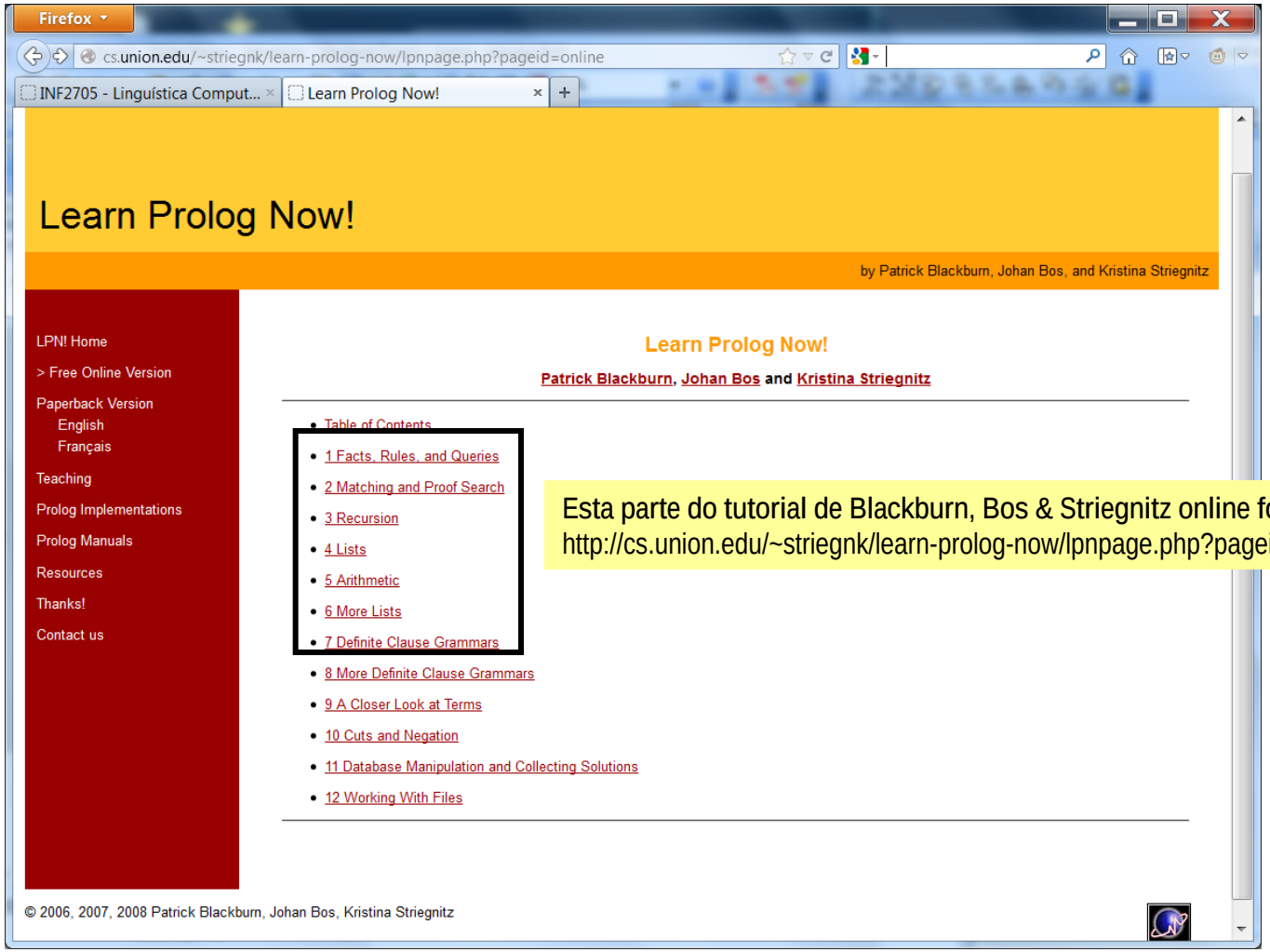
Linguística Computacional Interativa

Redes de Transição Recursivas
RTN's → ATN's (Woods, 1970)
Gramáticas de Cláusulas Definidas
ATN's → DCG's

