Tópicos em LP

Fabio Mascarenhas – 2018.1

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

Exercício

- Construa um analisador sintático para a gramática a seguir, usando os tokens gerados pelo analisador léxico:

```
exp  -> exp aop term | term
term -> term mop fac | fac
fac  -> number | id | '(' exp ')'
aop  -> '+' | '-'
mop  -> '*' | '/'
```

- Modifique o parser para fazer análise léxica em paralelo com a análise sintática (analisador sintático "scannerless")

Parsers determinísticos

- Se todos os nossos parsers primitivos produzem no máximo um resultado, a única maneira de um parser produzir mais de um resultado é usando o combinador `choice`

- Abrindo mão dele temos parsers que sempre produzem no máximo um resultado

- Assim podemos simplificar o tipo do nosso parser: ao invés de produzir uma lista de resultados, ele produz apenas um par `(resultado, resto)` ou `nil`, que sinaliza uma *falha*

PEGs

- As gramáticas de expressões de parsing, ou PEGs (parsing expression grammars) são uma linguagem para especificar parsers determinísticos

- Ao contrário das gramáticas livres de contexto, PEGs têm um mapeamento natural para os combinadores que estamos usando

- Uma *expressão de parsing* pode ser a expressão vazia `''`, um *terminal* `'a'`, um *não-terminal* `A`, uma *sequência* `pq`, onde `p` e `q` são expressões de parsing, uma *escolha ordenada* `p/q`, uma *repetição* `p*`, ou um *predicado* `!p`

Não-terminais e gramáticas

- Uma PEG é um mapeamento de não-terminais para expressões de parsing
- Quando aplicamos o parser de uma PEG a uma entrada, fica implícito que qualquer não-terminal encontrado é resolvido no contexto dessa PEG: o efeito de aplicar um não-terminal é o efeito de aplicar a expressão de parsing correspondente
- Não-terminais dão o poder de *recursão* às PEGs, mas com duas restrições:
  - Todo não-terminal referenciado tem que ser definido
  - Não pode haver *recursão à esquerda* direta ou indireta

Ações

- Uma maneira de introduzir ações semânticas em nossa linguagem de PEGs é ter uma sintaxe para um operador similar ao combinador `map`, 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 map

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)}`