Universidade Regional de Blumenau
Centro Tecnológico
Departamento de Sistemas e Computação

Interface em Linguagem Natural

Trabalho de Conclusão de Curso

Jomi Fred Hübner

para obtenção do título de
Bacharel em Ciências da Computação

Blumenau, novembro de 1992

Universidade Regional de Blumenau
Centro Tecnológico
Departamento de Sistemas e Computação
Trabalho de Conclusão de Curso

Interface em Linguagem Natural

Aluno: Jomi Fred Hübner
Orientador: Paulo de Tarso Luna - M. Eng.

Blumenau, novembro de 1992

Agradecimentos

Ao professor Roque, coordenador do curso, pela iniciativa de implantar o TCC, fato que implicou na experiência deste trabalho, aos colegas, aos professores e funcionários da FURB que tornaram agradavél a nossa estada nesta Universidade.

Ao professor e orientador, Paulo Luna, que me auxiliou, não somente neste trabalho, mas em muitas outras atividades.

Especialmente a Ilze que tem dispensado seu tempo para me auxiliar, motivar e amar.

Índice

Índice de Figuras

1. Introdução

1.1. Importância das Interfaces em Linguagem Natural

Os sistemas computacionais atuais atingiram um patamar onde há grande disponibilidade de hardware, onde o software é cada vez mais complexo e especializado. Conseqüentemente, os usuários têm várias opções de sistemas para cada uma de suas diversas necessidades. Entretanto, para que o usuário utilize convenientemente tais sistemas, deve demandar considerável esforço para aprender a utiliza-los. Perdendo as vezes, muito tempo até utiliza-los efetivamente, tempo este, não disponível ao usuário.

Grande parte das pessoas - principalmente usuários ocasionais, como executivos - que poderiam fazer bom uso dos recursos disponíveis solucionando muitos dos seus problemas, não o fazem por "medo" do computador, "medo" da complexidade e da diversidade das interfaces utilizadas. Os que se aventuram na tentativa de descobrir todos os segredos do sistema só o conseguem adaptando-se a interface oferecida pelo sistema.

1.2. Objetivo

O **P**rocessamento de **L**inguagem **N**atural (PLN) é uma área de estudo da **I**nteligência **A**rtificial (IA) que têm por objetivo dotar as interfaces de computadores da

capacidade de comunicar-se com seu usuário na língua deste. Para atingir este fim existem as seguintes sub-áreas de estudo no PLN:

- Conversão da fala em texto;
- Compreensão do texto (**C**ompreensão de **L**inguagem **N**atural - CLN);
- Geração de fala (conversão de texto em voz);
- Tradução de textos;
- Correção de textos.

Este trabalho irá se ater ao segundo item, tendo por objetivo ilustrar técnicas utilizadas na implementação de interfaces em **L**inguagem **N**atural (LN). Para a concretização deste objetivo será desenvolvido, em PROLOG, um protótipo de interface em LN.

Inicialmente a função do protótipo será converter uma entrada em LN para uma linguagem com gramática formal, entendida por computador. Na implementação, o vocabulário será limitado a um universo específico, no caso, o da administração financeira, especificamente, o ciclo económico financeiro.

Na criação do protótipo haverá a preocupação em criar as funções de forma genérica, possibilitando sua reutilização em outros experimentos.

1.3. Justificativa

Existem muitas formas de comunicação programa-usuário, por exemplo: menus, linguagem de comandos, voz, gráfica, linguagem natural, e outras. Dentre estas, a interface gráfica tem sido uma tendência dos sistemas atuais, principalmente com o surgimento do ambiente *windows*.

O ambiente *windows* tem como vantagem ser fácil de usar, devido a comunicação por meio de ícones, earcones (ícones sonoros), janelas, caixas de diálogo, etc. Entretanto, apresenta limitações quanto ao poder de

expressão em situações onde um grande número de alternativas está disponível, sendo necessárias vários ícones. O mesmo problema se apresenta no caso das interfaces por menu [MONT91]. Estas interfaces tem ainda um outro problema: tornam-se cansativas quando há muitos níveis hierárquicos de menus.

Nas interfaces de linguagens de comando o usuário comunica-se com o programa por meio de teclas de função ou códigos com funções pré-estabelecidas. Por exemplo: F2 - salva, CLS - limpa tela, etc. Esta abordage tem como problema a rigidez de sintaxe, forçando o usuário a decora-la, ou ter o manual sempre a disposição. Em contra partida, tem um bom aproveitamento, tanto a nível de capacidade de expressão, como performance, para usuários experientes.

A LN tem um poder de expressão muito grande. Tudo que se pensa é pensado em uma linguagem: uma linguagem natural. Sua utilização torna-se muito simples ao usuário comum, bastando a ele comunicar-se da forma que está habituado, não precisando conhecer o computador/sistema. Entretanto, o uso constante desta interface torna-se enfadonho quando, para exprimir um ideia, temos que escrever toda uma frase. Um recurso que solucionar este problema é o uso de uma interface de voz.

Para que um sistema possa ter uma interface com Compreensão da Linguagem Natural, este deve possuir muitas características próprias dos seres humanos como: aquisição, retenção e exposição do conhecimento. Estas tarefas são alvo de estudo da IA, sendo que técnicas que proporcionem aos sistemas as características acima e que serão estudadas neste trabalho, podem ser utilizadas em muitas outras áreas de atuação da IA. Ou seja, a compressão da linguagem natural está intimamente ligada às faculdades mentais humanas.

"A linguagem interpenetra a experiência, de tal forma, que as categorias mais profundas do pensamento são diferentes nas diferentes culturas. Quando os homens se comunicam, eles fazem muito mais que apenas informar" ([HOUA91] p.42)

Optou-se pelo uso da linguagem de programação PROLOG pelas seguintes características: (1) grande parte das experiências em LN tem sua implementações em PROLOG, fornecendo, assim, um bom acervo de experiências já testadas e comprovadas; (2) o tempo gasto para o desenvolvimento pode ser reduzido, o que no caso de um protótipo é relevante, já que o PROLOG traz consigo implementado um mecanismo de inferência.

Os recursos computacionais atuais já permitem a construção de interfaces em LN utilizáveis comercialmente em domínios restritos.

1.4. Organização do Trabalho

Na primeira parte do trabalho (capítulo 2) é feito um levantamento histórico das teorias e técnicas utilizadas na CLN. Neste capítulo, inicialmente é vista a evolução da lingüística juntamente com seus conceitos básicos, segundo Saussure e Chomsky. São estudados os tipo de gramática, sua representação através de árvores de derivação e notação com base em clausulas. Em seguida, são vistos brevemente os conceitos da gramática transformacional. Ao final, são descritas as principais linhas de pesquisa para sistemas com interface em LN.

No terceiro capítulo, é descrita a construção do protótipo. Primeiramente, é explicado o macro funcionamento do modelo proposto. Em seguida, é abordado cada parte do modelo, com suas caraterísticas, dificuldades de implementação e soluções encontradas. Destacam-se os módulos: (1) analisador sintático, onde é verificada a gramática da frase; (2) analisador semântico, que interpreta a frase; e (3) o Simulador, que executa o comando. Ao final do trabalho é feito um exemplo completo do funcionamento do modelo, desde a frase digitada até o resultado ser mostrado ao usuário.

