Introdução à Compilação: Linguagem *Straight-line*

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2017.1

1 A linguagem *straight-line*

Tópicos

1 A linguagem *straight-line*

Straight-line

- *Straight-line*é uma micro linguagem de programação usada na série de livros *Modern Compiler Implementation* escritos por Andrew Appel.
- É uma linguagem de comandos e expressões muito simples, sem laços e estruturas condicionais.
- A sintaxe é indicada por uma gramática livre de contexto.
- Exemplo de programa:

```
a := 5+3;
b := (print(a, a-1), 10*a);
print(b)
```

- Quando executado, produz a saída

```
8 7
80
```

Sintaxe de *straight-line*

Stm → *Stm* `;` *Stm*	**CompoundStm**
Stm → `id :=` *Exp*	**AssignStm**
Stm → `print (` *ExpList* `)`	**PrintStm**
Exp → `id`	**IdExp**
Exp → `num`	**NumExp**
Exp → *Exp* *Binop* *Exp*	**OpExp**
Exp → `(` *Stm* `,` *Exp* `)`	**EseqExp**
ExpList → *Exp* `,` *ExpList*	**PairExpList**
ExpList → *Exp*	**LastExpList**
Binop → `+`	**Plus**
Binop → `-`	**Minus**
Binop → `*`	**Times**
Binop → `/`	**Div**

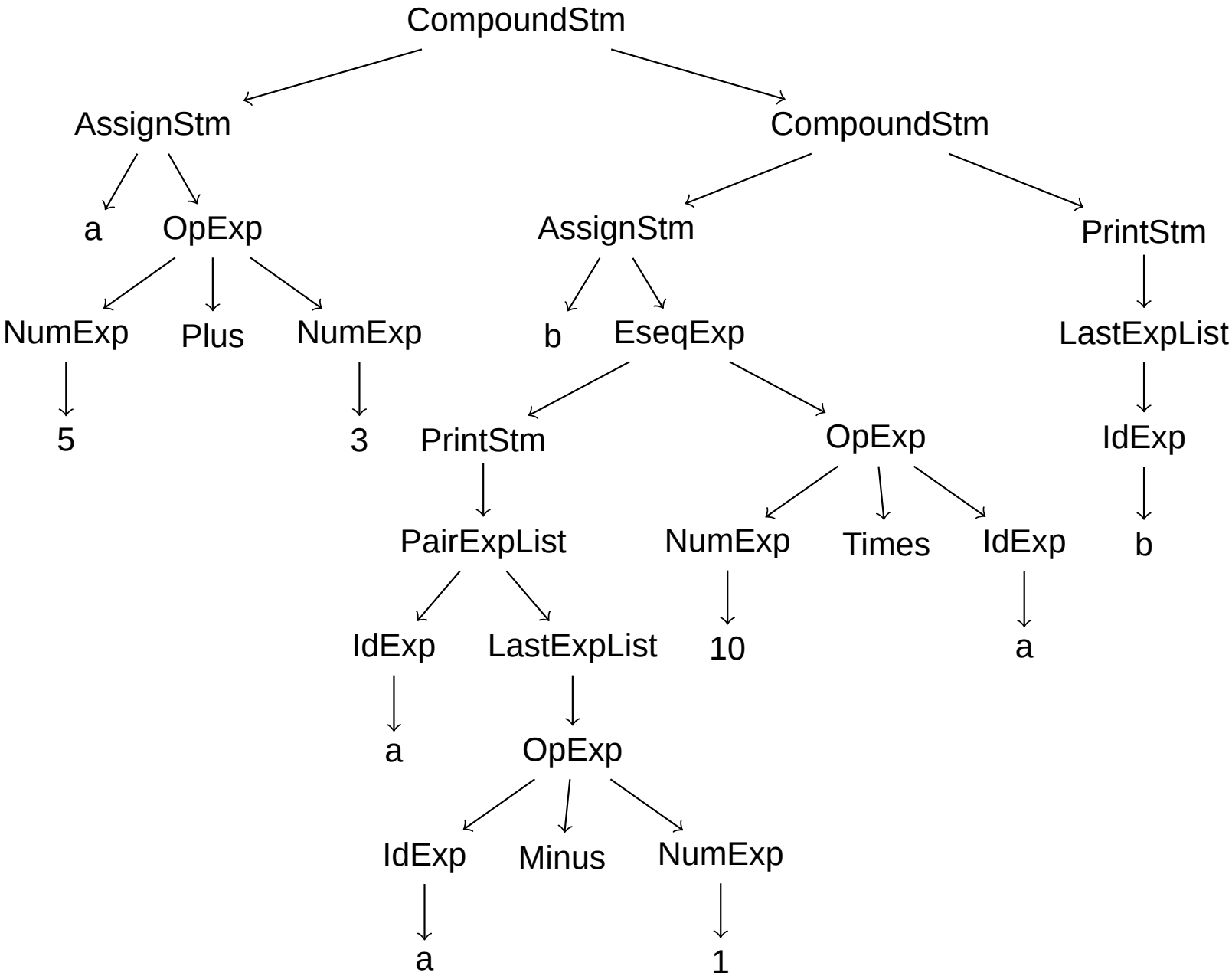
Semântica informal de *straight-line*

- s_1`;` s_2 executa o comando s_1 e em seguida o comando s_2
- *i* `:=` *e* avalia a expressão *e*, e então armazena o resultado na variável *i*
- `print(`e_1`,` e_2`, ···,` e_n`)` exibe os valores de todas as expressões avaliadas da esquerda para a direita, separados por por espaços, terminando por uma mudança de linha
- uma expressão identificador *i* resulta no valor atual da variável *i*
- um número resulta no próprio valor numérico
- uma operação binária e_1 *op* e_2 avalia $e_!$, e então avalia e_2, e então aplica a operação dada aos valores obtidos
- uma expressão sequência `(`*s*`,` *e*`)` executa o comando *s* (pelos seus efeitos colaterais), e então avalia a expressão *e*.

Representação de programas internamente no compilador

- Código fonte (sequência linear de caracteres): inconveniente
- Estrutura de dados **árvore**, com um nó para cada frase no programa: comandos (*Stm*) e expressões (*Exp*).
- Os símbolos do lado direito da regra de produção correspondem a subárvores.
- A gramática pode ser traduzida diretamente para definições de estruturas de dados.
- Cada símbolo gramatical corresponde a um tipo na estrutura de dados.
- Para cada regra de produção há um construtor que pertence ao tipo correspondente ao seu lado esquerdo.

Árvore sintática



Tipos na linguagem OCaml

```
type id = string

type binop = Plus | Minus | Times | Div

type stm = CompoundStm of stm * stm
         | AssignStm of id * exp
         | PrintStm of exp list

and exp = IdExp of id
        | NumExp ofint
        | OpExp of exp * binop * exp
        | EseqExp of stm * exp
```