Processamento de Linguagem Natural

Inteligência Artificial

- Nesta aula são vistos conceitos básicos sobre linguagens formais para Processamento de Linguagem Natural (PLN)
- Dentre as gramáticas, é abordada a notação de Gramática de Cláusulas Definidas (DCG) em Prolog

José Augusto Baranauskas
Departamento de Física e Matemática – FFCLRP-USP
E-mail: augusto@usp.br
URL: http://dfm.fmrp.usp.br/~augusto

Linguagem

- Uma **linguagem formal** é um conjunto de sentenças (cadeias ou frases)
- Cada **sentença** é uma concatenação de símbolos terminais (palavras)
  - Por exemplo, no Cálculo Proposicional, os símbolos terminais incluem $\wedge$, p e q e uma sentença típica é $p \wedge q$
  - A sentença $p\ q \wedge$ não é um elemento da linguagem
- Linguagens formais, tais como Cálculos Proposicional e Relacional, Pascal, C++ e Java têm definições matemáticas rigorosas
- Linguagens naturais, tais como português, chinês e inglês não são definidas rigorosamente, mas utilizadas por uma comunidade de pessoas
- Nesta apresentação trataremos as linguagens naturais como se fossem linguagens formais, embora a correspondência não seja perfeita

Gramáticas

- Uma gramática é um mecanismo formal usado para definir conjuntos de seqüências de símbolos, utilizando **regras de produção** que especificam uma linguagem
- A maioria das regras gramaticais baseia-se no princípio de **estrutura sintagmática** de que as sentenças são compostas por sub-sentenças denominadas **sintagmas**
  - "a menina", "o artista famoso", "o agente no canto direito da sala" são exemplos de **sintagma nominal** (**sn**)
  - "está atenta", "pinta", "está dançando" são exemplos de **sintagma verbal** (**sv**)
- Os sintagmas correspondem a elementos semânticos a partir dos quais o significado de uma expressão pode ser construído, por exemplo, os sintagmas nominais se referem a objetos no mundo
- A divisão dos sintagmas em categorias também auxilia a descrever as sentenças permitidas na linguagem
  - Podemos dizer que qualquer **sn** pode se combinar com um **sv** para formar um sintagma sentencial (**s**)
  - Sem a noção de sintagmas é difícil explicar por que "a menina está atenta" é uma sentença enquanto "menina a viva está" não é

Gramáticas

- Em uma gramática, os sintagmas como **sn**, **sv** e **s** são símbolos **não-terminais** (que dividem as sub-frases da linguagem)
- As **regras de produção** (ou **regras de reescrita**) definem os símbolos não-terminais, utilizando uma notação, normalmente a BNF (Backus-Naur Form)
- Na notação BNF os símbolos não-terminais são colocados entre <>
- Por exemplo, a notação BNF para uma sentença **s** que consiste de qualquer sintagma nominal **sn** seguido por qualquer sintagma verbal **sv** é:
  - <s> ::= <sn> <sv>

Gramáticas

- Formalmente, uma gramática G(S,T,N,R) é definida por quatro componentes:
  - S: símbolo inicial da gramática (S $\in$ N)
  - T: conjunto de símbolos terminais, também denominado **léxico** (ou conjunto de palavras da linguagem)
  - N: conjunto de símbolos não-terminais
  - R: conjunto de regras de produção
- Símbolos terminais nunca podem ser substituídos
- O conjunto de seqüências reconhecidas ou geradas pela gramática G é a **linguagem** definida pela gramática G, denotada por **L(G)**

Gramáticas: Exemplos

- G(s, {a,alerta,está,menina}, {s,sn,sv,art,subst,vl, adj}, R)
  - R = {
    - <s> ::= <sn> <sv>
    - <sn> ::= <art> <subst>
    - <sv> ::= <vl> <adj>
    - <art> ::= a
    - <subst> ::= menina
    - <adj> ::= alerta
    - <vl> ::= está

    }
- G(s, {a,b}, {s}, R)
  - R = {
    - <s> ::= a b
    - <s> ::= a <s> b

    }

Gramáticas

- Na gramática G(s,{a,b},{s},R) onde R é composto pelas regras:

  <s> ::= a b

  <s> ::= a <s> b

  - A primeira regra diz que sempre que o símbolo **s** aparecer, ele pode ser substituído pela seqüência **ab**
  - A segunda regra diz que **s** pode ser substituído pela seqüência: **a** seguido por **s** seguido por **b**
- Em BNF, as duas regras de produção anteriores também podem ser escritas da forma
  - <s> ::= a b | a <s> b

Gramáticas

- Uma gramática G(S,T,N,R) pode ser usada para **gerar** uma sentença
  - A geração sempre começa pelo símbolo não-terminal inicial **S**
  - A seguir, os símbolos não-terminais são substituídos por outras seqüências de acordo com as regras **R** da gramática
  - A geração termina quando a seqüência atual não contém símbolos não-terminais **N**, mas somente símbolos terminais **T**
- Uma gramática também pode ser usada para **reconhecer** (***parsing***) uma dada sentença cujo o processo é basicamente o inverso da geração

