# Implementação em Prolog de Analisadores Gramaticais para Gramáticas de Estrutura Sintagmática Independentes de Contexto

**Luiz Arthur Pagani (UFPR)**

http://www.ufpr.br/~arthur

arthur@ufpr.br

# Sumário

# Resumo

Um **analisador gramatical** (AG) em **Prolog** para uma **gramática de estrutura sintagmática independente de contexto** (GES-IC) é um algoritmo, especificado através do Prolog (uma linguagem de programação usada no Processamento de Língua Natural e na Inteligência Artificial), que, a partir de uma expressão lingüística e de uma GES-IC, infere a estrutura gramatical da expressão (ou mais de uma, caso ela seja ambígua), além do seu mero ordenamento seqüencial.

Os AGs têm sido classificados, pela sua operação, em duas dimensões:

1. sentido:
   - se começam analisando as palavras para depois aplicar as regras sintagmáticas na construção da estrutura (**busca ascendente**),
   - ou se começam construindo uma estrutura, a partir das regras sintagmáticas, para depois conferir se ela combina com a expressão analisada (**busca descendente**); e
2. percurso:
   - se a análise percorre primeiro todo um ramo da árvore, antes de percorrer outro (**busca em profundidade**),
   - ou se a análise produz todos os nós irmãos antes de produzir os seus nós descendentes (**busca em largura**).

A partir destas possibilidades, algumas implementações se destacaram, recebendo nomes específicos. Assim, nos interpretadores de Prolog, é comum dispormos de uma notação, conhecida como **DCG**, na qual podemos representar facilmente gramáticas de estrutura sintagmática de uma forma muito parecida com o formato em que os lingüistas escrevem as suas regras de constituência. Além da DCG, destacam-se ainda o **analisador gramatical por deslocamento e redução** e os **analisadores tabulares**, dentre os quais o mais importante talvez seja o chamado "**algoritmo de Earley**".

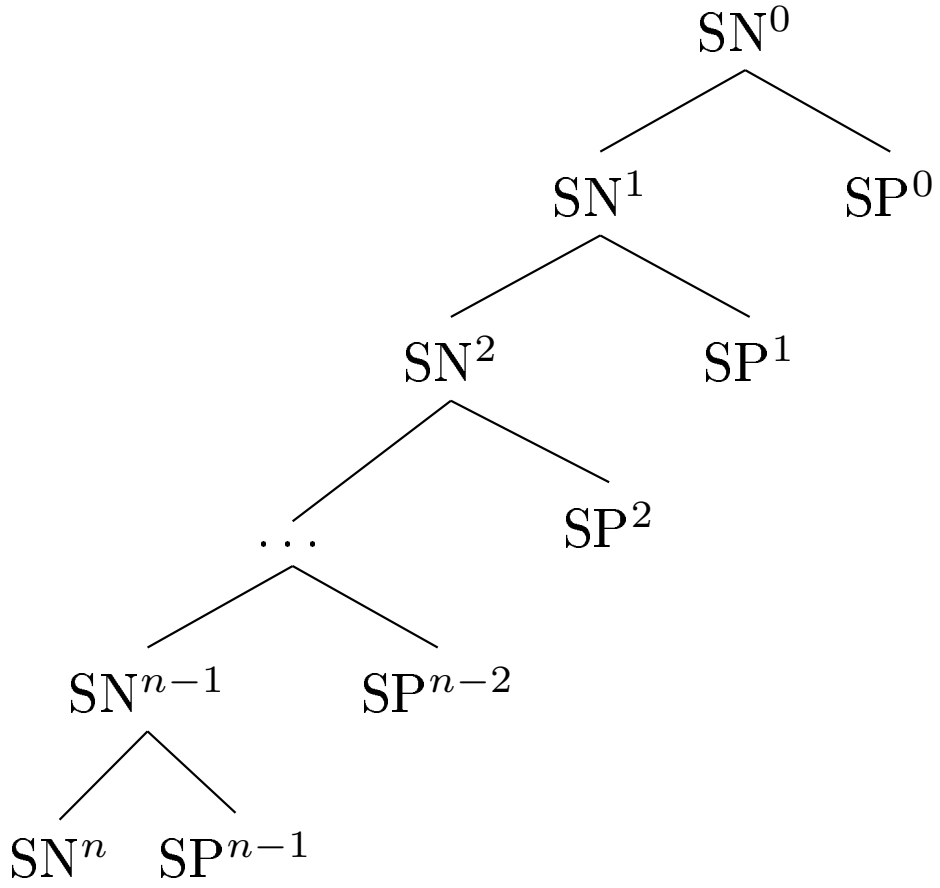
Do ponto de vista lingüístico, os analisadores gramaticais não são apenas ferramentas para produzir automaticamente uma análise gramatical, mas também podem ser usados como **modelo** do **processamento lingüístico humano** (PLH), replicando não só a acuidade das análises gramaticais, mas também as **preferências de análise** exibidas pelos falantes das línguas.

# 1 Introdução

## 1.1 Apresentação

- Dois aspectos do trabalho de investigação lingüística:
  - princípios de estruturação das expressões das línguas
  - heurística para construir uma análise das expressões
- Outra distinção metodológica:
  - conhecimento lingüístico dos falantes (ou, do ponto de vista metateórico, o conhecimento do lingüista)
  - aplicação que os falantes fazem desses conhecimentos para processar as expressões (ou, ainda metateoricamente, a explicação produzida pelo lingüista)

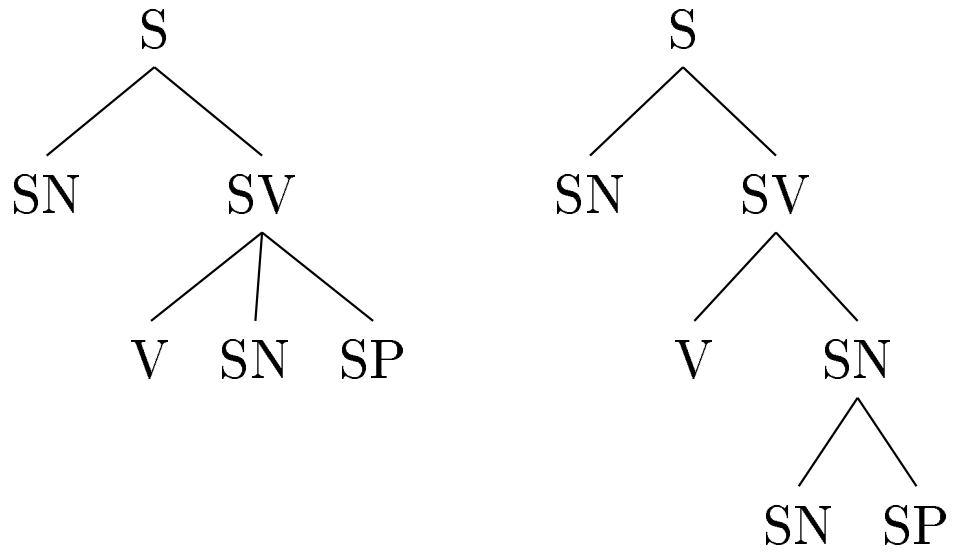
- Recursão infinita com regra recursiva à esquerda
  - $SN \rightarrow SN\ SP$



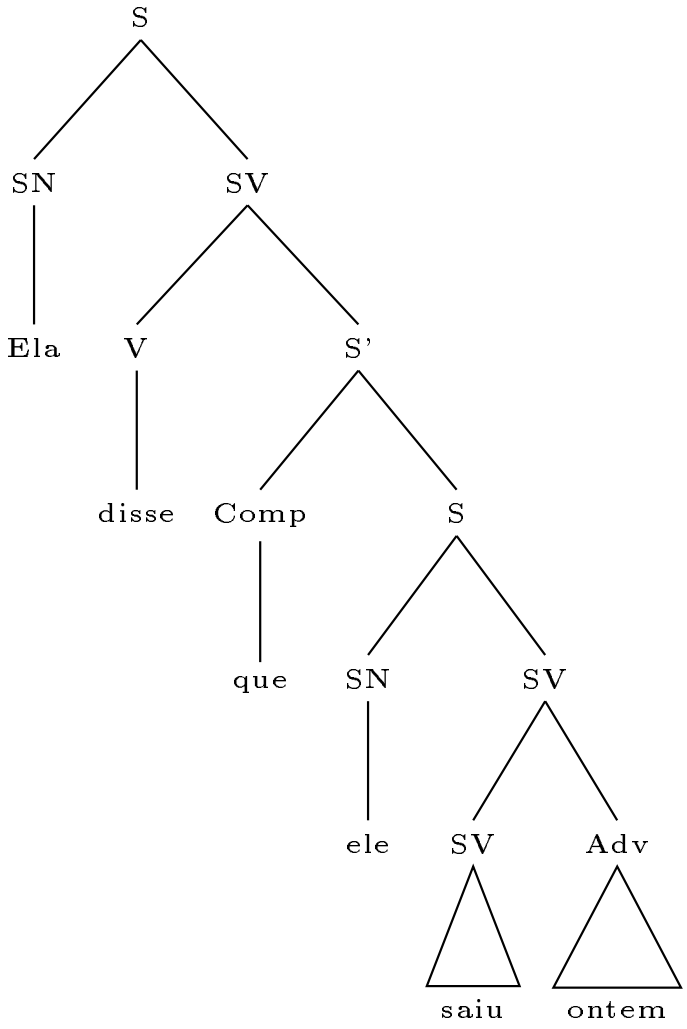
- Preferência de análises
  - Ambigüidade global × local:
    - "O homem viu a mulher com o telescópio"
    - "Quando a Maria estava costurando a blusa caiu"
  - Sentenças-labirinto
    - "The horse raced past the barn fell"
    - "O navio angolano entrava no porto o navio brasileiro"
  - Sete princípios, de Kimball
  - Modelo em dois estágios, de Frazier & Fodor

