Construção de Compiladores
Aula 9 - Comando If

Bruno Müller Junior

Departamento de Informática
UFPR

11 de Setembro de 2014

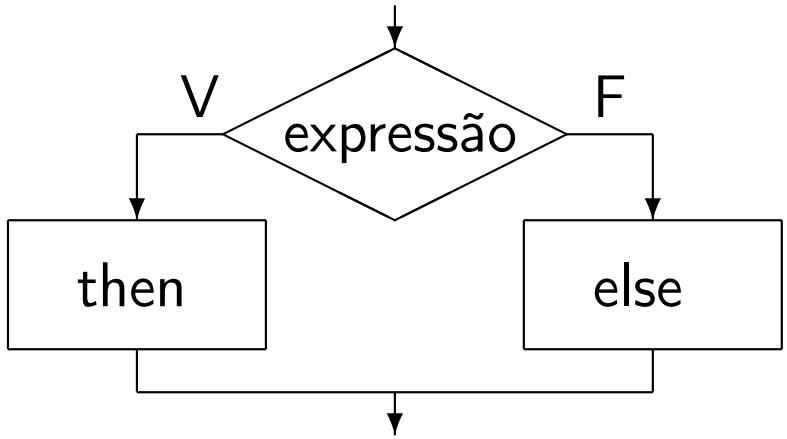
Sintaxe

- O comando `while` (regra 22) é uma das possibilidades de `<comando sem rótulo>` (regra 18).
- O analisador sintático irá escolher a regra 18 quando estiver na regra 23 e encontrar o token `IF`.
- É importante destacar que o comando `ELSE` é opcional, cuja gramática em formato ascendente (bison) é ambígua.

```
18.  <comando sem rótulo> ::= <atribuição>           |
                              ...                    |
                              <comando condicional>
22. <comando condicional>  ::= IF <expressão>
                                  THEN <comando sem rótulo>
                                  [ELSE <comando sem rótulo>]
```

Fluxo

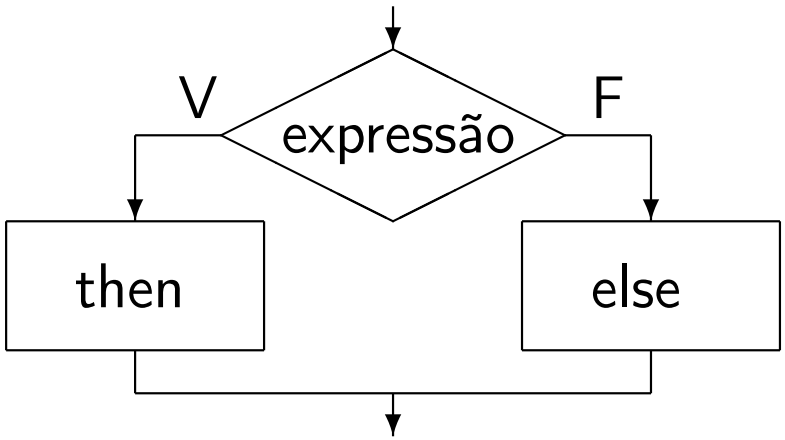
- Assim como no comando `WHILE`, o fluxo de execução do comando `IF` não é sequencial.
- Também existem desvios do fluxo para locais específicos.
- Porém, a figura ao lado não ajuda muito a ver onde colocar estes desvios de fluxo.



Esquema de Tradução

- A maneira mais simples de ver onde inserir os comandos é no esquema de tradução.

IF	
E	*Traduz Expressão* `DSVF R00`
THEN	
<comandos>	*Traduz Comandos* `DSVS R01`
ELSE	`R00: NADA`
<comandos>	*Traduz Comandos* `R01: NADA`



Tradução

```
1  program comandoIf (input, output);
2        var   i, j: integer;
3  begin
4        read(j)
5        i:=0;
6        while (i < j) do
7        begin
8              if (i div 2 * 2 = i)
9                    then write(i,0)
10                   else write(i,1);
11             i := i+1
12       end;
13 end.
```

Projeto

- Uma das formas de implementar o comando `IF` para bison implica na criação de uma gramática ambígua

```
comando condicional  : IF expressão THEN comSemRot |
                       IF expressão THEN comSemRot ELSE comSemRot
```

- O que pode ser comprovado ao criar as duas árvores sintáticas para o exemplo abaixo.

```
if E1 then
   if E2 then C1
else C2 (* A quem pertence este else? *)
```

Soluções

- Os primeiros compiladores da linguagem `Algol 60` não observaram este fato.
- Alguns associaram o `ELSE` ao primeiro `IF` e outros ao segundo (um mesmo programa gerava dois códigos diferentes).
- Na bibliografia, este problema é conhecido como *dangling else*, e foi estipulado que o `ELSE` deve ser associado ao `IF` mais interno.
- A implantação disto em bison não é "bonita", e implica em associar prioridades a tokens.

Bison

- Existem várias soluções para isto em bison. A mais simples de e explicar é associar precedências aos operadores.
- Assim, ao encontrar um ELSE, o bison deverá priorizar a conclusão da árvore sintática do ELSE.

```
// Precedências são crescentes, logo "lower_than_else" < "else".
%nonassoc "lower_then_else"
%nonassoc "else"
%%
stm: "if" "(" exp ")" stm  %prec "lower_then_else" |
     "if" "(" exp ")" stm "else" stm
```

- Uma alternativa detalhada pode ser encontrado em http://www.inf.ufpr.br/bmuller/CI211/IF.y