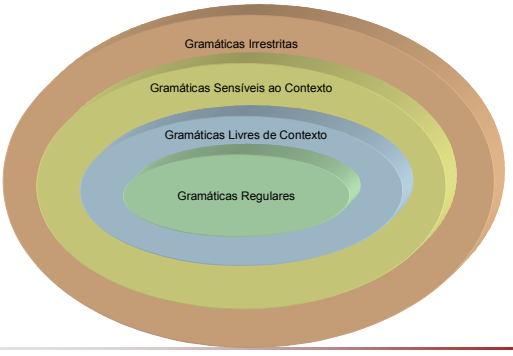
Gramáticas

- Ainda considerando o exemplo G(s,{a,b},{s},R), onde R é composto por:
  - <s> ::= a b
  - <s> ::= a <s> b
- Geração
  - Início: <s>
  - Aplicando a segunda regra <s> é trocado por a<s>b
  - Aplicando a segunda regra novamente a<s>b é trocado por aa<s>bb
  - Aplicando a primeira regra em aa<s>bb, obtém-se a sentença aaabbb
  - Neste caso, gerou-se a sentença aaabbb
- Reconhecimento
  - Assuma que deseja-se saber se a sentença aabb pertence à L(G)
  - Início: aabb
  - Aplicando a primeira regra aabb é trocado por a<s>b
  - Aplicando a segunda regra a<s>b é trocado por <s>
  - Como atingiu-se o símbolo inicial da gramática a sentença foi reconhecida como pertencente à L(G)
- Nesta gramática L(G) = $\{a^n b^n\}$, ou seja, uma seqüência de **n** cópias do símbolo **a** seguido por **n** cópias do símbolo **b**

Capacidade Gerativa

- As gramáticas podem ser classificadas por sua capacidade gerativa, ou seja, o conjunto de linguagens que elas podem gerar
- Chomsky (1957) descreve quatro classes de gramáticas que diferem apenas na forma das regras de produção e são organizadas hierarquicamente em poder de representação, listadas a seguir com a classe mais poderosa em primeiro lugar:
  - Tipo-0: Gramáticas Irrestritas (ou Recursivamente Enumeráveis)
  - Tipo-1: Gramáticas Sensíveis ao Contexto
  - Tipo-2: Gramáticas Livres de Contexto
  - Tipo-3: Gramáticas Regulares

Capacidade Gerativa



Capacidade Gerativa

- As **Gramáticas Irrestritas** incluem todas as linguagens formais
- Estas gramáticas utilizam regras irrestritas: ambos os lados das regras de reescrita podem ter qualquer número de símbolos terminais e não terminais; por exemplo, a regra
  - <a> w <b> <c> ::= <p> x <q>
- Essas gramáticas são equivalentes a máquinas de Turing em poder de representação

Capacidade Gerativa

- As **Gramáticas Sensíveis ao Contexto** são restritas apenas no fato que o lado direito deve conter pelo menos tantos símbolos quanto o lado esquerdo
- A denominação "sensível ao contexto" resulta do fato que em uma regra como
  - <a> <s> <b> ::= <a> <y> <b>
  - diz que <s> pode ser reescrito como <y> no *contexto* de um <a> precedente e um <b> seguinte
- Essas gramáticas podem representar linguagens como L(G) = $\{a^n b^n c^n\}$

Capacidade Gerativa

- Nas **Gramáticas Livres de Contexto** o lado esquerdo consiste em um único não-terminal
- Por exemplo
  - <s> ::= w <a> <y>
  - <s> ::= <a> <b> w z
  - <s> ::= <a> w x <b> z
- Dessa forma, cada regra define a reescrita de um não-terminal em qualquer contexto
- Essas gramáticas podem representar linguagens como L(G) = $\{a^n b^n\}$ mas não L(G) = $\{a^n b^n c^n\}$

Capacidade Gerativa

- Nas **Gramáticas Regulares** em toda regra:
  - O lado esquerdo tem um único símbolo não terminal
  - O lado direito tem um único símbolo terminal, opcionalmente seguido (gramática regular à direita) ou precedido (gramática regular à esquerda) por um único símbolo não-terminal
- Exemplos
  - Gramática regular à direita
    - <s> ::= w
    - <s> ::= w <a>
  - Gramática regular à esquerda
    - <s> ::= w
    - <s> ::= <a> w
- Uma gramática é regular se for uma gramática regular à direita ou à esquerda (mas não ambas)
- Essas gramáticas podem representar linguagens como L(G) = $\{a^* b^*\}$, ou seja, uma seqüência de qualquer número de símbolos **a** seguida por qualquer número de símbolos **b**