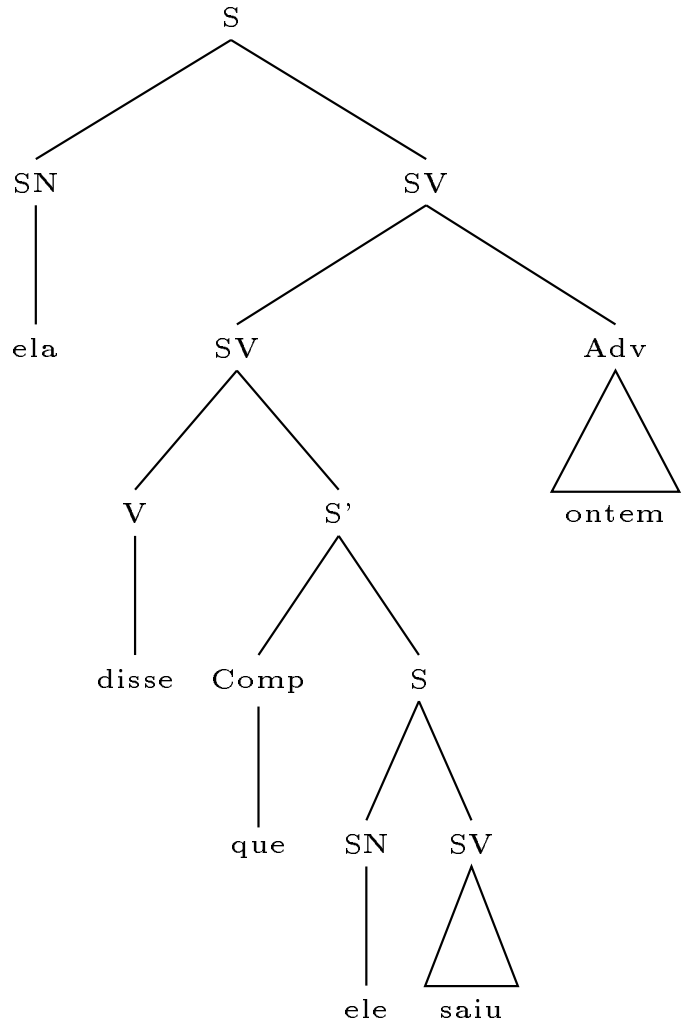
– Princípios descritivos

  * Anexação mínima (*minimal attachment*)



* Associação à direita (*right association*; ou encerramento tardio (*late closure*))