Aula de 11 de setembro de 2012

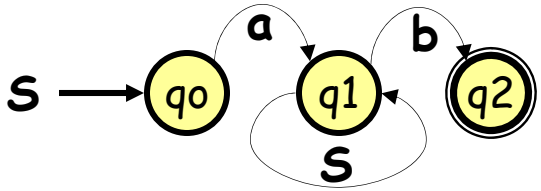
Pressuposto para a aula:

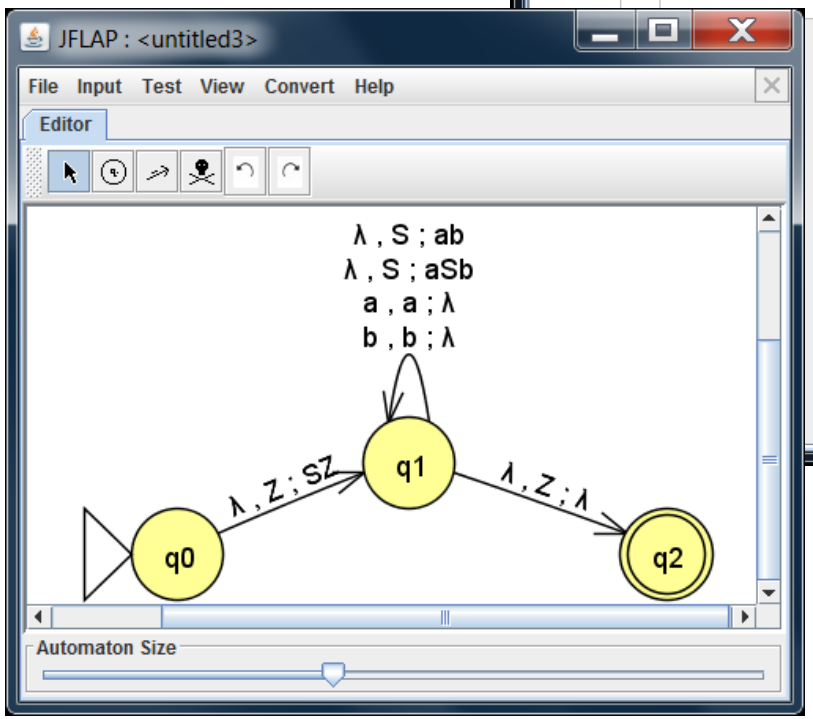


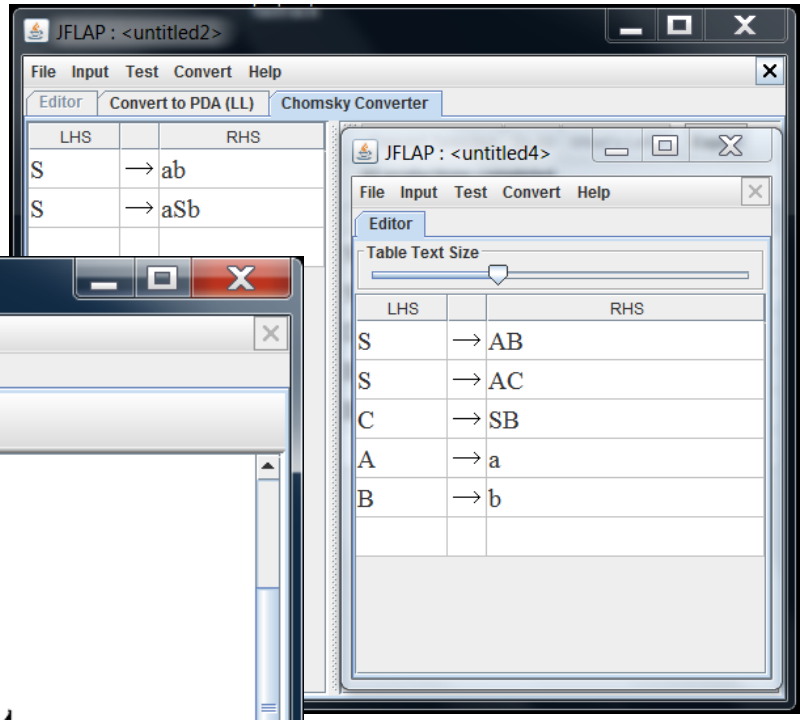
Esta parte do tutorial de Blackburn, Bos & Striegnitz online foi estudada
http://cs.union.edu/~striegnk/learn-prolog-now/lpnpage.php?pageid=online