Capacidade Gerativa

- As gramáticas posicionadas em ordem mais alta na hierarquia têm maior poder de representação, mas os algoritmos que as manipulam são menos eficientes
- Até meados da década de 1980, os lingüistas se concentravam em linguagens livres de contexto e sensíveis ao contexto
- Com a necessidade de processar gigabytes de texto on-line, houve uma ênfase crescente nas gramáticas regulares

Notação DCG

- DCG (*Definite Clause Grammar*) ou Gramática de Cláusulas Definidas é uma notação especial de regras gramaticais, similar à BNF, suportada por muitas implementações Prolog
- O mapeamento BNF para DCG é dado por:
  - O símbolo **::=** é substituído por **-->**
  - Não terminais não ficam mais entre **<>**
  - Terminais ficam entre **[ ]**, tornando-os listas Prolog
  - Símbolos da gramática são separados por vírgulas e cada regra é terminada com um ponto final como em toda cláusula Prolog
- Por exemplo
  - Em BNF:
    - <s> ::= a <s> b
  - Em DCG:
    - s --> [a], s, [b].

Notação DCG

- Em Prolog, o reconhecimento e a geração esperam que as sentenças sejam representadas como **diferença de listas** de símbolos terminais
  - Cada sentença é representada por duas listas: a sentença representada consiste na diferença entre ambas as listas
- As duas listas não são únicas, por exemplo, a sentença **aabb** pode ser representada
  - pelas listas **[a,a,b,b]** e **[ ]**
  - ou pelas listas **[a,a,b,b,c]** e **[c]**
  - ou pelas listas **[a,a,b,b,1,0,1]** e **[1,0,1]**
  - ...

Notação DCG

- Portanto, em Prolog, cada símbolo não-terminal recebe dois parâmetros:
  - O primeiro é uma lista com a sentença a ser reconhecida ou gerada
  - O segundo é a lista que resta após a análise do primeiro parâmetro
    - Como sempre queremos que a sentença seja reconhecida, o segundo parâmetro é, geralmente, a lista vazia
- Considerando esta representação de sentenças, a gramática pode ser questionada a reconhecer algumas sentenças:
  - `?- s([a,a,b,b], []).` % reconhece sentença *aabb*
  - `?- s([a,a,b,b,c], [c]).` % idem
  - `?- s([a,a,b,b,1,0,k], [1,0,k]).` % idem

Notação DCG: Mapeamento Prolog

- Em DCG:
  - n --> n1, n2, ..., nn.
  - onde n_i são não-terminais
- Em Prolog:
  - n(Lista1, Resto) :-
    n1(Lista1, Lista2),
    n2(Lista2, Lista3),
    ...
    nn(ListaN, Resto).

Notação DCG: Mapeamento Prolog

- Em DCG:
  - n --> n1, [t2], n3, [t4].
  - onde n_i são não-terminais e t_i são terminais
- Em Prolog:
  - n(Lista1, Resto) :-
    n1(Lista1, [t2 | Lista3]),
    n3(Lista3, [t4 | Resto]).

Exemplo 1

- Gramática simples (em DCG) para PLN:
  - sentenca --> sujeito, predicado.
  - sujeito --> artigo, substantivo.
  - predicado --> verbo, artigo, substantivo.
  - artigo --> [o].
  - substantivo --> [gato].
  - substantivo --> [rato].
  - verbo --> [assustou].
- Sentenças geradas pela gramática:

```
?- sentenca(S,[]).
S = [o, gato, assustou, o, gato] ;
S = [o, gato, assustou, o, rato] ;
S = [o, rato, assustou, o, gato] ;
S = [o, rato, assustou, o, rato]
```


Exemplo 1

- Gramática simples (em DCG) para PLN:
  - sentenca --> sujeito, predicado.
  - sujeito --> artigo, substantivo.
  - predicado --> verbo, artigo, substantivo.
  - artigo --> [o].
  - substantivo --> [gato].
  - substantivo --> [rato].
  - verbo --> [assustou].
- Representação Prolog:
  - sentenca(A, B) :-
    sujeito(A, C),
    predicado(C, B).
  - sujeito(A, B) :-
    artigo(A, C),
    substantivo(C, B).
  - predicado(A, B) :-
    verbo(A, C),
    artigo(C, D),
    substantivo(D, B).
  - artigo([o|A], A).
  - substantivo([gato|A], A).
  - substantivo([rato|A], A).
  - verbo([assustou|A], A).