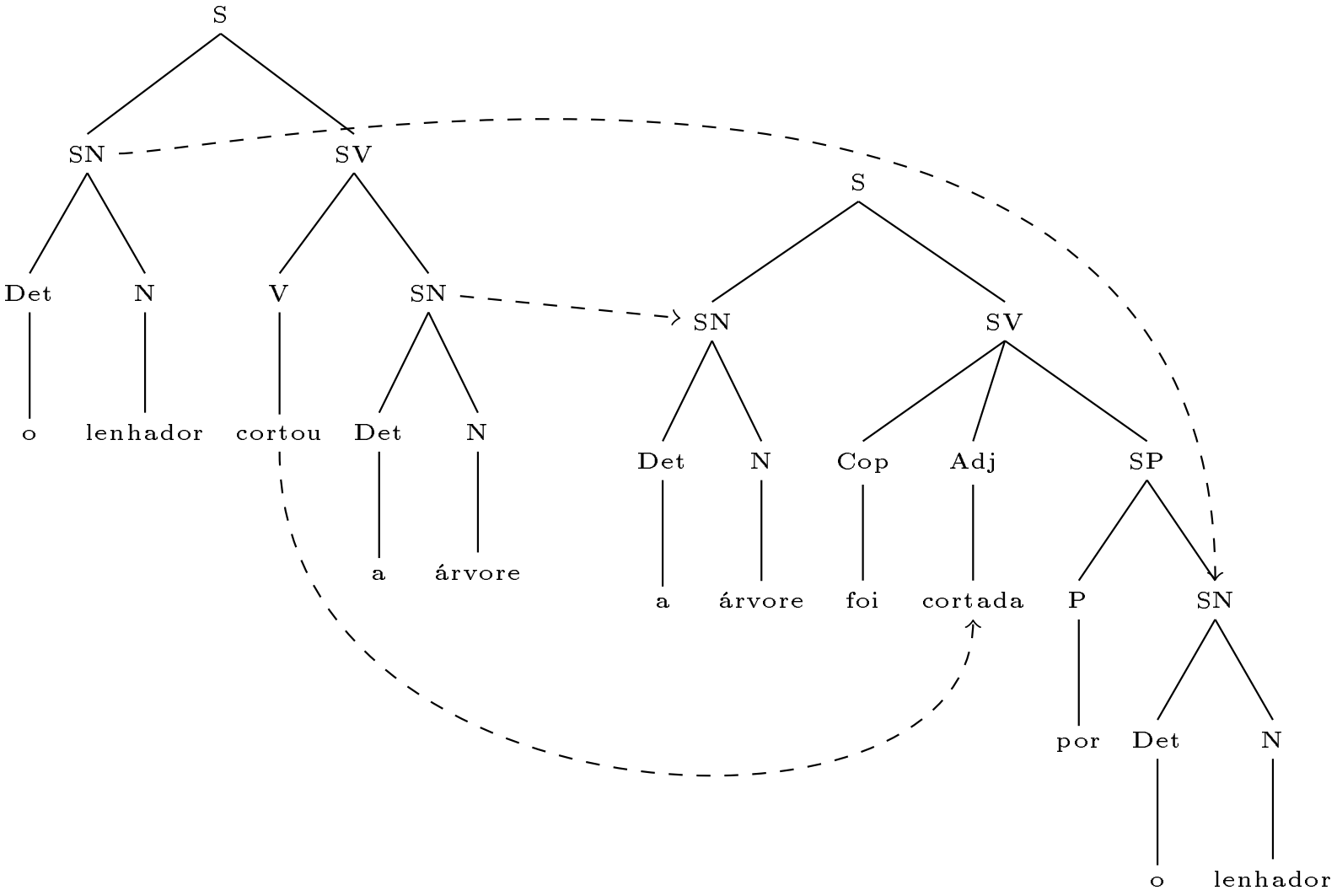


## 1.2 Expressão, gramática e análise gramatical

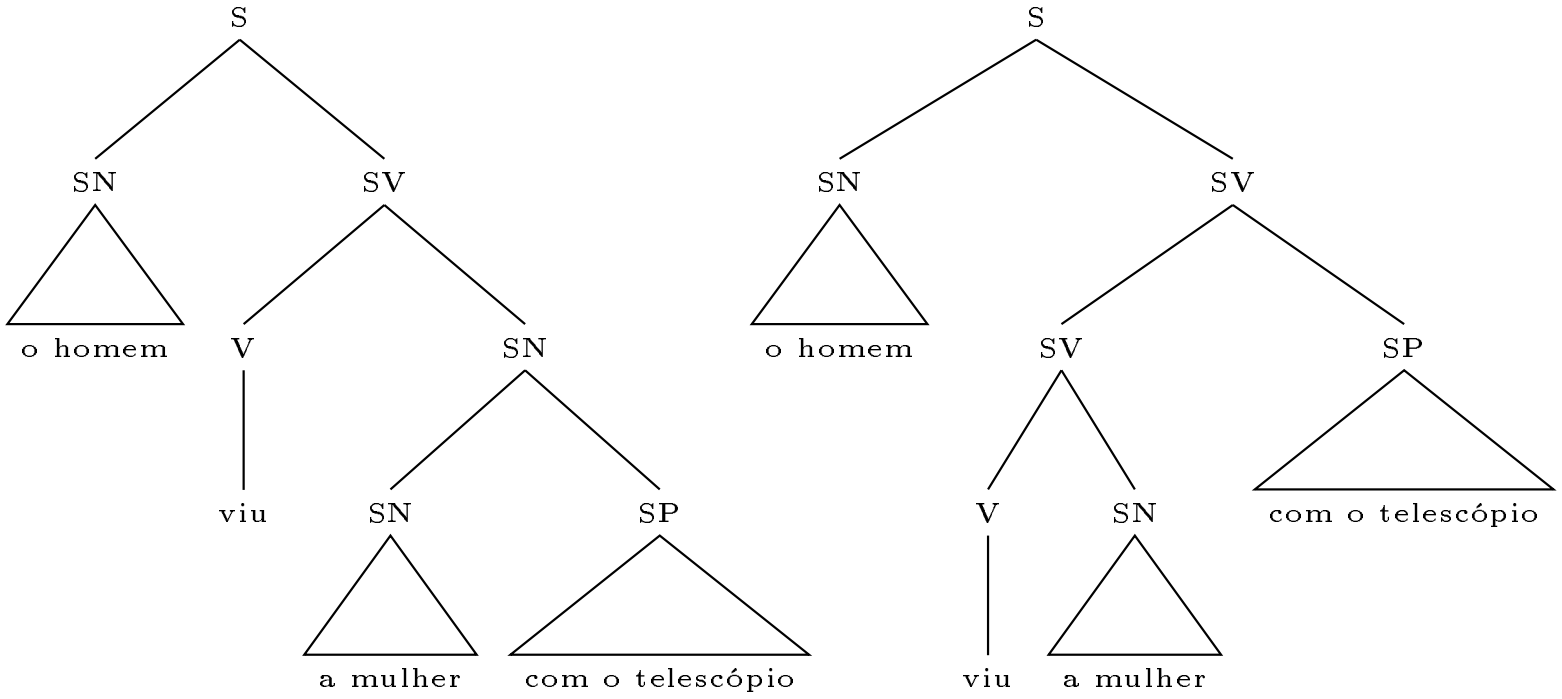
- "ordenação das unidades em seqüência linear, de maneira que a realização substancial de um elemento deva preceder, no tempo, a realização de outro elemento" (Lyons 1979: 78)
- as expressões complexas "não são apenas seqüências lineares de elementos, mas formam-se de 'camadas' de *constituintes imediatos*, cada constituinte de nível inferior fazendo parte de um constituinte de nível superior" (Lyons1979: 221)
- eixos saussureanos:
  - paradigmático
  - sintagmático:
    - seqüência
    - hierarquia

- ativa × passiva



- ambigüidade estrutural:

  "o homem viu a mulher com o telescópio"

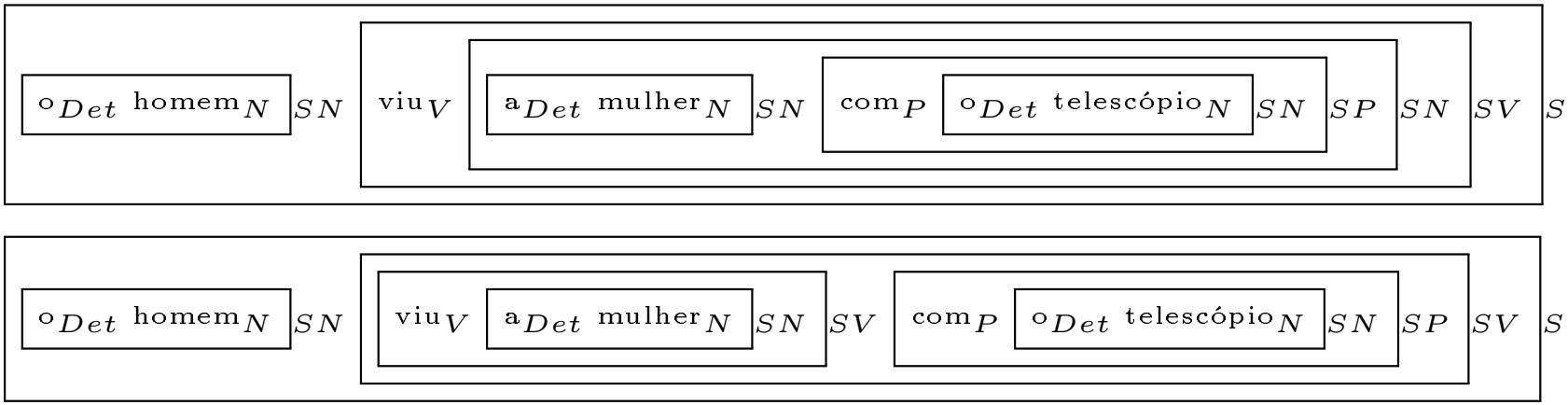


- parênteses rotulados

[ [ o homem ]$_{SN}$ [ viu [ [ a mulher ]$_{SN}$ [ com [ o telescópio ]$_{SN}$ ]$_{SP}$ ]$_{SN}$ ]$_{SV}$ ]$_{S}$

[ [o homem]$_{SN}$ [ [ viu [ a mulher ]$_{SN}$ ]$_{SV}$ [ com [ o telescópio ]$_{SN}$ ]$_{SP}$ ]$_{SV}$ ]$_{S}$

- diagrama em blocos



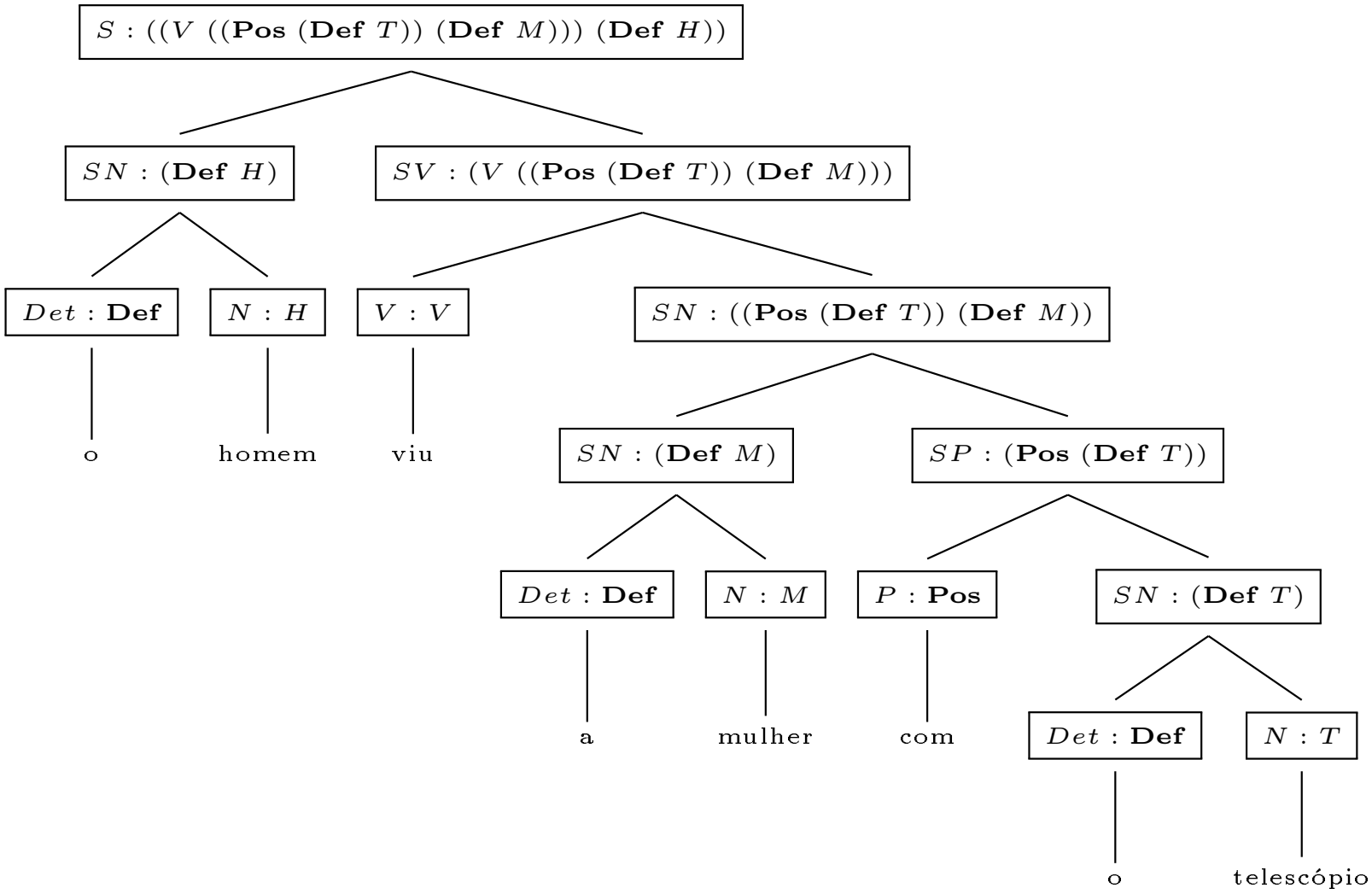
### 1.2.1 GES-IC: sintaxe

| Regras sintagmáticas | | | | Regras lexicais | | |
|---|---|---|---|---|---|---|
| $S$ | $\rightarrow$ | $SN$ | $SV$ | $Det$ | $\rightarrow$ | a |
| $SN$ | $\rightarrow$ | $Det$ | $N$ | $Det$ | $\rightarrow$ | o |
| $SN$ | $\rightarrow$ | $SN$ | $SP$ | $N$ | $\rightarrow$ | homem |
| $SV$ | $\rightarrow$ | $V$ | $SN$ | $N$ | $\rightarrow$ | mulher |
| $SV$ | $\rightarrow$ | $SV$ | $SP$ | $N$ | $\rightarrow$ | telescópio |
| $SP$ | $\rightarrow$ | $P$ | $SN$ | $V$ | $\rightarrow$ | viu |
| | | | | $P$ | $\rightarrow$ | com |