PUC RIO

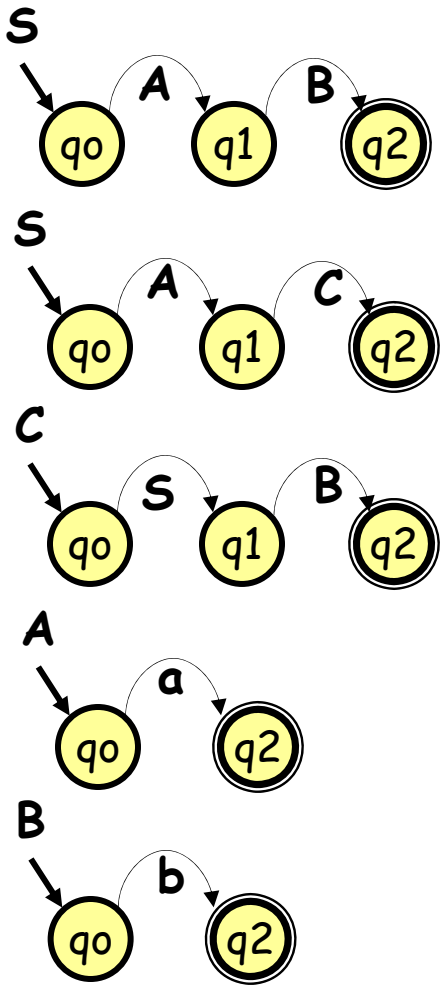
Redes de Transição Recursivas







- RT's cujos elos são rotulados por símbolos que 'chamam' RT's (a mesma, ou outra).



RTN's e PDA's

- As redes de transição recursivas (RTN's) são equivalentes, em poder gerativo / descritivo, a gramáticas livres de contexto (CFG's) ou autômatos de pilha (PDA's).

- Diversas linguagens computacionais podem ser descritas através destes três formalismos.

- Porém, com amplamente discutido em estudos linguísticos, as linguagens humanas *não podem ser devidamente tratadas por gramáticas livres de contexto (e seus equivalentes).*

ATN's – *Augmented Transition Networks*

- Propostas por William Woods em 1970, como um formalismo eficiente para tratar do processamento da linguagem natural.
- Têm poder gerativo /descritivo equivalente às gramáticas transformacionais de Chomsky (à época o formalismo mais completo para o processamento de LN).
- Como têm o poder gerativo/descritivo de uma Máquina de Turing, são adequadas para processar qualquer linguagem computacional.

Especificação formal das ATN's (Woods, 1970: p.593)

⟨transition network⟩ → (⟨arc set⟩⟨arc set⟩*)
⟨arc set⟩ → (⟨state⟩⟨arc⟩*)
⟨arc⟩ → (CAT ⟨category name⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(PUSH ⟨state⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(TST ⟨arbitrary label⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(POP ⟨form⟩⟨test⟩)
⟨action⟩ → (SETR ⟨register⟩⟨form⟩)|
(SENDR ⟨register⟩⟨form⟩)|
(LIFTR ⟨register⟩⟨form⟩)
⟨term act⟩ → (TO ⟨state⟩)|
(JUMP ⟨state⟩)
⟨form⟩ → (GETR ⟨register⟩)|
*|
(GETF ⟨feature⟩)|
(BUILDQ ⟨fragment⟩⟨register⟩*)|
(LIST ⟨form⟩*)|
(APPEND ⟨form⟩⟨form⟩)|
(QUOTE ⟨arbitrary structure⟩)

Especificação formal das ATN's (Woods, 1970: p.593)

⟨transition network⟩ → (⟨arc set⟩⟨arc set⟩*)
⟨arc set⟩ → (⟨state⟩⟨arc⟩*)

Uma ATN é constituída por um ou mais 'arc sets'.

Um 'arc set' é constituído por um estado seguido de 0..n arcos.