Exemplo 2

- Expandindo a gramática com as formas no plural:
  - sentenca --> sujeito, predicado.
  - sujeito --> artigo, substantivo.
  - predicado --> verbo, artigo, substantivo.
  - artigo --> [o].
  - artigo --> [os].
  - substantivo --> [gato].
  - substantivo --> [gatos].
  - substantivo --> [rato].
  - substantivo --> [ratos].
  - verbo --> [assustou].
  - verbo --> [assustaram].
- Algumas sentenças geradas:

```
?- sentenca(S,[]).
S = [os, gatos, assustaram, o, gato] ;    % OK
S = [o, ratos, assustou, os, gato] ;      % ERRO
S = [o, rato, assustaram, o, ratos] ;     % ERRO
...
```

Dependência de Contexto

- No Exemplo 2, o verbo tem que concordar em número com o artigo e o substantivo do sujeito (que também devem concordar em gênero)
- Este fenômeno é chamado **dependência de contexto**
- Isso pode ser resolvido, acrescentando parâmetros extras na gramática DCG

Exemplo 3

- Expandindo a gramática com dependência de contexto (G: gênero; N: número):
  - sentenca(G,N) --> sujeito(G,N), predicado(N).
  - sujeito(G,N) --> artigo(G,N), substantivo(G,N).
  - predicado(N) --> verbo(N), artigo(G1, N1), substantivo(G1,N1).
  - artigo(masc, sing) --> [o].
  - artigo(masc, plural) --> [os].
  - substantivo(masc, sing) --> [gato].
  - substantivo(masc, plural) --> [ratos].
  - verbo(sing) --> [assustou].
  - verbo(plural) --> [assustaram].
- Sentenças geradas

```
?- sentenca(G,N,S,[]).
G = masc N = sing S = [o, gato, assustou, o, gato] ;
G = masc N = sing S = [o, gato, assustou, os, ratos] ;
G = masc N = plural S = [os, ratos, assustaram, o, gato] ;
G = masc N = plural S = [os, ratos, assustaram, os, ratos]
```


Exemplo 3

```
?- sentenca(masc,sing,[o,gato,assustou,os,ratos],[]).
yes

?- sentenca(G,plural,[os,gatos,assustou,os,ratos],[]).
no

?- sentenca(G,N,[o,gato,assustou,os,ratos],[]).
G = masc N = sing

?- sentenca(masc,sing,[o,Que,assustou,os,ratos],[]).
Que = gato
```


Dependência de Contexto

- Quando as regras são convertidas para Prolog, os argumentos extras dos não-terminais são simplesmente adicionados (no início) aos argumentos da cláusula Prolog, com a convenção que as duas listas (sentença a ser analisada e o que sobra) aparecem no final
- Por exemplo
  - sentenca(G,N) --> sujeito(G,N), predicado(N).

  é convertida em:
  - sentenca(G, N, Lista1, Resto) :-
    sujeito(G, N, Lista1, Lista2),
    predicado(N, Lista2, Resto).

Exercício

- Dada a seguinte gramática em BNF

```
<sentenca>     ::= <sintagma nominal> <sintagma verbal>
<sintagma nominal> ::= <artigo> <substantivo> |
                       <artigo> <substantivo> <qualificador>
<sintagma verbal>  ::= <verbo> |
                       <verbo> <sintagma nominal>
<qualificador> ::= preto|grande|preta|pretos|pretas
<substantivo>  ::= gato|cachorro|gata|cachorra|
                   gatos|cachorros
<verbo>        ::= corre|correm|assusta|assustam
<artigo>       ::= o|a|os|as
```

- Escreva um programa (na notação DCG) para reconhecer sentenças desta linguagem, com concordância de gênero e número
- Quais são as sentenças geradas por esta gramática?

Solução sem Gênero/Número

```
s --> sn, sv.

sn --> artigo, subst.
sn --> artigo, subst, q.

sv --> verbo.
sv --> verbo, sn.

q --> [preto].
q --> [pretos].
q --> [grande].
q --> [preta].
q --> [pretas].

subst --> [gato].
subst --> [gatos].
subst --> [cachorro].
subst --> [cachorros].
subst --> [gata].
subst --> [cachorra].

verbo --> [corre].
verbo --> [correm].
verbo --> [assusta].
verbo --> [assustam].

artigo --> [o].
artigo --> [a].
artigo --> [os].
artigo --> [as].
```

Solução com Gênero/Número

```
s(G,N) --> sn(G,N), sv(N).

sn(G,N) --> artigo(G,N),
   subst(G,N).
sn(G,N) --> artigo(G,N),
   subst(G,N), q(G,N).

sv(N) --> verbo(N).
sv(N) --> verbo(N), sn(G1,N1).

q(masc,sing) --> [preto].
q(masc,plural) --> [pretos].
q(masc,sing) --> [grande].
q(fem,sing) --> [grande].
q(fem,sing) --> [preta].
q(fem,plural) --> [pretas].
```

```
subst(masc,sing) --> [gato].
subst(masc,plural) --> [gatos].
subst(masc,sing) --> [cachorro].
subst(masc,plural) -->
   [cachorros].
subst(fem,sing) --> [gata].
subst(fem,sing) --> [cachorra].

verbo(sing) --> [corre].
verbo(plural) --> [correm].
verbo(sing) --> [assusta].
verbo(plural) --> [assustam].

artigo(masc,sing) --> [o].
artigo(fem,sing) --> [a].
artigo(masc,plural) --> [os].
artigo(fem,plural) --> [as].
```


