precisamos apenas de combinadores especiais para fornecer informação suficiente sobre tokens da entrada para nossas mensagens de erro baseadas nas falhas mais distantes

- Com um pouco de abstração nossa *DSL* para escrever parsers não fica muito diferente da gramática léxica+sintática original