Especificação formal das ATN's (Woods, 1970: p.593)

⟨arc⟩ → (CAT ⟨category name⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(PUSH ⟨state⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(TST ⟨arbitrary label⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(POP ⟨form⟩⟨test⟩)

Arcos podem ser de 4 tipos, cada qual com uma estrutura particular de constituintes: arcos CAT; arcos PUSH; arcos TST; e arcos POP. Exceto os arcos POP, todos os demais contêm um TESTE seguido de 0..n AÇÕES e de uma 'terminal action'. O teste dos arcos POP é precedido de um 'form' e não tem nenhuma AÇÃO associada.

Especificação formal das ATN's (Woods, 1970: p.593)

⟨action⟩ → (SETR ⟨register⟩⟨form⟩)|
(SENDR ⟨register⟩⟨form⟩)|
(LIFTR ⟨register⟩⟨form⟩)

⟨term act⟩ → (TO ⟨state⟩)|
(JUMP ⟨state⟩)

⟨form⟩ → (GETR ⟨register⟩)|
*|
(GETF ⟨feature⟩)|
(BUILDQ ⟨fragment⟩⟨register⟩*)|
(LIST ⟨form⟩*)|
(APPEND ⟨form⟩⟨form⟩)|
(QUOTE ⟨arbitrary structure⟩)

Ações entre Redes

Ações Terminais (travessia de arcos)

Formas (processamento de constituintes)

Entrada para outras MT's

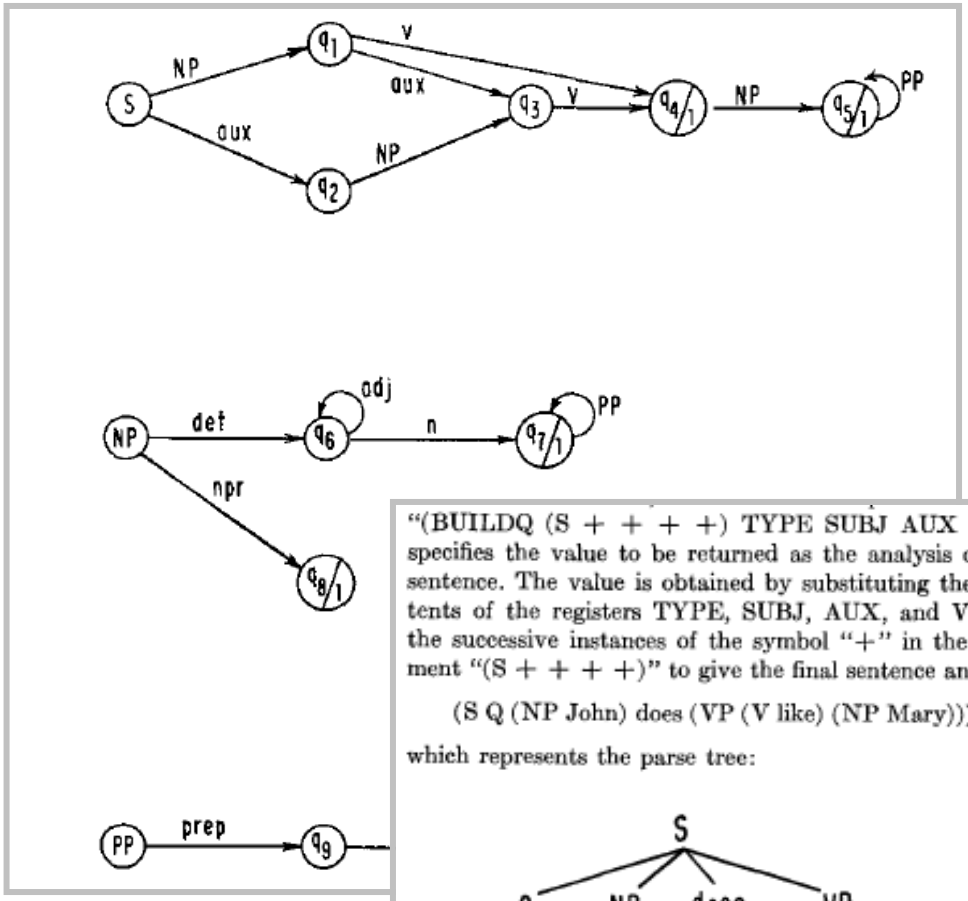
Especificação formal das ATN's (Woods, 1970: p.593)

⟨action⟩ → (SETR ⟨register⟩⟨form⟩)|
(SENDR ⟨register⟩⟨form⟩)|
(LIFTR ⟨register⟩⟨form⟩)