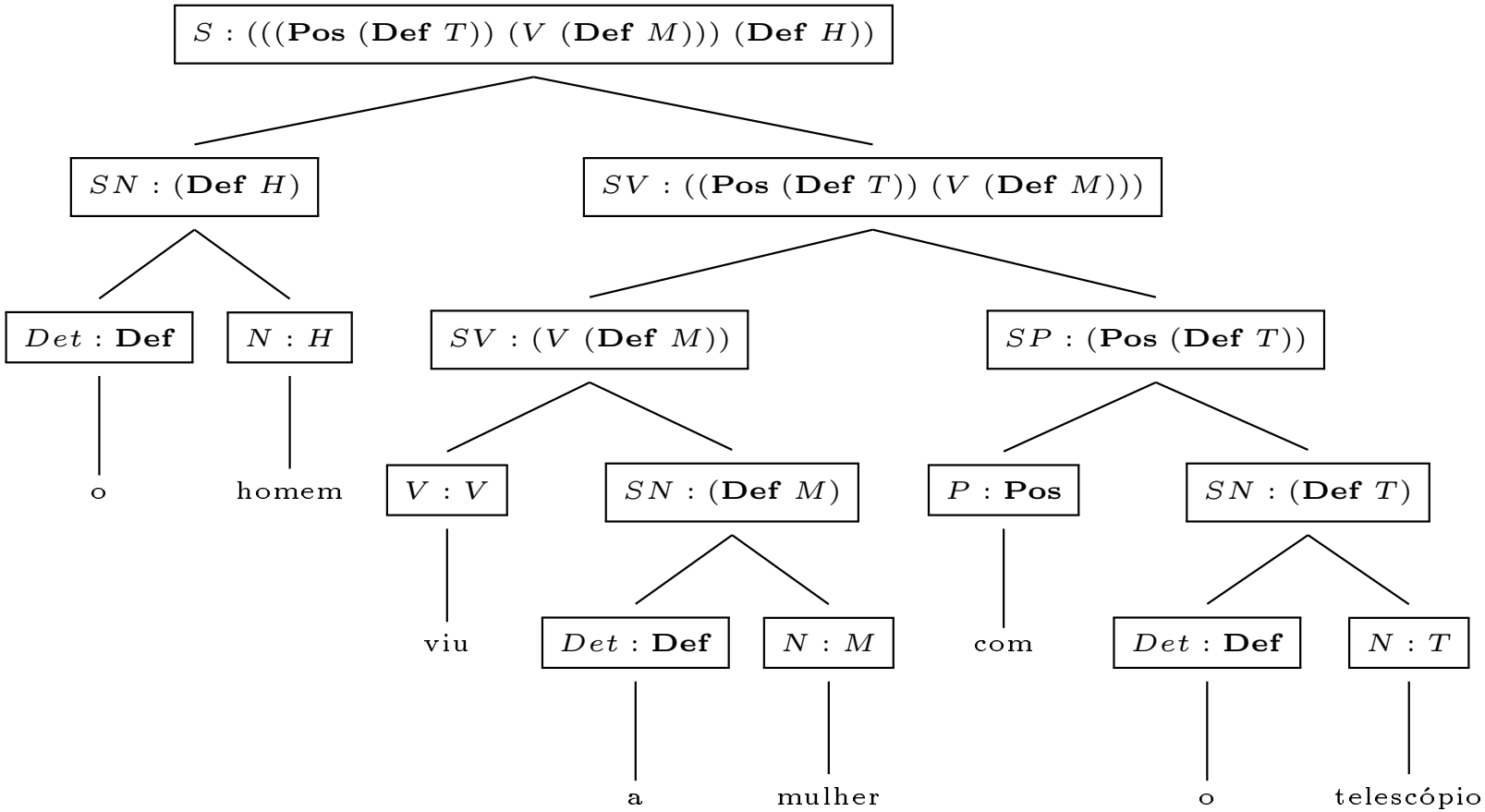
### 1.2.2 GES-IC: semântica

| Regras sintagmáticas | | | Regras lexicais | | |
|---|---|---|---|---|---|
| $[\![S]\!]$ | $=$ | $([\![SV]\!]\ [\![SN]\!])$ | $[\![\text{a}]\!]$ | $=$ | **Def** |
| $[\![SN]\!]$ | $=$ | $([\![Det]\!]\ [\![N]\!])$ | $[\![\text{o}]\!]$ | $=$ | **Def** |
| $[\![SN]\!]$ | $=$ | $([\![SP]\!]\ [\![SN]\!])$ | $[\![\text{homem}]\!]$ | $=$ | $H$ |
| $[\![SV]\!]$ | $=$ | $([\![V]\!]\ [\![SN]\!])$ | $[\![\text{mulher}]\!]$ | $=$ | $M$ |
| $[\![SV]\!]$ | $=$ | $([\![SP]\!]\ [\![SV]\!])$ | $[\![\text{telescópio}]\!]$ | $=$ | $T$ |
| $[\![SP]\!]$ | $=$ | $([\![P]\!]\ [\![SN]\!])$ | $[\![\text{viu}]\!]$ | $=$ | $V$ |
| | | | $[\![\text{com}]\!]$ | $=$ | **Pos** |

- **adjunção adnominal**: 'o evento de ver teve como paciente a mulher que tinha o telescópio como instrumento, e como agente o homem'



- **adjunção adverbial**: 'o evento de ver, que teve o telescópio como instrumento, teve como paciente a mulher, e como agente o homem'



## 1.3 Conhecimento procedimental/declarativo

- Aristóteles & Humboldt:
  - ato:
    - processo: execução da análise
    - produto: resultado da análise
  - potência: gramática
- tipo de processamento:
  - procedimental: "como"
  - declarativo: "o quê"

## 1.4 Reconhecedor e analisador gramatical

- "Um dispositivo computacional que infere uma estrutura a partir de cadeias gramaticais de palavras é conhecido como *analisador gramatical* (*parser*), e a maior parte da história do Processamento de Língua Natural nos últimos 20 anos foi dedicada à elaboração de analisadores gramaticais." (Gazdar & Mellish 1989: 5)
- reconhecedor gramatical: juízo de gramaticalidade (diz apenas se a expressão pertence ou não à língua, mas não infere nenhuma estrutra)

- exemplo rudimentar de reconhecedor

```
% reconhecedor.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

s(S) :-
   append(SN, SV, S),
   sn(SN),
   sv(SV).

sn(SN) :-
   append(Det, N, SN),
   det(Det),
   n(N).

sv(SV) :-
   append(V, SN, SV),
   v(V),
   sn(SN).

det([a]).
det([o]).

n([homem]).
n([mulher]).

v([viu]).
```

```
? - [reconhecedor].<enter>
...
?- s([o, homem, viu, a, mulher]).<enter>
true ;
false.

?-
```

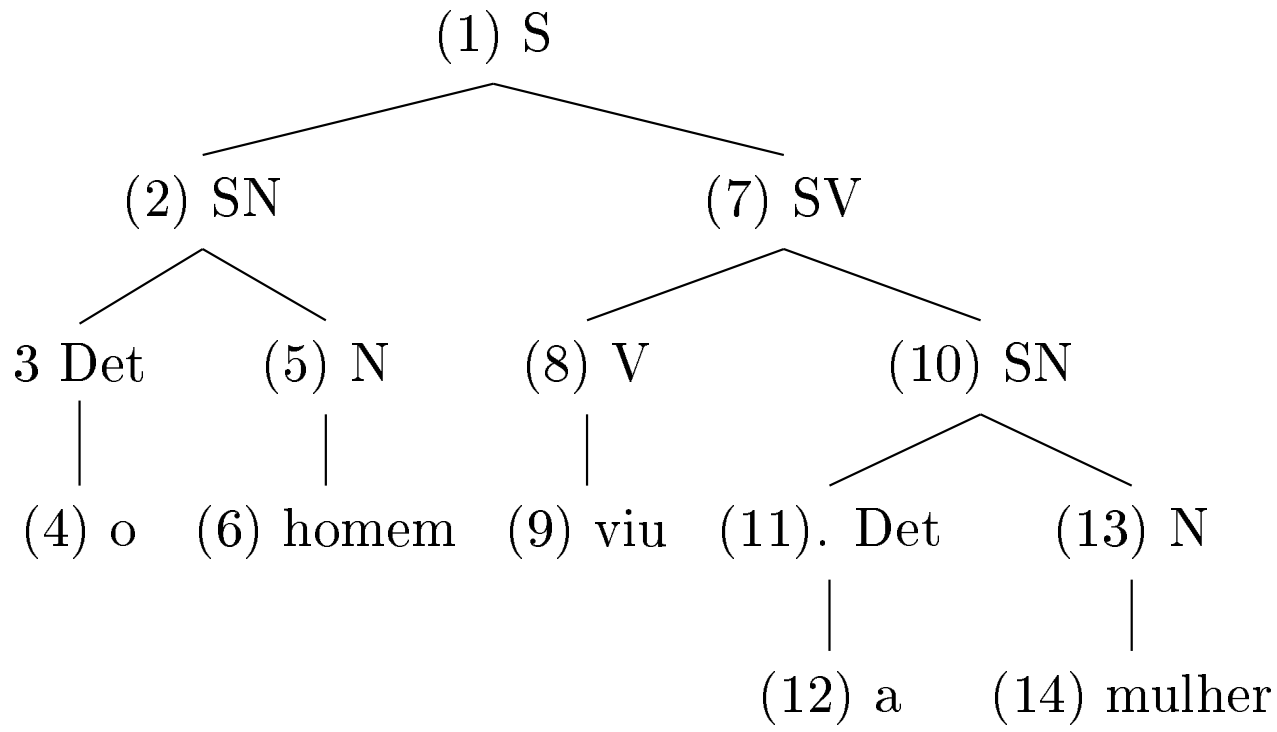
# 2 Tipos de algoritmo para a análise gramatical