2. Histórico

Por muito tempo a história da CLN esteve nas mão dos lingüístas, que formularam os primeiros conceitos e desenvolveram as primeiras técnicas para a CLN. Somente mais tarde, com o surgimento de computador e da IA, a CLN passou a ser assunto também para os cientistas da computação.

Considerando este passado da CLN, vê-se oportuno um breve histórico da lingüística.

2.1. Lingüística

Na história da lingüística houve dois momentos-chave, conforme [ORLA90], marcados por duas tendências:

1ª) Formalismo (séc. XVII) ⇒ se preocupou com o percurso psíquico da linguagem, observando a relação entre linguagem e pensamento. Buscou-se nas diversas línguas, o que é único, universal, constante. Época onde os estudos foram fortemente marcados pelo racionalismo, estudando a linguagem como representação do pensamento e procurando mostrar que as línguas obedecem princípios racionais gerais e lógicos. Surgiu desta escola as gramáticas gerais, que regem todas as línguas, um modelo desta gramática é a Gramática de *Port Royal* dos franceses Ci. Lancelot e A. Arnaud (1690).

2ª) Sociologismo (séc. XIX) ⇒ se preocupou em estudar o percurso social, explorando a relação entre língua e sociedade. Procurou o que é múltiplo, variado e diverso. Estudou principalmente as mudanças e a história das línguas no tempo. O alvo visado é a língua-mãe, da qual todas as outras surgiram.

A lingüística como ciência surge somente no século XX, na Suiça, a partir dos trabalhos do pesquisador (considerado o pai da lingüística moderna) Ferdinand Saussure com sua principal obra: Curso da Lingüística Geral (1916).

A contribuição de Saussure foi criar uma fundamentação teórica para a lingüística, lançando seus principais conceitos. A seguir, alguns destes conceitos serão brevemente explicados, conforme [ORLA90].

Semiologia ⇒ ciência que estuda os signos.

Signos ⇒ sinais que o ser humano produz quando fala, canta, desenha, escreve. Com eles o ser humano se comunica, representa seus pensamentos, exerce seu poder, elabora sua cultura, sua identidade, etc.

Meta linguagem ⇒ é o ato de usar a linguagem para falar da própria linguagem.

Lingüística ⇒ ciência que estuda os signos verbais.

Objeto da Lingüística ⇒ Língua

Língua ⇒ é um conjunto de signos.

Signo verbal ⇒ é uma associação entre um *significante* (imagem acústica ou som) e um *significado* (conceito). Por exemplo: o significante "cão" tem como significado: animal quadrupede, mamífero, etc.
O laço que une o significante ao significado é arbitrário, convencional e imotivado. Não há razão para que "cão" se chame "cão". O

significado se define pela posição que ocupa na rede de relações que constitui o sistema total da língua.
Uma boa analogia para a compreensão deste conceito é o jogo de xadrez. Nele cada peça pode ser vista como um signo, que tem forma arbitrária (há pessoas que jogam xadrez com sementes), um significado comum aos jogadores e somente pode ser compreendida se vista no contexto de um jogo particular. Neste exemplo a língua que os jogadores estariam "falando" seria composta por um conjunto de signos (peças) e de regras conhecidas.

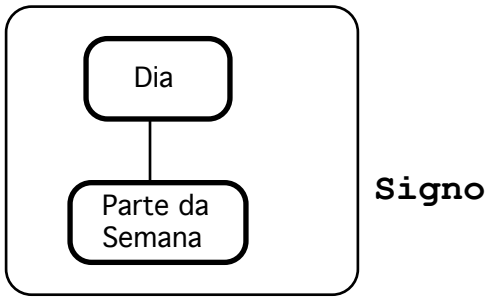


Figura 2.1

Disciplinas da lingüística:

- Fonologia : estudo das unidades sonoras;
- Sintaxe : estudo das estruturas das frases;
- Morfologia: estudo das formas das palavras;
- Semântica : estudo dos significados.

2.2. Gramática

Um problema que surgiu no estudo das línguas, principalmente para os lingüístas da corrente formalista, é: como representar e definir uma linguagem?

Para [HOUA91] cada língua é um sistema altamente organizado e estruturado, em conseqüência, o saber interiorizado do falante é de uma enorme complexidade interna. De que modo, então, o falante possui este

sistema interiorizado, e o põe em execução, sempre que fala?

Um grande pensador nesta área da lingüística foi Noam Chomsky (1950). Chomsky propõe um teoria a que chama gramática e centra seus estudos na sintaxe.

Conforme Chomsky, existe por trás da língua, de um modo não palpável, um corpo de generalizações, princípios e regras abstratas em número finito, que determinam as frases da língua, a sua gramaticalidade, suas propriedades e características. Este corpo altamente organizado chama-se gramática. Cada ser humano possui então uma gramática interiorizada adquirida enquanto criança num período relativamente curto e possivelmente na base de alguns princípios inatos, próprios à espécie humana, "a faculdade da linguagem".

"A pessoa que adquiriu conhecimento de um língua interiorizou um sistema de regras que relaciona o som e o significado de um modo particular. O lingüísta ao construir a gramática de um língua está efetivamente propondo uma hipótese com relação a esse sistema interiorizado (...) Evidentemente, o conhecimento da língua - o sistema interiorizado de regras - é apenas um dos muitos fatores que determinam o modo como a expressão oral será usada ou entendida em um situação particular." [CARV82]

Alguns conceitos básicos são necessários à uma introdução nesta área, são eles:

Gramática Gerativa ⇒ número limitado de regras a partir do qual se pode gerar um número infinito de frases que formam uma língua, dando-lhe um caráter aberto, dinâmico e criativo.

Frase ⇒ é uma unidade de linguagem que comunica um pensamento ou a intenção de uma pessoa.

Sintaxe ⇒ é o estudo das regras que determinam quais cadeias de palavras de um vocabulário podem

formar frases. Nem toda cadeia de palavras sintaticamente correta é uma frase.

Para ilustrar o conceito de gramática generativa, vamos supor um vocabulário limitado a um par de letras {a, b}. E frases bem formadas, que respeitam a sintaxe da linguagem, seriam as seguintes:

abba, bbaaabbaaabb, bababbabab.

onde, a segunda metade é igual à primeira invertida. As regras sintáticas que geraram estas frases podem ser as seguintes:

(1) S → aa (2) S → aSa
(3) S → bb (4) S → bSb

ou, convencionalmente:

S → aa | aSa | bb | bSb

onde | pode ser lido como "ou".

Estas regras geram as frases conforme o seguinte procedimento: inicia-se com o símbolo *S* e o substituí pelo lado direito da regra, se este lado também possuir um S, repete-se o processo até que não haja mais S. Por exemplo:

S
(2) aSa
(4) abSba
(4) abbSbba
(2) abbaSabba
(3) abbabbabba

os elementos escritos em minúsculo são chamados **terminais** (no caso *a* e *b*), ou seja, partindo somente deles não é mais possível aplicar-se qualquer regra. Enquanto os escritos em maiúsculo são **não-terminais** (no caso *S*).