⟨term act⟩ → (TO ⟨state⟩)|
(JUMP ⟨state⟩)

⟨form⟩ → (GETR ⟨register⟩)|
*|
(GETF ⟨feature⟩)|
(BUILDQ ⟨fragment⟩⟨register⟩*)|
(LIST ⟨form⟩*)|
(APPEND ⟨form⟩⟨form⟩)|
(QUOTE ⟨arbitrary structure⟩)

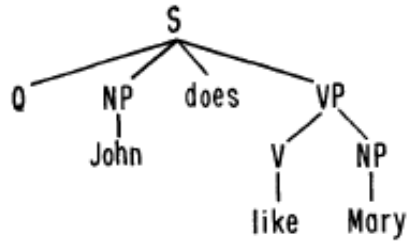
Estruturas Recursivas



"(BUILDQ (S + + + +) TYPE SUBJ AUX VP)" specifies the value to be returned as the analysis of the sentence. The value is obtained by substituting the contents of the registers TYPE, SUBJ, AUX, and VP for the successive instances of the symbol "+" in the fragment "(S + + + +)" to give the final sentence analysis

(S Q (NP John) does (VP (V like) (NP Mary))),

which represents the parse tree:




```
(S/ (PUSH NP/ T
        (SETR SUBJ *)
        (SETR TYPE (QUOTE DCL))
        (TO Q1))
    (CAT AUX T
        (SETR AUX *)
        (SETR TYPE (QUOTE Q))
        (TO Q2)))
(Q1 (CAT V T
        (SETR AUX NIL)
        (SETR V *)
        (TO Q4))
    (CAT AUX T
        (SETR AUX *)
        (TO Q3)))
(Q2 (PUSH NP/ T
        (SETR SUBJ *)
        (TO Q3)))
(Q3 (CAT V T
        (SETR V *)
        (TO Q4)))
(Q4 (POP (BUILDQ (S+++(VP+)) TYPE SUBJ AUX V) T)
    (PUSH NP/ T
        (SETR VP (BUILDQ (VP (V+) *) V))
        (TO Q5)))
(Q5 (POP (BUILDQ (S++++) TYPE SUBJ AUX VP) T)
    (PUSH PP/ T
        (SETR VP (APPEND (GETR VP) (LIST *)))
        (TO Q5)))
```

FIG. 3. An illustrative fragment of an augmented transition network

PUC RIO

DCG's – *Definite-Clause Grammars*

- Propostas por Fernando Pereira e David Warren em 1980, como formalismo equivalente a ATN's mas que herda o estilo e as facilidades da *programação em lógica*.

- Ao passo que as ATN's estão mais próximas de uma especificação através de redes de transição, as DCG's estão mais próximas de uma especificação por *regra de reescrita*.

- Têm 'alta usabilidade' para pesquisadores em linguística formal: os programas para processar LN são bem mais 'manejáveis' do que no caso das ATN's.

Pereira & Warren, 1980: pp. 241-242

We now show a simple CFG to illustrate the notation. The grammar covers sentences such as "John loves Mary" and "Every man that lives loves a woman":

sentence → noun_phrase, verb_phrase.
noun_phrase → determiner, noun, rel_clause.
noun_phrase → name.
verb_phrase → trans_verb, noun_phrase.
verb_phrase → intrans_verb.
rel_clause → [that], verb_phrase.
rel_clause → [].
determiner → [every].
determiner → [a].
noun → [man].
noun → [woman].
name → [john].
name → [mary].
trans_verb → [loves].
intrans_verb → [lives].

3.2.1. *Notation*

The notation for DCGs extends our notation for context-free grammars in the following way:

(1) **Non-terminals** are allowed to be compound terms in addition to the simple atoms allowed in the context-free case, e.g.

np(X,S) sentence(S)

(2) In the right-hand side of a rule, in addition to non-terminals and lists of terminals, there may also be sequences of **procedure calls**, written within the brackets '{' and '}'. These are used to express extra conditions which must be satisfied for the rule to be valid, e.g.,

Funções arbitrárias entre '{ }' (MT's)

noun(N) → [W], {rootform(W,N), is_noun(N)}.