- Duas dimensões:

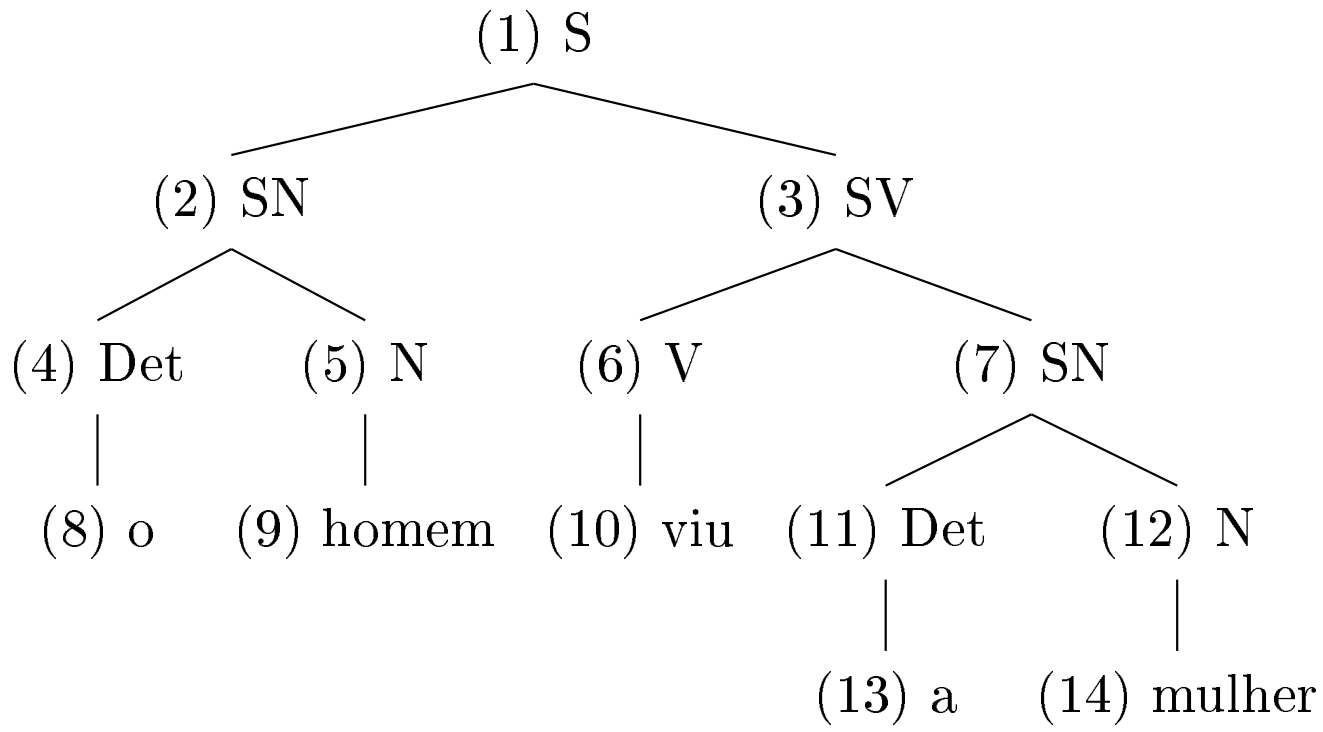
1. sentido:
   (a) começo pela identificação das palavras, para depois aplicar as regras sintagmáticas (**busca ascendente**), ou
   (b) começo construindo uma estrutura, a partir das regras, para depois conferir com a expressão (**busca descendente**); e
2. percurso
   (a) se a análise percorre primeiro todo um ramo da árvore, antes de percorrer outro (**busca em profundidade**), ou
   (b) se a análise produz todos os nós irmãos antes de produzir os seus nós descendentes (**busca em largura**).

- profundidade:
  - pilha: o primeiro elemento a ser armazenado é o último a ser recuperado
    - põe o elemento $a$ na pilha vazia []: $[a]$
    - põe o elemento $b$ na fila $[a]$: $[b, a]$
    - tira um elemento da fila $[b, a]$: $[a]$
- largura:
  - fila: o primeiro elemento a ser armazenado é o primeiro a ser recuperado
    - põe o elemento $a$ na fila vazia []: $[a]$
    - põe o elemento $b$ na fila $[a]$: $[a, b]$
    - tira um elemento da fila $[a, b]$: $[b]$

