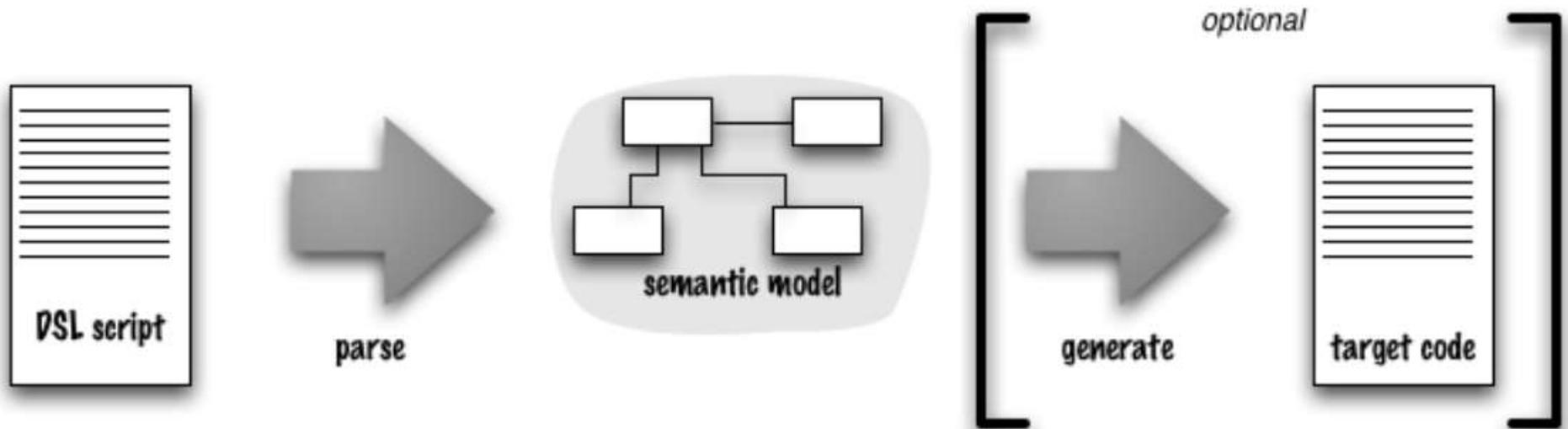


Linguagens de Domínio Específico

Fabio Mascarenhas – 2016.1

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

Processamento de uma DSL



Construindo ASTs com combinadores

- Com combinadores, precisamos fazer os combinadores poderem retornar um pedaço da AST, além do sufixo
- Com tipos genéricos podemos manter combinadores bem genéricos, mas que constroem ASTs heterogêneas

```
public interface Parser<A,B> {  
    Result<A,B> parse(State<B> in);  
}
```

```
public class Result<A,B> {  
    public final A res;  
    public final State<B> 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(_sep.seqr(_p)),
            (A a, List<A> la) -> {
                List<A> res = new ArrayList<A>();
                res.add(a); res.addAll(la);
                return res;
            });
    }

    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