Exercício

- Considere a gramática

```
<sentenca>     ::= <sintagma nominal> <sintagma verbal>
<sintagma nominal> ::= <artigo> <substantivo>
<sintagma verbal>  ::= <verbo> <sintagma nominal>
<substantivo>   ::= menino|meninos|menina|meninas
<verbo>  ::= beijou|beijaram
<artigo> ::= o|a|os|as
```

- Estenda essa codificação de forma que apenas sejam geradas/reconhecidas sentenças corretas quanto à
  - (a) gramática da Língua Portuguesa
  - (b) etiqueta brasileira (sem beijos entre indivíduos do sexo masculino)

Solução

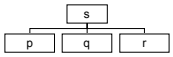
```
s(fem,N) --> sn(fem,N), sv(G1,N).
s(masc,N) --> sn(masc,N), sv(fem,N).

sn(G,N) --> artigo(G,N), subst(G,N).
sv(G,N) --> verbo(N), sn(G,N1).

subst(masc,sing) --> [menino].
subst(masc,plural) --> [meninos].
subst(fem,sing) --> [menina].
subst(fem,plural) --> [meninas].

verbo(sing) --> [beijou].
verbo(plural) --> [beijaram].

artigo(masc,sing) --> [o].
artigo(fem,sing) --> [a].
artigo(masc,plural) --> [os].
artigo(fem,plural) --> [as].
```


Interpretação Semântica

- A interpretação semântica é responsável em extrair o significado das sentenças de uma linguagem
- Há duas formas elementares de realizar a análise semântica:
  - Utilizar Árvores Sintáticas
    - Gerar a árvore sintática da sentença
    - Processá-la para computar o significado (caso as estruturas sintáticas e semânticas sejam similares)
  - Incorporar tratamento semântico diretamente na gramática

Árvores Sintáticas

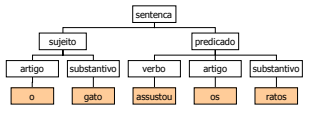
- A **árvore sintática** de uma sentença é uma árvore com as seguintes propriedades
  - Todas as folhas correspondem a símbolos terminais
  - Todos os nós internos correspondem a símbolos não-terminais; a raiz da árvore corresponde ao símbolo não-terminal que corresponde à sentença
  - A relação entre nós pais e filhos é especificada pelas regras da gramática
    - Exemplo: se a gramática contém a regra **s --> p, q, r.** então a árvore pode conter o nó **s** cujos filhos são **p**, **q** e **r** (nesta ordem):



Árvores Sintáticas

- Considerando a sentença [o, gato, assustou, os, ratos] na gramática
  - sentenca --> sujeito, predicado.
  - sujeito --> artigo, substantivo.
  - predicado --> verbo, artigo, substantivo.
  - artigo --> [o].
  - artigo --> [os].
  - substantivo --> [gato].
  - substantivo --> [ratos].
  - verbo --> [assustou].
  - verbo --> [assustaram].
- A árvore sintática resultante é:

```
sentenca(
  sujeito(
    artigo(o),
    substantivo(gato)
          ),
  predicado(
    verbo(assustou),
    artigo(os),
    substantivo(ratos)
          )
        )
```



Árvores Sintáticas

- Uma árvore sintática pode ser representada por um termo Prolog cujo *functor* é a raiz da árvore e seus argumentos são sub-árvores da árvore sintática
  - Por exemplo, a árvore sintática para o **sujeito** 'o gato' pode ser representada como
    - sujeito(artigo(o), substantivo(gato))
- Para gerar a árvore sintática, a gramática DCG pode ser alterada para incluir à cada não-terminal sua árvore sintática correspondente
  - No exemplo, a árvore sintática para o **sujeito** tem a forma
    - sujeito(ÁrvoreArtigo, ÁrvoreSubstantivo)
  - Neste caso **ÁrvoreArtigo** e **ÁrvoreSubstantivo** são árvores sintáticas de um **artigo** e um **substantivo**, respectivamente
  - Adicionando essas árvores sintáticas como argumentos na regra gramatical sobre o **sujeito** resulta na regra modificada
    - sujeito(sujeito(ÁrvoreArtigo,ÁrvoreSubstantivo)) --> artigo(ÁrvoreArtigo), substantivo(ÁrvoreSubstantivo)
    - Esta regra pode ser lida como: Um sujeito cuja árvore sintática é sujeito(ÁrvoreArtigo, ÁrvoreSubstantivo) consiste de
      - um artigo cuja árvore sintática é **ÁrvoreArtigo** e
      - um substantivo cuja árvore sintática é **ÁrvoreSubstantivo**