A partir deste exemplo, pode-se ilustrar como definir formalmente uma gramática (G). Ela é composta por quatro conjuntos de elementos. Assim:

G = (N,Σ,P,S_O), onde
N - conjunto dos não-terminais
Σ - conjunto dos terminais
P - regras gramaticais ou de produção
S_O - é um não-terminal que serve como símbolo inicial

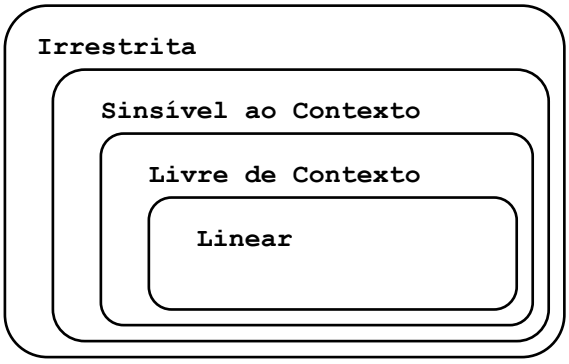
por exemplo:

G = ({S},{a,b},{S→aa, S→aSa, S→bb, S→bSb}, S_O).

A linguagem gerada por esta gramática é representada como:

L(G) = {conjunto de todas as frases de G}

As regras de produção podem ser formadas de muitas maneiras, caracterizando seu poder de expressão e conseqüentemente implicando na complexidade da linguagem. Chomsky classificou as gramáticas como sendo de quatro tipos: linear, livre de contexto, sensível ao contexto e irrestrita. Pode ser observado que uma gramática irrestrita engloba uma gramática sensível ao contexto que, por sua vez, engloba uma gramática livre de contexto que engloba a linear.



- **Linear**: um gramática é dita linear se todas as regras de P apresentam o seguinte formato:

  X → αY ou X → α.

onde X e Y são não-terminais pertencentes a N e α é um terminal, ou seqüência de terminais, (pertencente a Σ).

Exemplo 2.1.: S → aX | bY
Y → a | aX
X → b | bY

♦ **Livre de Contexto** (Context Free - CF): se todas as regras de produção de P tem o seguinte formato:

$X \rightarrow \alpha$

onde X está em N e α está em $(N \cup \Sigma)^+$.

$(N \cup \Sigma)$ é a união dos não-terminais com os terminais;
$^+$ o conjunto resultante tem um ou mais elementos.
* o conjunto resultante tem zero ou mais elementos.

Exemplo 2.2.: a gramática: S → bX | aSSb
X → a | abXb | aSb

gera sequências como: S ⇨ bX
⇨ babXb
⇨ babab

♦ **Sensível ao contexto** (Context Sensitive - CS): se todas as regras de produção de P possuem o seguinte formato:

$y_1 X y_2 \rightarrow y_1 \alpha y_2$

onde, X pertence a N; y_1, y_2 e α estão em $(N \cup \Sigma)^+$. Intuitivamente, X pode ser restrito como α, mas somente no contexto de y_1 e y_2.

Exemplo 2.3.: uma gramática que gera uma linguagem onde $L(G) = \{ a^n b^n c^n \mid n \geq 1\}$ e $\Sigma = \{a,b,c\}$, pode ter P:

a) S → abc | aX_1bc
b) X_1c → X_2bcc
c) aX_2 → aaX_1 | aa
d) bX_2 → X_2b
e) X_1b → bX_1

onde a seguinte seqüência pode ser gerada:

$S \Rightarrow aX_1bc$
$\Rightarrow abX_1c$
$\Rightarrow abX_2bcc$
$\Rightarrow aX_2bbcc$
$\Rightarrow aaX_1bbcc$
$\Rightarrow aabX_1bcc$
$\Rightarrow aabbX_1xx$
$\Rightarrow aabbX_2bccc$
$\Rightarrow aaX_2bbbccc$
$\Rightarrow aaabbbccc = (a^3b^3c^3)$

Em uma gramática deste tipo as regras somente podem ser utilizadas em contextos onde o lado-esquerdo, incluindo seus terminais e não-terminais, "casa" com o conjunto que se deseja provar. Na gramática definida para o exemplo 2.3 o não-terminal X_1 só pode ser substituído se for seguido por c (regra b) ou por b (regra e).

- **Irrestrita** : se todas as regras de produção de P apresentam a seguinte forma:

$$\alpha \rightarrow \beta$$

onde, α pertence ao conjunto $(N \cup \Sigma)^+$ e
β pertence a $(N \cup \Sigma)^*$.

Exemplo 2.4.: A → BCD
BC → aBb
C → null (característica desta gramática)
....

Grande parte das implementações de verificadores de sintaxe, como os utilizados em compiladores, *queries* de bancos de dados, e até mesmo para linguagem natural, utilizam gramáticas do tipo livre de contexto. Isto se deve ao fato desta gramática ter uma performance muito melhor devido à sua maior simplicidade de implementação e utilização e porque atende aos principais requisitos exigidos por estas aplicações.

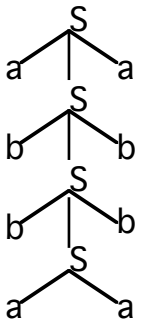
2.3. Parsers ou Árvores de Derivação

Na geração de sentenças para gramáticas lineares e livres de contexto podemos observar que a partir do símbolo inicial S_O, outros símbolos são "chamados" até que a seqüência de caracteres da entrada esteja completa. Este processo pode muito bem ser representado por uma árvore, onde S_O é a raiz e os não-terminais (X,Y,...) da regra S_O seus ramos. Os ramos (X, Y.), por sua vez, podem ter outros ramos, até chegarmos as folhas da árvore, os símbolos terminais.

Exemplo 2.5.: para a gramática

S → aa | aSa | bb | bSb,

com a seqüência abbaabba, temos a seguinte árvore:



Exemplo 2.6.: vamos considerar uma gramática, onde:

N = { S,SN,SV,SP} onde:
S = Sentença;
SN = Sintagma Nominal;
SV = Sintagma Verbal; e
SP = Sintagma Preposicional.

Σ = { Artigo,Substantivo,Numeral,Verbo,Prep.}

P = { S → SN SV
SN → Artigo Substantivo
SN → Numeral
SN → SP
SV → Verbo
SV → Verbo SP
SV → Verbo SN
SP → Preposição SN
}

S_O = S

Podemos interpretar P (conjunto das regras gramaticais) da seguinte forma: um SN é formado por um artigo seguido

de um substantivo, ou somente um numeral, ou ainda um SP. A mesma interpretação pode ser feita para os demais regras.

