Compiladores II

Fabio Mascarenhas - 2014.2

http://www.dcc.ufrj.br/~fabiom/comp2

PEGs

- A gramática livre de contexto abaixo descreve a linguagem das PEGs:

```
peg  -> id '<-' exp peg | id '<-' exp
exp  -> term '/' exp | term
term -> pred term | pred
pred -> '!' pred | many
many -> simp '*' | simp
simp -> string | id | '(' exp ')'
```

PEG → parser

Ações

- Uma maneira de introduzir ações semânticas em nossa linguagem de PEGs é ter uma sintaxe para um operador similar ao combinador `bind`, que passaria o resultado de uma expressão para uma função, definida fora da gramática:

```
term -> bind term | bind
bind -> pred '->' id | pred
```

- A alta precedência é proposital, para ficar parecido com a precedência do operador `^` que estamos usando para bind

- Se usamos `bind` para `->` então a função descrita por id pode afetar a análise, já que ela retorna um parser

Capturas

- Uma outra maneira de ter ações semânticas em PEGs é mudar o resultado do tipo parser para ter uma lista de *capturas* como primeiro elemento
- Os parsers primitivos `''` e `'a'` produzem listas vazias
- Uma *captura* `{p}` captura a entrada antes de aplicar `p` para obter o prefixo que foi consumido por `p`, produzindo a lista de capturas de `p` mais esse prefixo
- Uma ação `p -> id` agora passa cada elemento da lista de capturas de `p` como argumento para a função `id`, e os valores retornados por `id` formam a lista de capturas de `p -> id`
- A lista de capturas não é a lista de resultados de um parser não-determinístico! `({resultado}, resto)` vs `{(resultado, resto)}`