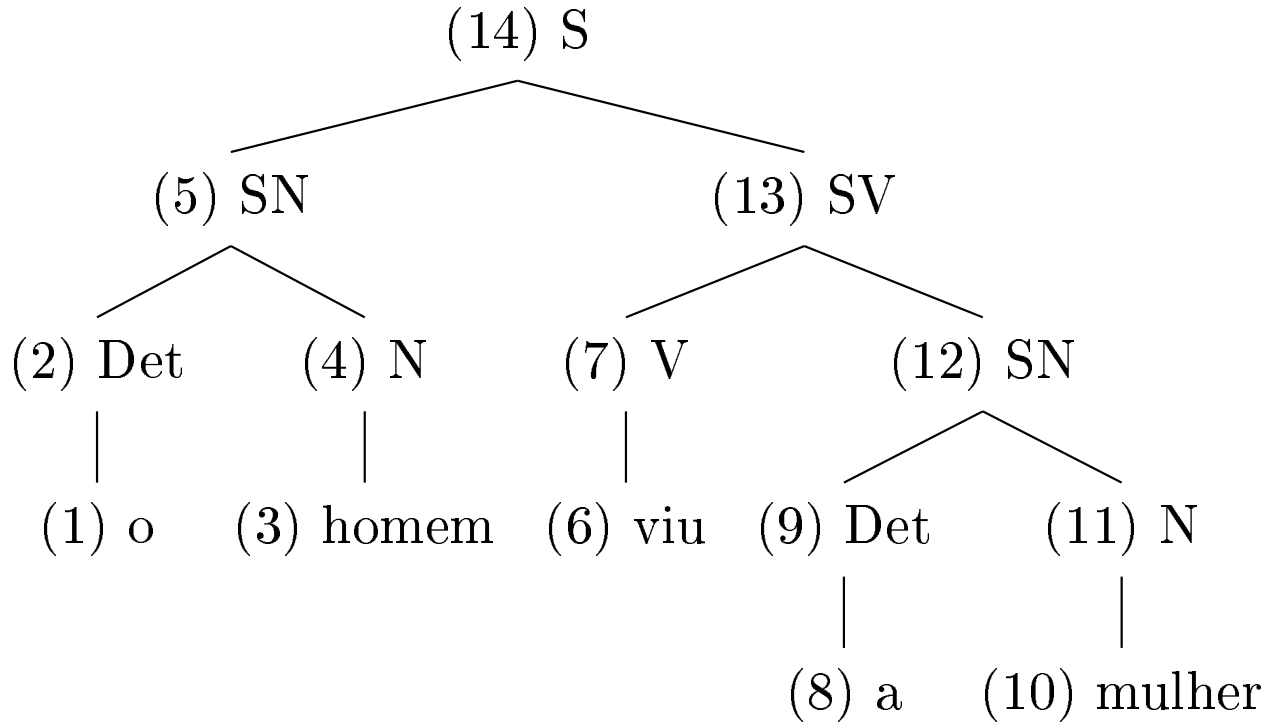
## 2.1 Descendente em profundidade



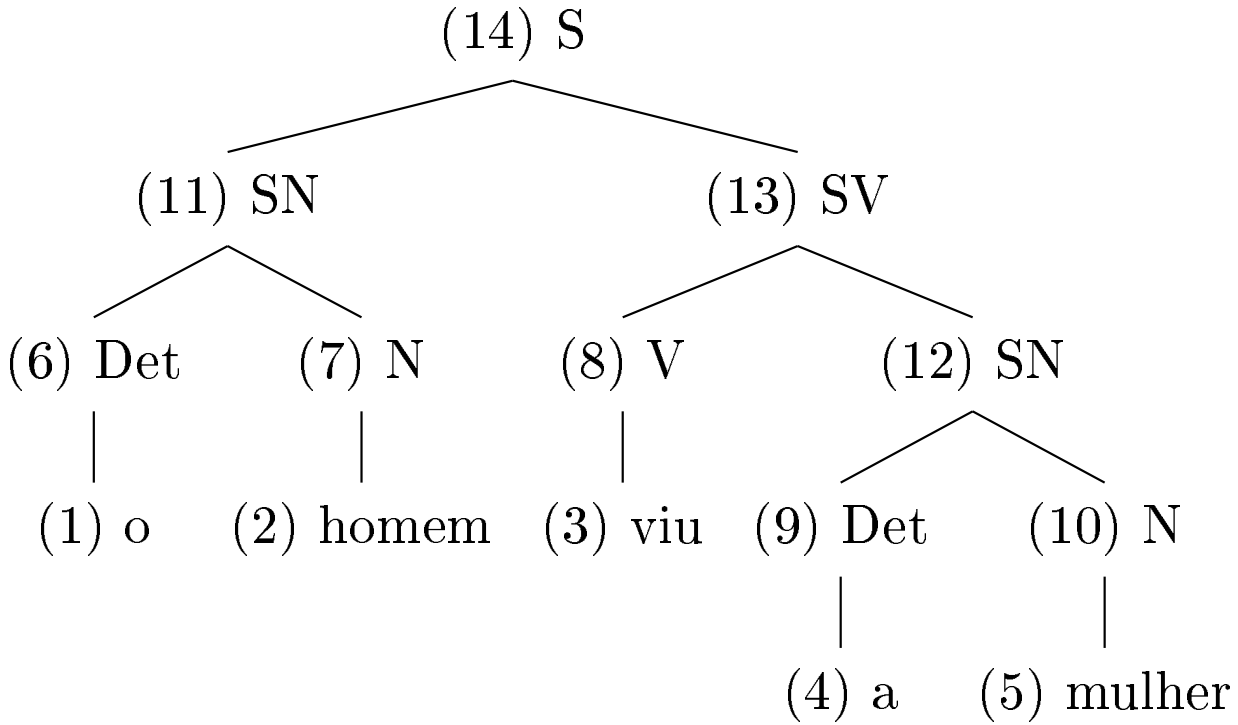
## 2.2 Descendente em largura



## 2.3 Ascendente em profundidade



## 2.4 Ascendente em largura



# 3 Gramática de Cláusula Definida

```
% claus_def.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

s(S, Resto) :-
   sn(S, SV),
   sv(SV, Resto).

sn(SN, Resto) :-
   det(SN, N),
   n(N, Resto).

sv(SV, Resto) :-
   v(SV, SN),
   sn(SN, Resto).

det([a|Resto], Resto).
det([o|Resto], Resto).

n([homem|Resto], Resto).
n([mulher|Resto], Resto).

v([viu|Resto], Resto).
```

```
?- [clau_def].<enter>
...
?- s([o, homem, viu, a, mulher], []).<enter>
true ;
false.

?-
```

## 3.1 Notação

```
% dcg_rec.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

s --> sn, sv.

sn --> det, n.

sv --> v, sn.

det --> [a].
det --> [o].

n --> [homem].
n --> [mulher].

v --> [viu].
```

## 3.2 Argumentos

### 3.2.1 Concordância

```
?- sn([a, homem], []).<enter>
true ;
false.

?-
```

```
% dcg_conc.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

s --> sn, sv.

sn --> det(Gen), n(Gen).

sv --> v, sn.

det(fem) --> [a].
det(masc) --> [o].

n(masc) --> [homem].
n(fem) --> [mulher].

v --> [viu].
```

### 3.2.2 Analisador

```
% dcg_ana.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

:- op(500, xfx, @).

s(SV@SN) --> sn(SN),
             sv(SV).

sn(Det@N) --> det(Det),
              n(N).

sv(V@SN) --> v(V),
             sn(SN).

det('Def') --> [a].
det('Def') --> [o].

n('H') --> [homem].
n('M') --> [mulher].

v('V') --> [viu].
```

```
?- s(Int, [o, homem, viu, a, mulher], []).
Int = ('V'@ ('Def'@'M'))@ ('Def'@'H') ;
false.

?-
```

# 4 Deslocamento e redução

```
% rec_esq.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

s --> sn, sv.

sn --> det, n.
sn --> sn, sp.

sv --> v, sn.
sv --> sv, sp.

sp --> p, sn.
```

```
det --> [a].
det --> [o].

n --> [homem].
n --> [mulher].
n --> [telescópio].

v --> [viu].

p --> [com].
```

```
?- s([o,homem,viu,a,mulher,com,o,telescópio], []).
true ;
ERROR: Out of local stack
   Exception: (7) sv([viu, a, mulher, com, o, telescópio], []) ? abort
% Execution Aborted
?-
```

## 4.1 Algoritmo

- **Deslocamento:** Desloca um elemento para a pilha
- **Redução:** Reduz a pilha
- **Término:** Expressão vazia e apenas uma categoria na pilha

| | **Expressão** | **Pilha** | **Operação** |
|---|---|---|---|
| 1 | o homem viu a mulher | | Desloca ($Det \rightarrow$ o) |
| 2 | homem viu a mulher | $Det$ | Desloca ($N \rightarrow$ homem) |
| 3 | viu a mulher | $Det\ N$ | Reduz ($SN \rightarrow Det\ N$) |
| 4 | viu a mulher | $SN$ | Desloca ($V \rightarrow$ viu) |
| 5 | a mulher | $SN\ V$ | Desloca ($Det \rightarrow$ a) |
| 6 | mulher | $SN\ V\ Det$ | Desloca ($N \rightarrow$ mulher) |
| 7 | | $SN\ V\ Det\ N$ | Reduz ($SN \rightarrow Det\ N$) |
| 8 | | $SN\ V\ SN$ | Reduz ($SV \rightarrow V\ SN$) |
| 9 | | $SN\ SV$ | Reduz ($S \rightarrow SN\ SV$) |
| 10 | | $S$ | Ok |

```
% agdr_dcg_port.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Analisador gramatical      %
% por deslocamento e redução   %
%   em DCG para português      %
%       baseado no de          %
%    Covington 1994: 159       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% analisa(+Expressão, ?Categoria)
%  inicia o processo de análise da Expressão,
%  com a pilha vazia,
%  de forma que em Resultado apareça
%  a lista de categorias a que ela se reduz
analisa(Expressão, Categoria) :-
```

```
  analisa([], Categoria, Expressão, []).

% analisa(+Categorias, ?NovasCategorias, ...)
%  analisa (uma expressão através da DCG),
% a partir de uma pilha de Categorias,
% para chegar nas NovasCategorias
analisa(Categorias, NovasCategorias) -->
  desloca(Categorias, MaisCategorias),
  {reduz(MaisCategorias, CategoriasReduzidas)},
  analisa(CategoriasReduzidas, NovasCategorias).
analisa([Categoria], Categoria) --> [].

% desloca(+Categorias, ?MaisCategorias, ...)
%  desloca a Categoria da próxima Palavra
%  para a pilha de categorias
desloca(Categorias, [Categoria|Categorias]) --> [Palavra],
  {palavra(Categoria, Palavra)}.
```

```
% reduz(+Categorias, ?CategoriasReduzidas)
%  reduz repetidamente a pilha de Categorias
reduz(Categorias, CategoriasReduzidas) :-
  regra_invertida(Categorias, OutrasCategorias),
  reduz(OutrasCategorias, CategoriasReduzidas).
reduz(Categorias, Categorias).

% regra_invertida(+LDR, ?LER)
%  uma regra de estrutura sintagmática como A --> B C
%  fica como regra_invertida([B, C | Resto], [A | Resto])
regra_invertida([C, B | X], [A | X]) :-
  regra(A, [B, C]).

% Carrega gramática (regras e léxico)
:- [gram].


?- analisa([o,homem,viu,a,mulher,com,o,telescópio], Cat).
Cat = s: (('Pos'@ ('Def'@'T'))@ ('V'@ ('Def'@'N')))@ ('Def'@'H') ;
```

```
Cat = s: ('V'@ (('Pos'@ ('Def'@'T'))@ ('Def'@'N')))@ ('Def'@'H') ;
false.
```