Deve ser observado que **Σ** é formado por categorias de palavras, por exemplo, com as seguintes categorias:

Artigo = {o,a,os,as}
Substantivo = {dólar, mês, estoque}
Verbo = {passar,aumentar,atualizar,baixar}
Preposição = {para, por, de}

Poderiam ser geradas frases como:

(a) O dólar aumentou 30%.
(b) O estoque aumentou.

com árvores:

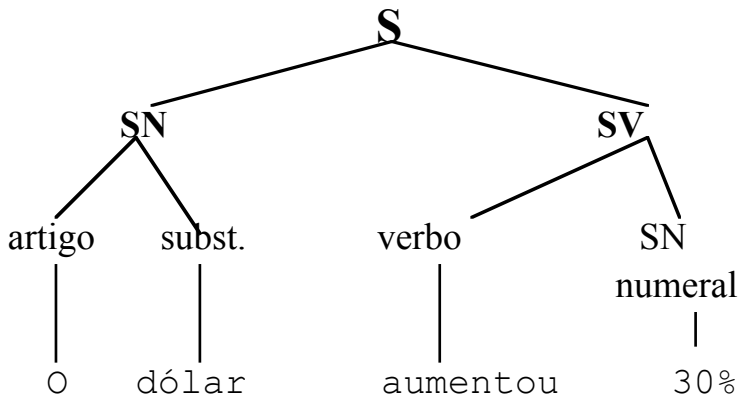


Figura 2.2
Árvore para a frase: O Dólar aumentou 30%.

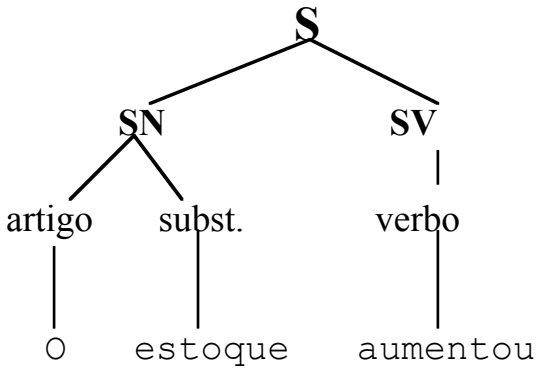


Figura 2.3
Árvore para a frase: O estoque aumentou.

Dado um $G=(N,\Sigma,P,S_O)$, o procedimento para criar uma árvore de derivação a partir das regras pode ser facilmente descrito como:

1°) Se $S_n \rightarrow X_1\ X_2\ X_3\ \ldots\ X_n$ pertencentes a P, então liga-se a raiz S_n aos nódos $X_1\ X_2\ X_3\ \ldots\ X_n$ como sendo seus descendentes.

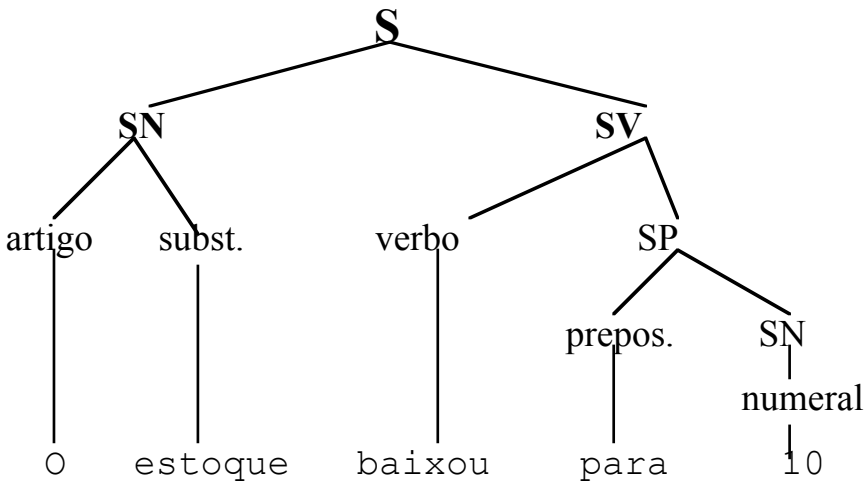
2°) Para cada descendente, $X_i \rightarrow Y_1\ Y_2\ Y_3\ \ldots\ Y_n$, liga-se X_i aos nódos $Y_1\ Y_2\ Y_3\ \ldots\ Y_n$ como sendo seus descendentes.

3°) Continue até que todos os conjuntos de descendentes sejam terminais, ou strings vazias.

Exemplo 2.7.: Utilizando a gramática definida no exemplo 2.6, e a frase:

O estoque baixou para 10.

as seguintes árvores de derivação poderiam ser criadas:



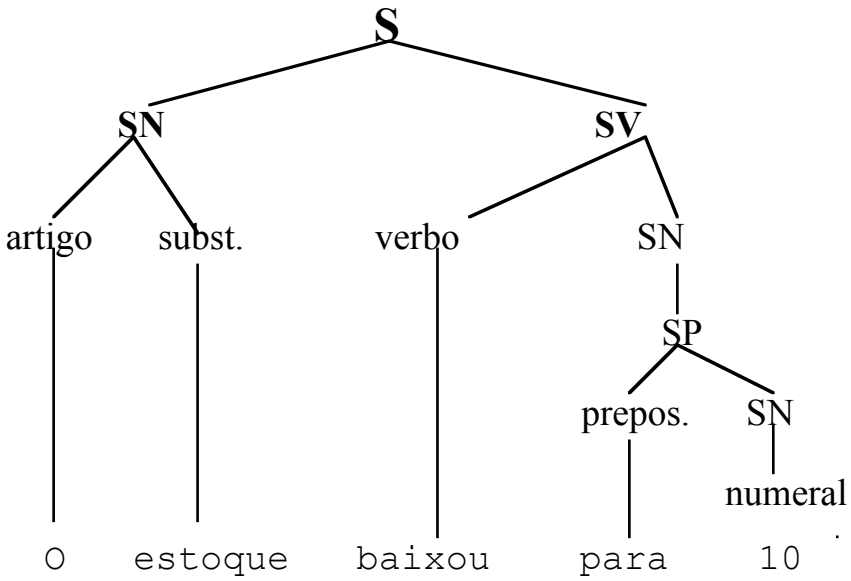


Figura 2.4
Duas possibilidades de árvore para a frase:
O estoque baixou para 10.

O que ocorre aqui é uma ambigüidade sintática, ou seja, a frase pode ser avaliada sintaticamente de duas formas, pois tem duas árvores de derivação possíveis. "A ambigüidade sintática é uma propriedade de cada gramática individualmente e não da linguagem em si"[KRUL92].

Para solucionar este problema deve-se encontrar uma **gramática equivalente.** Duas gramáticas são ditas equivalentes se as linguagens geradas são as mesmas:

$G_1 \Leftrightarrow G_2$ se $L(G_1) = L(G_2)$

No nosso exemplo, a seguinte gramática é equivalente, entretanto não apresenta ambigüidade:

(1) S → SN SV
(2) SN → Artigo Substantivo
(3) SN → Numeral
(4) SN → SP
(5) SV → Verbo
(6) ~~SV → Verbo SP~~
(7) SV → Verbo SN
(8) SP → Preposição SN

Como SN → SP, a regra (6) é equivalente a regra (7), podendo ser eliminada.

Para representar uma árvore de derivação convencionou-se a utilização de uma **notação clausal**. Nesta notação os nós da árvore (lado-esquerdo da regra) são substituídos por cláusulas, e seus ramos (lado-direito da regra) são termos da cláusula.