Exemplo 4

```
sentenca(G,N,sentenca(Sujeito,Predicado)) -->
  sujeito(G,N,Sujeito), predicado(N,Predicado).

sujeito(G,N,sujeito(Artigo,Subst)) -->
  artigo(G,N,Artigo), substantivo(G,N,Subst).

predicado(N,predicado(Verbo,Artigo,Subst)) --> verbo(N,Verbo),
   artigo(G1,N1,Artigo), substantivo(G1,N1,Subst).

artigo(masc,sing,artigo(o)) --> [o].
artigo(masc,plural,artigo(os)) --> [os].

substantivo(masc,sing,substantivo(gato)) --> [gato].
substantivo(masc,plural,substantivo(ratos)) --> [ratos].

verbo(sing,verbo(assustou)) --> [assustou].
verbo(plural,verbo(assustaram)) --> [assustaram].
```


Exemplo 4

```
?- sentenca(G,N,A,S,[]).
G = masc N = sing S = [o, gato, assustou, o, gato]
A = sentenca(sujeito(artigo(o), substantivo(gato)),
    predicado(verbo(assustou), artigo(o), substantivo(gato))) ;

G = masc N = sing S = [o, gato, assustou, os, ratos]
A = sentenca(sujeito(artigo(o), substantivo(gato)),
    predicado(verbo(assustou), artigo(os), substantivo(ratos))) ;

G = masc N = plural S = [os, ratos, assustaram, o, gato]
A = sentenca(sujeito(artigo(os), substantivo(ratos)),
    predicado(verbo(assustaram), artigo(o), substantivo(gato))) ;

G = masc N = plural S = [os, ratos, assustaram, os, ratos]
A = sentenca(sujeito(artigo(os), substantivo(ratos)),
    predicado(verbo(assustaram), artigo(os), substantivo(ratos)))
```


Incorporando Tratamento Semântico Diretamente na DCG

- Como a notação DCG permite incorporar tratamento semântico diretamente na gramática, a construção da árvore sintática é evitada
- Esta extensão permite inserir predicados Prolog nas regras da gramática, os quais devem estar entre **{** e **}**
  - n(X) --> n1, ..., nn, **{**predicado1, ..., predicadoN**}**.
- Exemplos
  - verbo(Sujeito, Objeto) --> [matar], **{**pessoa(Sujeito), odeia(Sujeito,Objeto)**}**.
  - digito(X) --> [X], **{**pertence(X,[0,1,2,3,4,5,6,7,8,9])**}**.
  - subst_proprio(X) --> [X], **{**homem(X); mulher(X)**}**.

Interpretação Semântica

- Assuma, inicialmente, a gramática
  - sentenca --> sn, sv.
  - sn --> subst_proprio.
  - sv --> vi.
  - sv --> vtd, sn.
  - vi --> [pinta].
  - vtd --> [ama].
  - subst_proprio --> [vinicius].
  - subst_proprio --> [vivian].
- A Lógica de Predicados é uma boa candidata para a representação de sentenças em linguagem natural, por exemplo:
  - "Vinícius pinta" é representado como **pinta(vinicius)**
  - "Vinícius ama Vivian" é representado como **ama(vinicius,vivian)**

Interpretação Semântica

- Vamos incorporar significado a estas regras, começando pelas categorias mais simples: substantivos e verbos
- O significado do substantivo próprio **vinícius** é simplesmente **vinícius**
  - subst_proprio(vinicius) --> [vinicius].

Interpretação Semântica

- O significado de um verbo intransitivo, por exemplo o verbo pintar, pode ser denotado por **pinta(Ator)** onde **Ator** é uma variável cujo valor apenas será conhecido a partir do contexto, ou seja, a partir do sintagma nominal
  - vi(Ator,pinta(Ator)) --> [pinta].
- A necessidade do argumento **Ator** é torná-lo visível fora do termo **pinta(Ator)** de forma que ele se torne acessível para instanciação
- Isso requer a redefinição de **sv** e **sentenca** para que **Ator** seja uma argumento extra
  - sv(Ator,SV) --> vi(Ator,SV).
  - sentenca(SV) --> sn(Ator), sv(Ator,SV).
- A redefinição de **sentenca** força o argumento **Ator** do significado do verbo a ser igual ao significado do sintagma nominal

Interpretação Semântica

- Esta técnica de tornar partes do significado visível funciona da seguinte forma:
  - O significado da frase é definido de forma de 'esqueleto' – por exemplo, **pinta(Ator)**
  - Isso define a forma geral do significado mais deixa algumas variáveis não-instanciadas– no exemplo, **Ator**
  - Essas variáveis não-instanciadas servem como **encaixes** que serão preenchidos mais tarde, dependendo do significado de outras frases no contexto
  - O preenchimento dos encaixes utiliza o mecanismo de unificação de Prolog e para facilitar isso os encaixes devem ser tornados visíveis por meio de sua adição como argumentos extras aos símbolos não-terminais
- Adotaremos a seguinte convenção quanto à ordem destes argumentos: primeiro aparecem os encaixes visíveis do significado da frase, seguidos pelo significado, por exemplo **sv(Ator,SignificadoSV)**

Interpretação Semântica

- Esta técnica pode ser aplicada a outros verbos, por exemplo
  - o significado do v.t.d. **ama** é ama(Alguém,Algo)
  - vtd(Alguém,Algo,ama(Alguem,Algo)) --> [ama].
- O sintagma verbal com um v.t.d. contém um sintagma nominal que fornece o valor para **Algo**
  - sv(Alguém,SV) --> vtd(Alguém,Algo,SV), sn(Algo)
- Analogamente para substantivos comuns
  - o significado de **homem** é homem(X)
  - substantivo(X,homem(X)) --> [homem].