The last example can be read as "a phrase identified as the noun N may consist of the single word W, where N is the root form of W and N is a noun".

Non-terminals, terminals and procedure calls in the right-hand side of a rule will be referred to collectively as **goals.**

Pereira & Warren, 1980: p. 251

For the first example, we want to recognise sentences comprising a verb 'precedes' or 'follows', and names of months, e.g., "May follows April". The interpretation given to a sentence of this type will be a term of the form 'before(M1,M2)'. For instance, the interpretation of "May follows April" will be 'before(april,may)'. The context-free grammar for this example is given by the rules:

```
sentence → month,verb,month.

verb → [precedes].
verb → [follows].

month → [january].
month → [february].
   etc.
```

Integração de Sintaxe & Semântica

To construct the required interpretation, we give the non-terminals extra arguments as follows:

```
sentence(S) → month(M1),verb(M1,M2,S),month(M2).

verb(M1,M2,before(M1,M2)) → [precedes].
verb(M1,M2,before(M2,M1)) → [follows].

month(january) → [january].
month(february) → [february].
   etc.
```

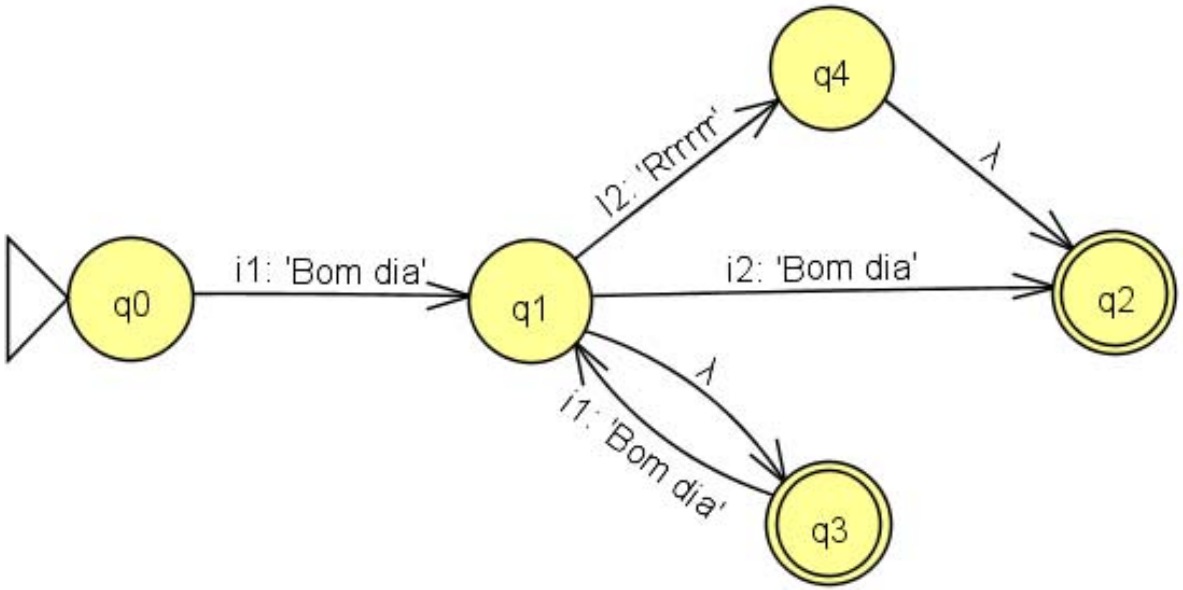
We read the non-terminal 'verb(M1,M2,S)' as "a verb whose interpretation in the context of a subject M1 and object M2, is S".

Exercícios e exemplos em sala

- Parsing / Geração de sentença simples
- Trabalhando com restrições simples
- Introdução de Flexão (Gênero e Número)
- Introdução de Semântica
- Introdução de Pragmática

- Uso das chamadas de funções arbitrárias:
  - Tradução
  - Caminhamento sobre modelos

Especificação gramatical de modelo de ‘conversa’ ;-)