Exemplo: representação clausal para as árvores da:

figura 2.2 ⇒ s(sn,sv)
⇒ s(sn(dólar),sv(aumentar,sn))
⇒ s(sn(dólar),sv(aumentar,sn(num(30,%)))

2.3 ⇒ s(sn(estoque),sv(aumentou))
2.4 ⇒ s(sn(estoque),sv(baixou,sp(para,sn(num(10)))))

Esta notação tem como vantagens: (1) ser facilmente implantada numa linguagem declarativa (como PROLOG); e (2) dispensa o desenho de uma árvore para representar a frase, já que o desenho, apesar de mais explicativo, requer mais tempo para ser feito e sua representação numa estrutura de dados é complexa.

2.4. Gramática transformacional

Outra contribuição de Chomsky para a lingüística foi a Gramática Transformacional (GT) [CHOM65]. Esta gramática foi desenvolvida em virtude da gramática estrutural (descrita acima) apresentar problemas para representar certos tipos de frases.

A GT contém dois componentes básicos, um é chamado **Componente Base** e outro **Componente Transformacional**.

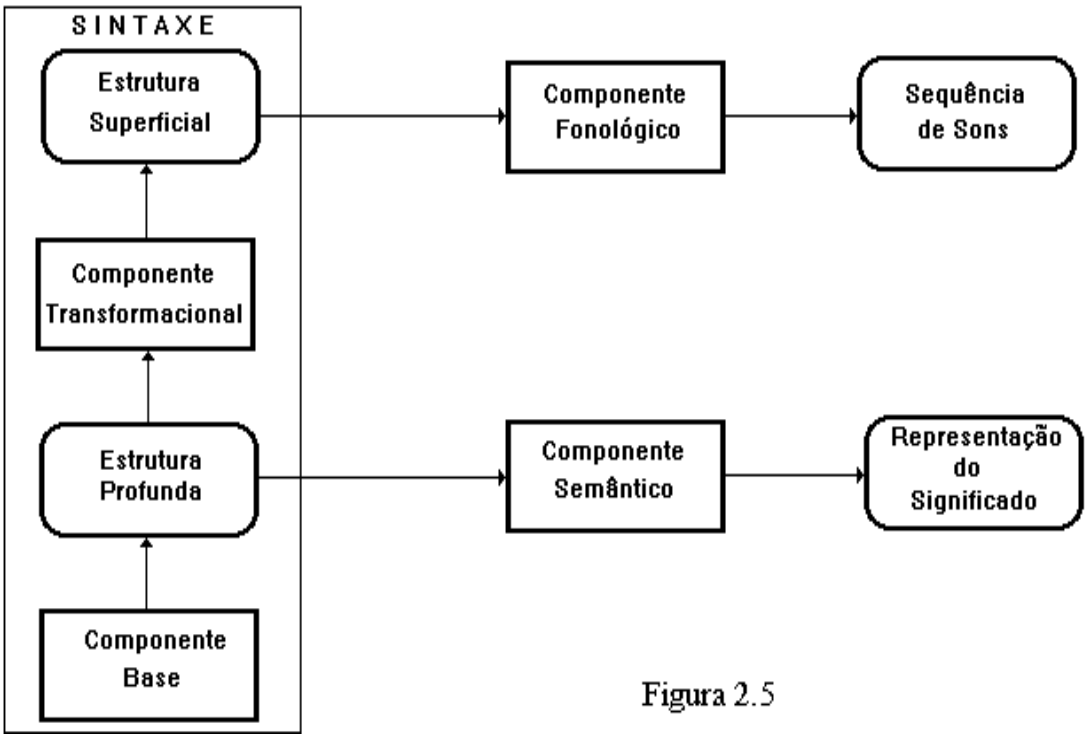


Figura 2.5

O primeiro é um conjunto de regras que determinam como serão montadas as frases, da mesma forma que na gramática estrutural. Mas acrescenta outras características aos componentes da frase, na forma de meta-variáveis, ou seja, cada frase possui um conjunto de variáveis que são instanciadas quando se atinge um símbolo terminal.

Por exemplo, na regra

SN → Nome[Número,Pessoa]

Número e Pessoa são variáveis que são instânciadas assim que um nome é encontrado, a partir deste momento todos os terminais que tem associado as variáveis Número e/ou Pessoa devem concordar (ter mesmo número e/ou pessoa) com Nome.

Exemplo 2.8.: o Componente Base

```
S           → SN SV[Voz]
SN          → (Artigo) Nome[Número,Pessoa](Adjetivo)
SN          → Numeral
SV[ativa]   → Verbo[Tempo,Pessoa] (SN) (SP)
```

```
    SV[passiva]→ Aux[Tempo] Verbo[particípio](SN)(SP)
    SP         → preposição SN
```

onde os termos entre () são opcionais na regra.

Para a frase:
    o ciclo financeiro foi aumentado para 30.
a seguinte árvore é gerada:

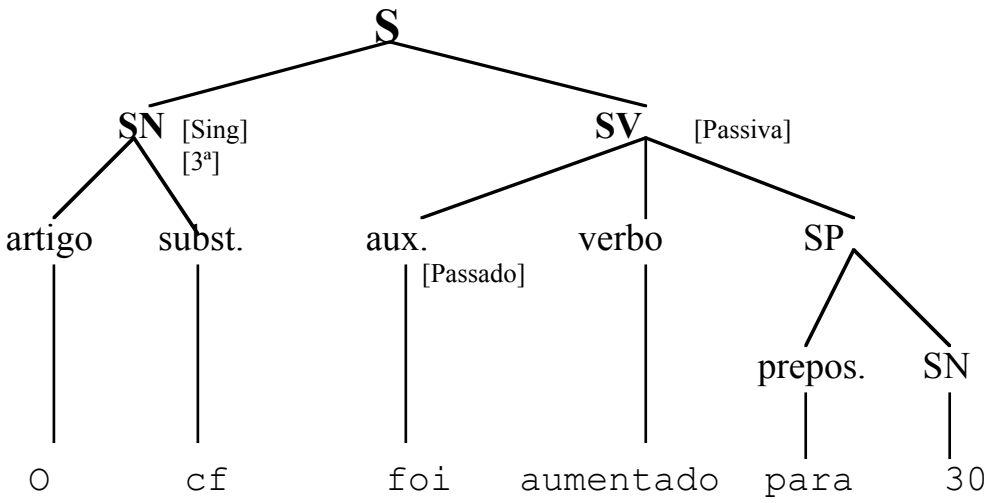


Figura 2.6

ou, em notação clausal:

```
   s(sn(ciclo financeiro[singular,3ª]),
     sv(foi[passado],aumentado,sn(num(30)))[passiva]
    )
```

O segundo componente da gramática transformacional é um conjunto de regras que transformam a frase da estrutura profunda (EP) gerada pelo componente base numa estrutura superficial (ES).

Estas regras de transformação possuem uma estrutura de descrição (ED) e uma de alteração (EA). Se a frase entrada tiver uma ED específica será transformada numa frase com uma EA específica.