Quantificador Existencial

- Considerando a sentença 'Um homem pinta' seria um erro considerar seu significado como 'pinta(homem)'
- A sentença na realidade significa que 'existe algum homem que pinta' representando em Prolog como:
  - existe(X, homem(X) & pinta(X))
- A forma geral é:
  - Existe algum X tal que X tem alguma propriedade (por exemplo, homem(X)) e alguma outra afirmativa sobre X é verdadeira (por exemplo, pinta(X))
  - existe(X, Propriedade & Afirmativa)
- Para facilitar a importação do significado de outras frases no contexto, temos que tornar **Propriedade** e **Afirmativa** visíveis
- Portanto, a regra DCG para o quantificador existencial é:
  - quantificador(X,Prop,Afirm, existe(X,Prop & Afirm)) --> [um].

Quantificador Universal

- O quantificador universal pode ser manipulado de forma similar
- Considerando a sentença 'Toda mulher dança', a interpretação lógica é 'para todo X se X é uma mulher então X dança' representando em Prolog como:
  - para_todo(X, mulher(X) -> dança(X))
- A forma geral é:
  - para_todo(X, Propriedade -> Afirmativa)
- Portanto, a regra DCG para o quantificador universal é:
  - quantificador(X,Prop,Afirm,para_todo(X,Prop -> Afirm)) --> [todo].

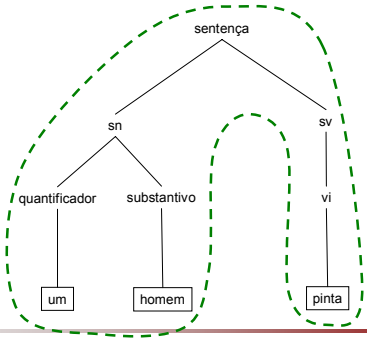
Frases Relativas

- Substantivos podem ser qualificados por frases relativas
  - Todo homem que pinta admira Monet
  - "Todo homem que pinta" é um sintagma nominal ao qual "que pinta" se refere
- Cujo significado é:
  - Para todo X, se X é um homem e X pinta então X admira Monet
  - para_todo(X, (homem(X) & pinta(X)) -> admira(X,monet))
- A forma geral torna-se:
  - para_todo(X, Prop1 & Prop2 -> Afirm)

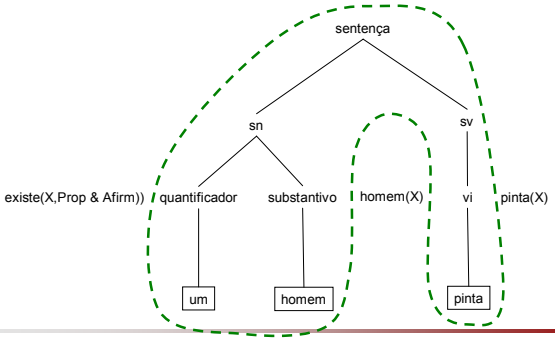
Integração do Significado

- Vamos analisar como o significado dos quantificadores se integra com os significados de outras frases no contexto, levando ao significado final de toda a sentença
- Na sentença 'Um homem pinta', cujo significado é:
  - existe(X, homem(X) & pinta(X))
- Definimos o significado de 'um' como:
  - existe(X, Propriedade & Afirmativa)
- Comparando esses dois significados é imediato perceber que a estrutura do significado da sentença é ditada pelo quantificador