Especificação gramatical de movimentação no plano

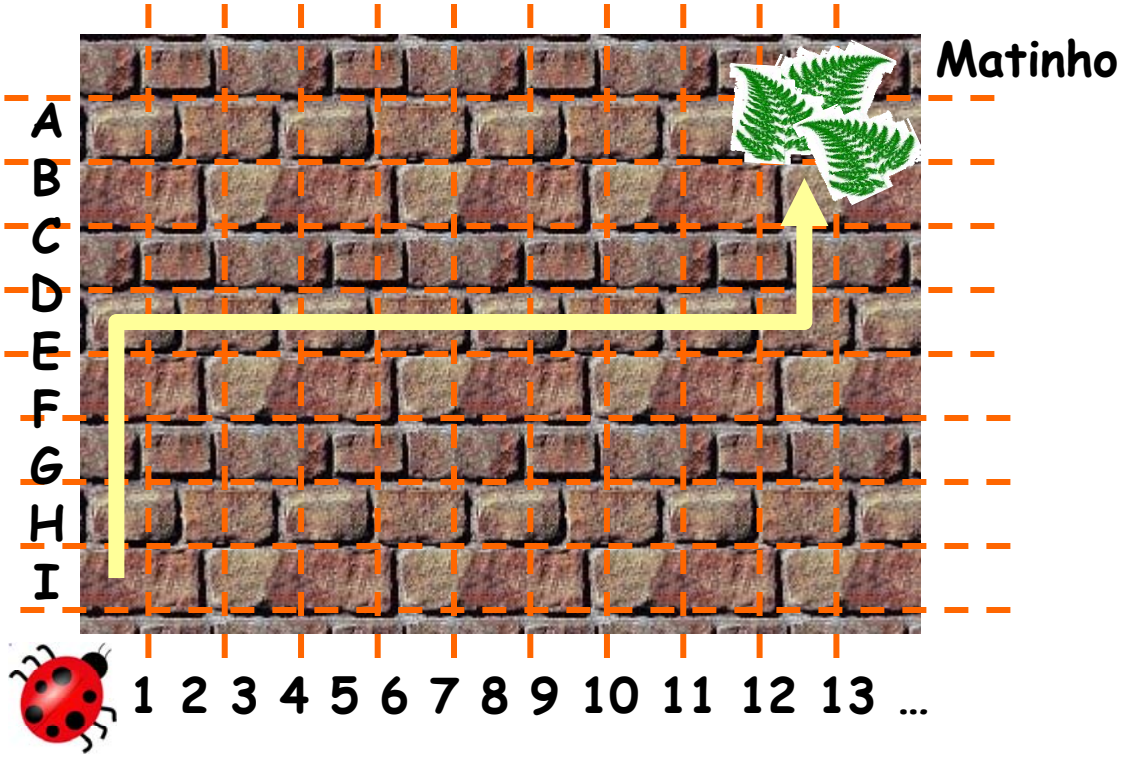
Ações da Joaninha

- Move Up
- Move Down
- Move Left
- Move Righ

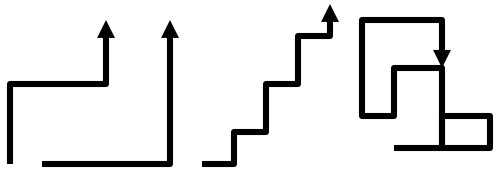
Exemplo de Trajeto

Move Up, Move Up, Move Up, Move Up, Move Left, …, Move Up, Move Up, Stop.

Coords(Matinho, X,Y)
Coords(Joaninha,W,Z)
perto ([X,Y],[W,Z])



Padrões de trajetos a verbalizar:



PUC RIO

Leituras sugeridas

1. W. A. Woods. 1970. **Transition network grammars for natural language analysis.**Communications of the ACM 13, 10 (October 1970), 591-606. DOI=10.1145/355598.362773 http://doi.acm.org/10.1145/355598.362773
2. Fernando C.N. Pereira, David H.D. Warren. 1980. **Definite clause grammars for language analysis -- A survey of the formalism and a comparison with augmented transition networks**. Artificial Intelligence 13, 3 (May 1980), 231-278, ISSN 0004-3702. DOI: 10.1016/0004-3702(80)90003-X. http://www.sciencedirect.com/science/article/pii/000437028090003
3. Stuart C. Shapiro. 1982. **Generalized augmented transition network grammars for generation from semantic networks**. Computational Linguistics 8, 1 (January 1982), 12-25. http://acl.ldc.upenn.edu/J/J82/J82-1002.pdf