Um exemplo simples: seja a regra de transformação

ED → EA, onde:

ED: s(sn(N),sv(V))

```
   EA: sv(V,sn(N))
```

para a frase:

```
   o dólar aumentou
```

o Componente Base cria a Estrutura Profunda

```
   EP: s(sn(dólar),sv(aumentou))
```

que é transformada, de acordo com a regra ED → EA considerada para:

```
   ES: sv(aumentou,sn(dólar))
```

ou, linearmente:

```
   ES: aumentou o dólar
```

Exemplo 2.9: com a GT podemos converter uma frase da voz passiva para a voz ativa.

```
   ED: s(sn(N[Pessoa]),
          sv(Aux[Tempo], V)[passiva]
         )
   EA: s(sn(N[Pessoa]),
          sv(V[Tempo,Pessoa])[ativa]
         )
```

para a frase:

```
   o ciclo financeiro foi aumentado.

   EP: s(sn(ciclo financeiro[3ª]),
          sv(foi[passado],aumentado)[passiva]
         )
   ES: s(sn(ciclo financeiro[3ª]),
          sv(aumentou)[ativa]
         )
```

2.5. Sistemas que Utilizam LN

Montenegro [MONT91] citando Feigenbaum [FEIG81], identifica quatro momentos na história da utilização de

interfaces em LN, marcados pelo surgimento de pesquisas e programas importantes. Estes momentos são:

- Os programas de linguagem natural iniciais procuravam obter somente resultados limitados em domínios específicos. Nos programas que se enquadram neste período estão: BASEBALL de Green, SAD-SAM de Lindsay, STUDENT de Bobrow e ELISA de Weizenbaum. Utilizam estruturas de dados para estocar informações sobre o domínio. As sentenças de entrada devem ser simples interrogativas, da qual o programa retira palavras-chave que serão utilizadas para encontrar a resposta. Por causa do domínio restrito e do processamento de palavras-chave, estes sistemas conseguem ignorar muitas das complexidades da linguagem e, muitas vezes, obter resultados bons.

- Outra abordagem foi tentada no PROTO-SYNTHEX-I (Simmons, Burger e Long, 1966) e na utilização de memórias semânticas (Quillian 1968). Estes sistemas estocam a representação do texto numa base de dados, usando uma variedade de esquemas (indexação inteligente) para recuperar materiais contendo palavras e frases específicas. Em consequência não tem problemas de domínio, embora somente respondam à perguntas para as quais foi explicitamente armazenada uma resposta na base de dados, ou seja, não utilizam nenhum mecanismo de inferência.

- Para abordar o problema de como caracterizar e usar o significado da sentença, um terceiro grupo de programas foi desenvolvido durante os anos 60. Nestes sistemas, onde se incluem: SIE (Raphael, 1968) e TLC (Quillian, 1969), a informação na base de dados foi estocada em notações formais. Existem mecanismos de tradução para converter as sentenças de entrada para a forma interna. Assim, estes sistemas podem inferir respostas que não estão armazenadas explicitamente na base, por exemplo: se os seguintes fatos estão na base: (a) Danger é um doberman; (b) dobermans são cães, estão o sistema pode inferir que Danger é um cão.

- O quarto grupo são os sistemas baseados em conhecimento, seu desenvolvimento é completamente entrelaçado com as

pesquisas em IA, principalmente quanto a representação de conhecimento. Estes programas usam de grande quantidade de informações sobre o domínio; para auxiliar o entendimento do conhecimento da sentença são estocados, dentro do programa, vários esquemas de representação do conhecimento, tais como: lógica, semântica procedural, redes semânticas, frames, etc.

A divisão em quatro momentos não implica que pesquisas em CLN atenham-se a um deles especificamente. Existem pesquisadores que combinam técnicas destes momentos.

3. O Modelo do Protótipo

3.1. Descrição

Para possibilitar melhor compreensão do problema por parte de pessoas não afetas à área de LN e para facilitar a definição, foi desenvolvido um modelo lógico para o protótipo, procurando representar diagramaticamente sua estrutura e funcionamento. Para isto, utilizou-se a filosofia pertinente ao paradigma da Análise Orientada a Objeto (AOO), conforme definições de Coad&Yourdon [COAD92], onde:

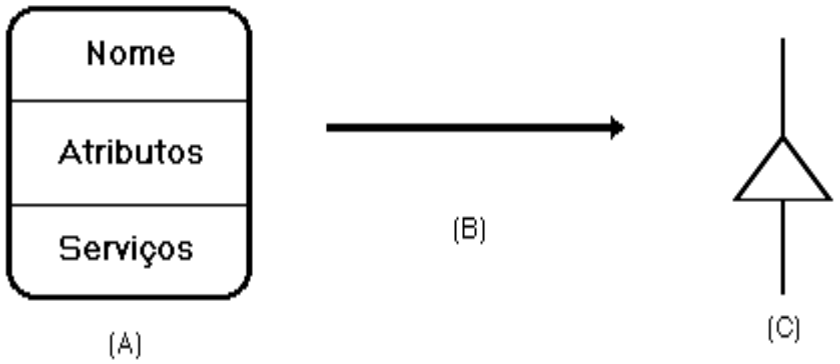


Figura 3.1

a figura 3.1.a representa um objeto, que representa um constituinte do mundo real. Um objeto possui intrinsecamente atributos (dados) e serviços (funções).

Os objetos têm a capacidade de agirem e se intercomunicarem aproximando-se de um modelo "vivo" de

elementos do mundo real. Para representar troca de mensagens entre objetos utiliza-se setas (figura 3.1.b) com origem no objeto emissor e apontando para o objeto destinatário da mensagem.

Da mesma forma que no mundo real, alguns objetos são formados por outros objetos, este tipo de estrutura é chamada de "estrutura todo-parte" e é representado pelo símbolo da figura 3.1.c, onde o objeto para o qual o símbolo aponta constitui o todo e o objeto da outra extremidade a parte.

Para o protótipo de interface em LN proposto chegou-se ao seguinte modelo:

