Linguagens de Domínio Específico

Fabio Mascarenhas – 2017.1

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

Construindo ASTs (2)

- Com combinadores, precisamos fazer os combinadores poderem retornar um pedaço da AST, além do restante da entrada

- Com tipos genéricos podemos manter combinadores bem genéricos, mas que constroem ASTs heterogêneas

```
public interface Parser<A> {
  Result<A> parse(String in);
}
```

```
public class Result<A> {
  public final A res;
  public final String out;
  ...
}
```

- O combinador de sequência fica mais complicado: precisamos passar cada sequência através de uma função responsável por combinar os pedaços da sequência

Combinadores de listas

- Podemos ter um combinador especializado em produzir listas com algum separador:

```
public class ListOf<A> implements Parser<List<A>> {
  public final Parser<List<A>> p;

  public ListOf(Parser<? extends A> _p, Parser<?> _sep) {
    p = seq(_p, star(seqr(_sep, _p)),
                (A a, List<A> la) -> {
                  la.add(0, a);
                  return la;
          });
  }

  public Result<List<A>> parse(State<Character> in) {
    return p.parse(in);
  }
}
```

Combinadores chainl e chainr

- Para listas de expressões binárias, também podemos criar combinadores especializados
- A ideia é que agora o separador produz uma função que pega o lado esquerdo e o lado direito e combina em uma nova
- Ao invés de produzir uma lista ele vai acumulando o resultado, como no analisador recursivo
- Podemos ter uma versão para associatividade à esquerda e uma para associatividade à direita