### 4.1.1 Gramática

```
% gram.pl
% -------------------
% Luiz Arthur Pagani
% arthur@ufpr.br

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gramática para implementação %
% de analisadores gramaticais  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Operadores
% -----------
% Aplicação funcional
:- op(500, xfx, @).
% Relação entre classe e interpretação
:- op(600, xfx, :).

% Regras gramaticais
% -------------------
```

```
% S --> SN SV
regra(s:SV@SN, [sn:SN, sv:SV]).
% SN --> Det N
regra(sn:Det@N, [det:Det, n:N]).
% SN --> SN SP
regra(sn:SP@SN, [sn:SN, sp:SP]).
% SV --> V SN
regra(sv:V@SN, [v:V, sn:SN]).
% SV --> SV SP
regra(sv:SP@SV, [sv:SV, sp:SP]).
% SP --> P SN
regra(sp:P@SN, [p:P, sn:SN]).
% Det --> 0 (para determinante nulo)
regra(det:'Indef', []).

% Carrega o léxico
:- [lex].
```

### 4.1.2 Léxico

```
% lex.pl
% -------------------
% Luiz Arthur Pagani
% arthur@ufpr.br

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Léxico para implementação  %
% de analisadores gramaticais %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Det:Def --> a
palavra(det:'Def', a).
% Det:Def --> o
palavra(det:'Def', o).
% N:H --> homem
```

```
palavra(n:'H', homem).
% N:M --> mulher
palavra(n:'M', mulher).
% N:T --> telescópio
palavra(n:'T', telescópio).
% V:V --> viu
palavra(v:'V', viu).
% P:Pos --> com
palavra(p:'Pos', com).
```

# 5 Tabular

| | **Expressão** | **Pilha** | **Operação** |
|---|---|---|---|
| 1 | viu a mulher com o telescópio | | Desloca |
| 2 | a mulher com o telescópio | *V* | Desloca |
| 3 | mulher com o telescópio | *V Det* | Desloca |
| 4 | com o telescópio | *V Det N* | Reduz |
| 5 | com o telescópio | *VSN* | Reduz |
| 6 | com o telescópio | *SV* | Desloca |
| 7 | o telescópio | *SV P* | Desloca |
| 8 | telescópio | *SV P Det* | Desloca |
| 9 | | *SV P Det N* | Reduz |
| 10 | | *SV P SN* | Reduz |
| 11 | | *SV SP* | Reduz |
| 12 | | *SV* | Ok |

| | **Expressão** | **Pilha** | **Operação** |
|---|---|---|---|
| 5’ | com o telescópio | $VSN$ | Desloca |
| 6’ | o telescópio | $V\ SN\ P$ | Desloca |
| 7’ | telescópio | $V\ SN\ P\ Det$ | Desloca |
| 8’ | | $V\ SN\ P\ Det\ N$ | Reduz |
| 9’ | | $V\ SN\ P\ SN$ | Reduz |
| 10’ | | $V\ SN\ P\ SN$ | Reduz |
| 11’ | | $V\ SN\ SP$ | Reduz |
| 12’ | | $V\ SN$ | Reduz |
| 13’ | | $SV$ | Ok |

```
% tabular.pl
% Luiz Arthur Pagani
% arthur@ufpr.br

%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Analisador tabular   %
%     baseado no de      %
% Matthews 1998: 239-242 %
%%%%%%%%%%%%%%%%%%%%%%%%%%

:- dynamic arco/5.

:- unknown(_, fail).

analisa(Expressão) :-
   abolish(arco/5),
   faz_tabela(1, Vn, Expressão),
   arco(1, Vn, Categoria, _, []),
```

```
    nl,
    write('-- Categoria = '),
    write(Categoria),
    fail.

faz_tabela(V, V, []).
faz_tabela(V1, Vn, [Palavra|Palavras]) :-
    V2 is V1 + 1,
    forall(palavra(Categoria, Palavra),
           faz_arco(V1, V2, Categoria, [], [])),
    faz_tabela(V2, Vn, Palavras).

faz_arco(V1, V2, Categoria, Achadas, Procuradas) :-
    arco(V1, V2, Categoria, Achadas, Procuradas), !.
faz_arco(V1, V2, Cat, Achadas, []) :-
    asserta(arco(V1, V2, Cat, Achadas, [])),
    write(arco(V1, V2, Cat, Achadas, [])), nl,
    forall(regra(Cat1, [Cat|Cats]),
```

```
        faz_arco(V1, V1, Cat1, [], [Cat|Cats])),
   forall(arco(V0, V1, Cat1, Achadas1, [Cat|RestoCats]),
        faz_arco(V0, V2, Cat1, [Cat|Achadas1], RestoCats)).
faz_arco(V0, V1, Cat, Achadas, [Cat1|RestoCats]) :-
   asserta(arco(V0, V1, Cat, Achadas, [Cat1|RestoCats])),
   write(arco(V0, V1, Cat, Achadas, [Cat1|RestoCats])), nl,
   forall(arco(V1, V2, Cat1, _, []),
        faz_arco(V0, V2, Cat, [Cat1|Achadas], RestoCats)).

% Carrega gramática (regras e léxico)
:- [gram].
```

```
?- analisa([o,homem,viu,a,mulher,com,o,telescópio]).
...
arco(1, 9, s: ((Pos@ (Def@T))@ (V@ (Def@M)))@ (Def@H), [sv: (Pos@ (Def@T))@ (V@ (Def@M)), sn:Def@H], [])
arco(4, 9, sn: (Pos@ (Def@T))@ (Def@M), [sp:Pos@ (Def@T), sn:Def@M], [])
arco(4, 4, s:_G690@ ((Pos@ (Def@T))@ (Def@M)), [], [sn: (Pos@ (Def@T))@ (Def@M), sv:_G690])
arco(4, 9, s:_G690@ ((Pos@ (Def@T))@ (Def@M)), [sn: (Pos@ (Def@T))@ (Def@M)], [sv:_G690])
arco(4, 4, sn:_G690@ ((Pos@ (Def@T))@ (Def@M)), [], [sn: (Pos@ (Def@T))@ (Def@M), sp:_G690])
arco(4, 9, sn:_G690@ ((Pos@ (Def@T))@ (Def@M)), [sn: (Pos@ (Def@T))@ (Def@M)], [sp:_G690])
arco(3, 9, sv:V@ ((Pos@ (Def@T))@ (Def@M)), [sn: (Pos@ (Def@T))@ (Def@M), v:V], [])
arco(3, 3, sv:_G759@ (V@ ((Pos@ (Def@T))@ (Def@M))), [], [sv:V@ ((Pos@ (Def@T))@ (Def@M)), sp:_G759])
arco(3, 9, sv:_G759@ (V@ ((Pos@ (Def@T))@ (Def@M))), [sv:V@ ((Pos@ (Def@T))@ (Def@M))], [sp:_G759])
arco(1, 9, s: (V@ ((Pos@ (Def@T))@ (Def@M)))@ (Def@H), [sv:V@ ((Pos@ (Def@T))@ (Def@M)), sn:Def@H], [])

-- Categoria = s: (V@ ((Pos@ (Def@T))@ (Def@M)))@ (Def@H)
-- Categoria = s: ((Pos@ (Def@T))@ (V@ (Def@M)))@ (Def@H)
false.
```