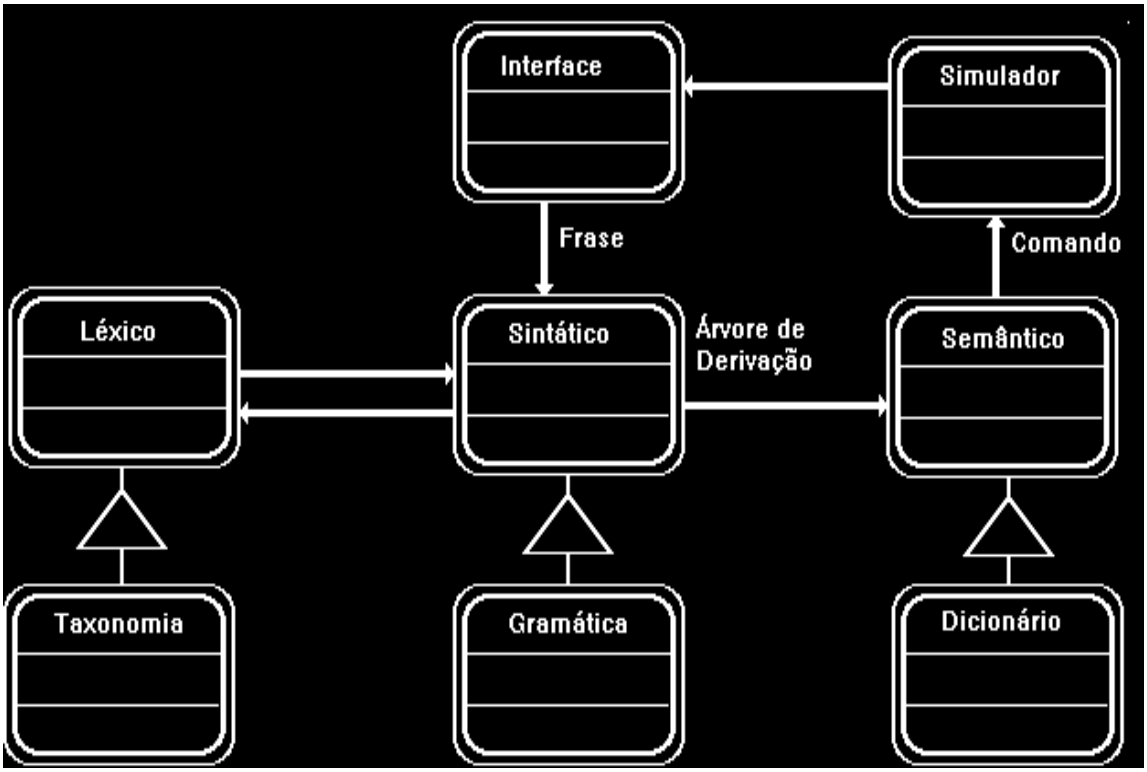


Figura 3.2

Os objetos do modelo acima apresentado são os seguintes:

Interface: é o objeto do protótipo responsável pela comunicação direta com o usuário, ou seja, ler as frases por ele digitadas e mostrar-lhe os resultados. Também compete à interface auxiliar o usuário quanto a operação do sistema.

Simulador: é o objeto ao qual são atribuídas todas as funções relativas a simulação do modelo financeiro proposto, cabe a ele fazer todos os cálculos requisitados por comandos e enviar à interface os resultados.

Analisador Sintático: é responsável pela verificação sintática da frase enviada pela Interface, isto é, verificar se a frase pertence à linguagem definida pela gramática. Concomitante a isto, tem a função de transformar a frase em si — que é linear — numa representação hierárquica, em forma de árvore, onde estão representadas as funções dos constituintes da frase. Nesta tarefa, para cada regra da gramática, existe uma outra regra que dita como montar a árvore analisada.

```
Exemplo.: F (frase) → pronome Inter.+Sn
          Sn(sintagma nominal) → Artigo+Nome
```

onde, para uma frase onde as regras acima foram utilizadas, será montada uma árvore como: questão(Nome); que significa: qual o valor de Nome.

```
F  → qual + Sn
Sn → o + cf
```

Após esta análise a árvore será enviada ao Analisador Semântico.

Gramática: neste objeto estão armazenadas todas as regras que determinam a estrutura da linguagem e a função dos elementos desta estrutura. É

a partir da gramática que podemos, por exemplo, saber que a frase "o menino carro pegou" está errada.

Analisador Léxico: verifica a forma (morfologia) de cada palavra da frase e a classifica segundo sua classe (substantivo, verbo, pronome, etc), gênero (masculino, feminino ou comum), número (singular ou plural), e demais classificações necessárias conforme a classe da palavra. Este objeto é utilizado conjuntamente com o Analisador Sintático, fornecendo a ele a taxonomia das palavras da frase.

Taxonomia: neste objeto está armazenado todo o domínio de palavras compreendidas pelo protótipo, bem como, suas classificações (classe, gênero, número, etc). Para evitar a explosão de palavras o objeto possui regras de produção que ditam como é gerado o feminino/masculino, plural/singular, conjugações de verbos a partir de seus radicais.

```
Exemplos: radicalNome(filh)
          radicalVerbo(mostr)
          nome(casa,fem,sing)
          pronome(qual,mas,sin,interrogativo)
          conjunção(que)
          adjetivo(bom)
          ...
```

Analisador Semântico: este objeto tem um dos papéis fundamentais no funcionamento do modelo. É ele que distingue Processamento de Linguagem Natural (que envolve basicamente verificação de sintaxe) de Compreensão de Linguagem Natural (que implica em entender o sentido da frase). A este objeto cabe determinar qual o significado da frase digitada depois de "dissecada" pelo analisador sintático. Para tanto, faz uso de um dicionário, onde estão armazenadas as informações necessárias para a

"compreensão" das palavras e suas respectivas ações. Estas ações serão transformadas em comandos e enviadas ao simulador.

Exemplo: para uma árvore com a estrutura:
sv(mostre,sn(cf))
é gerado o comando: mostre(cf)

Dicionário: armazena todos os signos do modelo, ou seja, os comandos associados a cada regra da frase.

Exemplo: para o radical de verbo *mostr* está associado o comando *mostre* que pode ser executado pelo Simulador.

3.2. Implementação

3.2.1. Interface

O objeto Interface lê a frase digitada pelo usuário e faz um pré-processamento antes de enviá-la ao Analisador Sintático. Este pré-processamento consiste nas seguintes etapas:

- Converter a cadeia de caracteres lida em uma lista de palavras, pois nesta forma a frase pode ser melhor manipulada.

  Exemplo: a frase
  Qual o ciclo financeiro
  é convertido na lista:
  [qual,o,ciclo,financeiro]

- "Explodir" as locuções prepositivas: esta etapa substitui as locuções prepositivas da frase por seus constituintes (no caso: preposição + artigo), simplificando sua análise sintática.

  Exemplo: para a frase de entrada:

```
          [quem, está, na, casa]
será criada a frase:
          [quem, está, em, a, casa]
```

3.2.2. Analisador Léxico e Taxonomia

Na implementação do objeto Taxonomia foram criados os seguintes objetos, com seus respectivos atributos, aqui exemplificados resumidamente:

- Determinante(Artigo,Gênero,Numero,Definido/Indefinido)

```
Exemplo 3.1: deter('o'   ,m,s,def).
             deter('os'  ,m,p,def).
             deter('a'   ,f,s,def).
             deter('as'  ,f,p,def).
             deter('um'  ,m,s,indef).
             deter('uns' ,m,p,indef).
             deter('uma' ,f,s,indef).
             deter('umas',f,p,indef).
```

- Pronomes(Pronome,Gênero,Numero,Tipo)

```
Exemplo 3.2: pron('qual'   ,_,s,interrogativo).
             pron('quais'  ,_,p,interrogativo).
             pron('quanto' ,m,s,interrogativo).
             pron('quantos',m,p,interrogativo).
```

- Conjunções(Conjunção,Tipo)

```
Exemplo 3.3: conj('e'                ,aditiva).
             conj('mas'              ,adversativa).
             conj('porém'            ,adversativa).
             conj('pois'             ,conclusiva).
             conj('logo'             ,conclusiva).
             conj('que'              ,explicativa).
             conj('porque'           ,explicativa).
             conj('se'               ,condicional).
             conj('caso'             ,condicional).
```

- Preposições(Preposição)

```
Exemplo 3.4: prep('a').
             prep('ante').
             prep('com').
             prep('de').
             prep('em').
             prep('para').
```