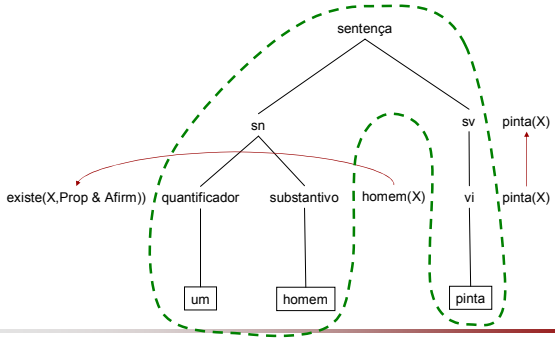
Integração do Significado



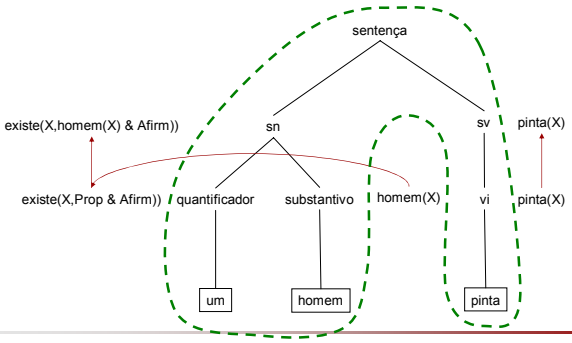
Integração do Significado



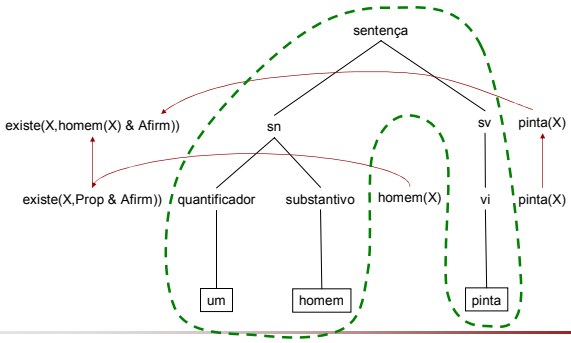
Integração do Significado



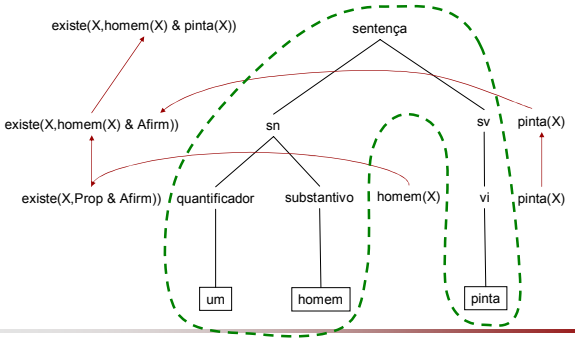
Integração do Significado



Integração do Significado



Integração do Significado



DCG Exemplo

- A gramática seguinte manipula a sintaxe e significado de um pequeno subconjunto de linguagem natural, tais como as seguintes sentenças:
  - Vinicius pinta
  - Um homem pinta
  - Todo homem que pinta admira Monet
  - Vivian admira todo homem que pinta
  - Toda mulher que ama um homem que pinta admira Monet

DCG Exemplo

```
:- op(100,xfy,&).
:- op(150,xfy,->).
sentenca(S) -->  sn(X,P,S), sv(X,P).
sn(X,P,S)-->  quant(X,P1,P,S), substantivo(X,P1).
sn(X,P,S)-->  quant(X,P12,P,S), substantivo(X,P1), relativa(X,P1,P12).
sn(X,P,P)-->  subst_proprio(X).
sv(X,P)  -->  vtd(X,Y,P1), sn(Y,P1,P).
sv(X,P)  -->  vi(X,P).
relativa(X, P1, P1 & P2) --> [que], sv(X,P2).
quant(X,P1,P,para_todo(X,P1->P))--> [todo].
quant(X,P1,P,para_todo(X,P1->P))--> [toda].
quant(X,P1,P, existe(X,P1 & P)) --> [um].
quant(X,P1,P, existe(X,P1 & P)) --> [uma].
substantivo(X, homem(X))  --> [homem].
substantivo(X, mulher(X)) --> [mulher].
subst_proprio(vinicius)   --> [vinicius].
subst_proprio(vivian)     --> [vivian].
subst_proprio(monet)      --> [monet].
vi(X, pinta(X))           --> [pinta].
vtd(X,Y, ama(X,Y))        --> [ama].
vtd(X,Y, admira(X,Y))     --> [admira].
```


DCG Exemplo

```
%% Alguns testes
teste1(Sig,Sent) :-
  Sent = [vinicius,pinta],
  sentenca(Sig,Sent,[]).
teste2(Sig,Sent) :-
  Sent = [um,homem,pinta],
  sentenca(Sig,Sent,[]).
teste3(Sig,Sent) :-
  Sent = [toda,mulher,pinta],
  sentenca(Sig,Sent,[]).
teste4(Sig,Sent) :-
  Sent = [todo,homem,que,pinta,admira,monet],
  sentenca(Sig,Sent,[]).
teste5(Sig,Sent) :-
  Sent = [vivian,admira,todo,homem,que,pinta],
  sentenca(Sig,Sent,[]).
teste6(Sig,Sent) :-
  Sent = [toda,mulher,que,ama,um,homem,que,pinta,admira,monet],
  sentenca(Sig,Sent,[]).
```


Slides baseados nos livros:

Bratko, I.;
Prolog Programming for Artificial Intelligence,
3rd Edition, Pearson Education, 2001.

Russell, S. J. & Norvig, P.;
Inteligência Artificial,
2ª edição, Campus, 2004.

Material elaborado por
José Augusto Baranauskas
Revisão 2005