# 6 Algoritmo de Earley

```
regra(det, []).

?- analisa([homem,viu,mulher,com,telescópio]).
arco(1, 2, n:H, [], [])
arco(2, 3, v:V, [], [])
arco(2, 2, sv:V@_G373, [], [v:V, sn:_G373])
arco(2, 3, sv:V@_G373, [v:V], [sn:_G373])
arco(3, 4, n:M, [], [])
arco(4, 5, p:Pos, [], [])
arco(4, 4, sp:Pos@_G403, [], [p:Pos, sn:_G403])
arco(4, 5, sp:Pos@_G403, [p:Pos], [sn:_G403])
arco(5, 6, n:T, [], [])
false.

?-
```

```
% earley.pl
% ------------------
% Luiz Arthur Pagani
% arthur@ufpr.br

%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Algoritmo de Earley   %
%      baseado no de      %
% Covington 1994: 178-183 %
%%%%%%%%%%%%%%%%%%%%%%%%%%

:- dynamic aresta/4.

:- unknown(_,fail).

analisa(Exp) :-
   abolish(aresta/4),
   writeln('Começa com:'),
```

```
    armazena(aresta(início, [Cat], Exp, Exp)),
    processa(Exp),
    aresta(Cat, [], Exp, []),
    Cat \= início, nl,
    write('-- Categoria = '),
    write(Cat),
    fail.

processa([]) :- !.
processa(Exp) :-
    nl, writeln('Previsão:'),
    previsão(Exp),
    nl, writeln('Varredura:'),
    varredura(Exp, RestExp),
    nl, writeln('Completamento:'),
    completamento(RestExp),
    processa(RestExp).
```

```
previsão(Exp) :-
   aresta(_, [Cat | _], _, Exp),
   prevê(Cat, Exp),
   fail.
previsão(_).

prevê(CatLER, Exp) :-
   regra(CatLER, [CatLDR | CatsLDR]),
   armazena(aresta(CatLER, [CatLDR | CatsLDR], Exp, Exp)),
   prevê(CatLDR, Exp),
   fail.
prevê(Cat, Exp) :-
   regra(Cat, []),
   armazena(aresta(Cat, [], Exp, Exp)),
   completa(Cat, Exp, Exp),
   fail.
prevê(_, _).
```

```
varredura([Pal | Pals], Pals) :-
   aresta(CatMeta, [Cat | Cats], Exp, [Pal | Pals]),
   CatMeta \== início,
   palavra(Cat, Pal),
   armazena(aresta(CatMeta, Cats, Exp, Pals)),
   fail.
varredura([_ | Palavras], Palavras).

completamento(RestExp) :-
   aresta(Cat, [], Exp, RestExp),
   completa(Cat, Exp, RestExp),
   fail.
completamento(_).

completa(Cat, Exp, RestExp) :-
   aresta(CatMeta, [Cat | Cats], OutraExp, Exp),
   CatMeta \== início,
   armazena(aresta(CatMeta, Cats, OutraExp, RestExp)),
```

```
    Cats == [],
    completa(CatMeta, OutraExp, RestExp),
    fail.
completa(_, _, _).

armazena(Aresta) :-
    Aresta =.. [aresta, Cat, Cats, Exp, RestExp],
    \+ (aresta(CatHiper, CatsHiper, Exp, RestExp),
        subsumes_chk(CatHiper, Cat),
        subsumes_chk(CatsHiper, Cats)),
    asserta(Aresta),
    write(' - '),
    writeln(Aresta).

% Carrega gramática (regras e léxico)
:- [gram].
```

```
?- analisa([homem,viu,mulher,com,telescópio]).
Começa com:
 - aresta(início, [_G311], [homem, viu, mulher, com, telescópio], [homem, viu, mulher, com, telescópio])

Previsão:
 - aresta(s:_G353@_G354, [sn:_G354, sv:_G353], [homem, viu, mulher, com, telescópio],
                                               [homem, viu, mulher, com, telescópio])
 - aresta(sn:_G391@_G392, [det:_G391, n:_G392], [homem, viu, mulher, com, telescópio],
                                                [homem, viu, mulher, com, telescópio])
 - aresta(det:Indef, [], [homem, viu, mulher, com, telescópio], [homem, viu, mulher, com, telescópio])
 - aresta(sn:Indef@_G453, [n:_G453], [homem, viu, mulher, com, telescópio], [homem, viu, mulher, com, telescópio])
 - aresta(início, [], [homem, viu, mulher, com, telescópio], [homem, viu, mulher, com, telescópio])
 - aresta(sn:_G391@_G392, [sn:_G392, sp:_G391], [homem, viu, mulher, com, telescópio],
                                                [homem, viu, mulher, com, telescópio])
 - aresta(sv:_G353@_G354, [v:_G353, sn:_G354], [homem, viu, mulher, com, telescópio],
                                               [homem, viu, mulher, com, telescópio])
 - aresta(sv:_G353@_G354, [sv:_G354, sp:_G353], [homem, viu, mulher, com, telescópio],
                                                [homem, viu, mulher, com, telescópio])
 - aresta(sp:_G353@_G354, [p:_G353, sn:_G354], [homem, viu, mulher, com, telescópio],
                                               [homem, viu, mulher, com, telescópio])

Varredura:
 - aresta(sn:Indef@H, [], [homem, viu, mulher, com, telescópio], [viu, mulher, com, telescópio])
 - aresta(início, [], [homem, viu, mulher, com, telescópio], [viu, mulher, com, telescópio])

Completamento:
 - aresta(sn:_G356@ (Indef@H), [sp:_G356], [homem, viu, mulher, com, telescópio], [viu, mulher, com, telescópio])
 - aresta(s:_G356@ (Indef@H), [sv:_G356], [homem, viu, mulher, com, telescópio], [viu, mulher, com, telescópio])

Previsão:
 - aresta(sv:_G362@_G363, [v:_G362, sn:_G363], [viu, mulher, com, telescópio], [viu, mulher, com, telescópio])
 - aresta(sv:_G362@_G363, [sv:_G363, sp:_G362], [viu, mulher, com, telescópio], [viu, mulher, com, telescópio])
 - aresta(sp:_G362@_G363, [p:_G362, sn:_G363], [viu, mulher, com, telescópio], [viu, mulher, com, telescópio])
```

```
Varredura:
 - aresta(sv:V@_G339, [sn:_G339], [viu, mulher, com, telescópio], [mulher, com, telescópio])

Completamento:

Previsão:
 - aresta(sn:_G356@_G357, [det:_G356, n:_G357], [mulher, com, telescópio], [mulher, com, telescópio])
 - aresta(det:Indef, [], [mulher, com, telescópio], [mulher, com, telescópio])
 - aresta(sn:Indef@_G418, [n:_G418], [mulher, com, telescópio], [mulher, com, telescópio])
 - aresta(sn:_G356@_G357, [sn:_G357, sp:_G356], [mulher, com, telescópio], [mulher, com, telescópio])

Varredura:
 - aresta(sn:Indef@M, [], [mulher, com, telescópio], [com, telescópio])

Completamento:
 - aresta(sn:_G350@ (Indef@M), [sp:_G350], [mulher, com, telescópio], [com, telescópio])
 - aresta(sv:V@ (Indef@M), [], [viu, mulher, com, telescópio], [com, telescópio])
 - aresta(sv:_G391@ (V@ (Indef@M)), [sp:_G391], [viu, mulher, com, telescópio], [com, telescópio])
 - aresta(s: (V@ (Indef@M))@ (Indef@H), [], [homem, viu, mulher, com, telescópio], [com, telescópio])
 - aresta(início, [], [homem, viu, mulher, com, telescópio], [com, telescópio])

Previsão:
 - aresta(sp:_G362@_G363, [p:_G362, sn:_G363], [com, telescópio], [com, telescópio])

Varredura:
 - aresta(sp:Pos@_G339, [sn:_G339], [com, telescópio], [telescópio])

Completamento:

Previsão:
 - aresta(sn:_G350@_G351, [det:_G350, n:_G351], [telescópio], [telescópio])
 - aresta(det:Indef, [], [telescópio], [telescópio])
```

```
 - aresta(sn:Indef@_G412, [n:_G412], [telescópio], [telescópio])
 - aresta(sn:_G350@_G351, [sn:_G351, sp:_G350], [telescópio], [telescópio])

Varredura:
 - aresta(sn:Indef@T, [], [telescópio], [])

Completamento:
 - aresta(sn:_G344@ (Indef@T), [sp:_G344], [telescópio], [])
 - aresta(sp:Pos@ (Indef@T), [], [com, telescópio], [])
 - aresta(sv: (Pos@ (Indef@T))@ (V@ (Indef@M)), [], [viu, mulher, com, telescópio], [])
 - aresta(sv:_G426@ ((Pos@ (Indef@T))@ (V@ (Indef@M))), [sp:_G426], [viu, mulher, com, telescópio], [])
 - aresta(s: ((Pos@ (Indef@T))@ (V@ (Indef@M)))@ (Indef@H), [], [homem, viu, mulher, com, telescópio], [])
 - aresta(início, [], [homem, viu, mulher, com, telescópio], [])
 - aresta(sn: (Pos@ (Indef@T))@ (Indef@M), [], [mulher, com, telescópio], [])
 - aresta(sn:_G420@ ((Pos@ (Indef@T))@ (Indef@M)), [sp:_G420], [mulher, com, telescópio], [])
 - aresta(sv:V@ ((Pos@ (Indef@T))@ (Indef@M)), [], [viu, mulher, com, telescópio], [])
 - aresta(sv:_G461@ (V@ ((Pos@ (Indef@T))@ (Indef@M))), [sp:_G461], [viu, mulher, com, telescópio], [])
 - aresta(s: (V@ ((Pos@ (Indef@T))@ (Indef@M)))@ (Indef@H), [], [homem, viu, mulher, com, telescópio], [])

-- Categoria = s: (V@ ((Pos@ (Indef@T))@ (Indef@M)))@ (Indef@H)
-- Categoria = s: ((Pos@ (Indef@T))@ (V@ (Indef@M)))@ (Indef@H)
false.
```

# 7 Modelamento

- Alegação de que deslocamento e redução simula o PLH (princípios de preferência)
  - deslocamento × redução: prioridade do deslocamento – associação à direita
  - redução × redução: prioridade para a regra com mais constituintes do lado direito – anexação mínima
- Tamanho da memória de trabalho (*spam*):
  - maior: tabular
  - menor: não tabular

# 8 Conclusão

- Compromisso entre gramática e analisador gramatical
- Ascendente × descendente
  - há línguas que são processadas ascendentemente e outras descendentemente?
  - ou, numa mesma língua, temos a opção de escolher uma estratégia ascendente ou descendente?
- Pilha × fila
  - há línguas que são processadas em pilha e outras em fila?
  - ou, numa mesma língua, temos a opção de escolher uma estratégia em pilha ou em fila?

- Tabular $\times$ não-tabular
  - há línguas que são processadas tabularmente e outras não?
  - ou, numa mesma língua, temos a opção de escolher uma estratégia tabular ou não-tabular?
- Quais as previsões empíricas a que nos levam cada uma das opções para a implementação do analisador?