- Adjetivos(Adjetivo,Gênero,Número): os adjetivos — masculino/feminino, plural e superlativo relativo — são gerados a partir de seus radicais, ou quando fogem à regra, são explicitamente declarados.

```
Exemplo 3.5: adj('pouco'  ,m).
             adj('bom'    ,m).
             adj('bons'   ,m,p).
             radj(muit).
             radj(pequen).
             radj(robust).
```

Algumas das regras que geram o gênero, número e superlativo são:

```
feminino  ⇒ radical + 'a'
        Ex.: muit + a = muita
masculino ⇒ radical + 'o'
        Ex.: muit + o = muito
plural palavras terminadas em vogal ⇒ acrescenta-se 's'
        Ex.: muito + s = muitos
...
Superlativo Relativo palavras terminadas em vogal
                ⇒ substitui-se a última letra por íssimo
                   muito + íssimo = muitíssimo
...
```

- Numeral(Valor Numérico, Extenso): onde existem regras que convertem um valor numérico para extenso e vice-versa.

```
Exemplo 3.6:
1.200.000 ⇒ 'um milhão e duzentos mil'
'um milhão e duzentos mil' ⇒ 1.200.000
```

- Substantivos(Substantivo,Gênero,Numero): da mesma maneira que ocorre com os adjetivos, os substantivos são gerados a partir do radical, salvo exceções. Os substantivos, para fins de implementação foram divididos em quatro categorias: radical, nome, nome próprio e nome composto.

```
Exemplo 3.7:radNome(filh)
            nome( 'cf'        , m).
            nome( 'pmr'       , m).
            nome( 'estoque'   , m).
            nome( 'imposto'   , m).
            nomeProprio( 'Estela',f).
            nomeComposto('cf' ,[ciclo,financeiro]).
```

```
nomeComposto('pmr',[prazo,médio,recebimento])
```

- Verbos(verbo, modo, tempo, conjugação, pessoa, número, radical, regência): para os verbos regulares são geradas, por meio de regras, todas as conjugações para seus radicais. No caso dos irregulares as conjugações devem ser definidas particularmente.

```
Exemplo 3.8: radicais de verbo (radical, conjugação,
regência, comando para o simulador)
  rVerbo(aument  ,1ª,trans,aumenteVl).
  rVerbo(mostr   ,1ª,trans,mostre).
  rVerbo(pass    ,1ª,trans,atualizeVl).
  rVerbo(atualiz ,1ª,trans,atualizeVl).
  rVerbo(acontec ,2ª,trans,mostre).
  rVerbo(diminu  ,3ª,trans,diminuaVl).

verbos irregulares:
  verboIrr( ser,indicativo, presente,
          [sou,'és','é', somos, sois,'são']).
```

3.2.3. Analisador Sintático e Gramática

Para Chomsky [CHOM65] a análise sintática tem um papel fundamental na compreensão da LN, isto tanto é verdade que a gramática transformacional está baseada na sintaxe (ver figura 2.5).

No protótipo implementado o Analisador Sintático tem a função de converter a frase digitada - e enviada pela Interface - em uma árvore de derivação, que será utilizada pelo Analisador Semântico. Para atingir este fim, ele interage com o Analisador Léxico e Gramática. Com o primeiro, para fornecer a classificação das palavras da frase, e com o segundo para verificar a função sintática desta palavra na frase.

Gramática contém em si a definição da gramática utilizada, conforme as definições do capítulo 2.2. Na sua especificação foram utilizadas as seguintes abreviaturas para o conjunto dos não-terminais (N):

```
F     = frase
```

```
S     = sentença
SN    = sintagma nominal
SC    = sintagma condicional
SV    = sintagma verbal
sPrep = sintagma preposicional
sComp = sintagma complementar
sAdj  = sintagma adjetivo
sNum  = sintagma numérico
G     = Gênero
N     = Número
```

A regras de gramática utilizadas (conjunto P →), as variáveis de contexto (entre []) e as regras para criação da árvore de derivação (←), são:

```
a) S     → F SC
         ← F se SC

b) S     → pronome interrogativo[G,N] F[G,N] SC
         ← q(F) se SC

c) S     → o que SV SC
         ← q(SV) se SC


d) F     → SN
         ← SN

e) F     → SN[N] SV[N]
         ← s(SN,SV)

f) F     → SV
         ← SV


g) SN    → artigo[G,N] nome[G,N] SComp
         ← sn(nome cple SComp)

h) SN    → nome SComp
         ← sn(nome cple SComp )

i) SN    → SNum
         ← SNum


j) SV    → verbo
         ← sv(verbo)
```

```
k) SV    → verbo[N] SN[N]
         ← sv(verbo,SN)

l) SV    → verbo[N] SPrep[N]
         ← sv(verbo,SPrep)


m) SC    → conjunção condicional S
         ← S

n) SC    → null
         ← null



o) SComp → SAdj
         ← SAdj

p) SComp → SPrep
         ← SPrep

q) SComp → null
         ← null


r) SPrep → preposição + SN[N]
         ← sPrep(SN)


s) SAdj  → adjetivo
         ← sAdj(adjetivo)

t) SAdj  → determinate + adjetivo
         ← sAdj(adjetivo)


u) SNum  → numeral
         ← num(numeral)

v) SNum  → numeral  unidade
         ← num(numeral,unidade)
```

O não-terminal inicial (S_O) é F.

Antes do continuar com o funcionamento do Analisador Sintático convém compreender melhor como funciona a máquina de inferência do PROLOG.

Máquina de Inferência

Para a implementação das regras de gramática utilizou-se a máquina de inferência do PROLOG que possui o funcionamento conforme descrito a seguir, exemplificado e abordado no que se refere às regras de sintaxe do protótipo.

A máquina de inferência tem por objetivo provar regras, tanto determinísticas (com uma hipótese de solução) como indeterminísticas (com n hipóteses de solução).

As regras são montadas com duas componentes: lado-esquerdo e lado-direito, separados por "→".

Ex.: SV → verbo (leia-se: um Sintagma Verbal é um verbo)

O lado-esquerdo da regra é o que se deseja provar, o lado-direito as condições para o lado-esquerdo ser verdadeiro. O lado-direito pode ser formado por dois tipos de membros: terminais e não-terminais. Os terminais (escritos em minúsculo) são fatos, que por sua própria natureza são verdadeiros. Os não-terminais (escritos em maiúsculo) são regras que novamente precisam ser provadas para satisfazerem a condição do lado-direito.

Ex.: F → S (leia-se: uma Frase é uma Sentença)

O lado-direito pode combinar elementos terminais e não-terminais, unindo-os com o sinal +. Desta forma todos os elementos devem ser provados para que a sentença seja provada.

Ex.: SN → artigo + nome (leia-se: um Sintagma Nominal é um artigo seguido de um nome)

Quando ocorre indeterminismo a máquina de inferência tenta provar uma das regras definidas, caso não consiga, tenta provar a outra, e assim sucessivamente até conseguir provar uma das regras, ou esgotarem as hipóteses. Neste caso a frase não pertence a linguagem definida pela gramática.

Ex.: S → SN (leia-se: uma Sentença é um SN ou
S → SV + SN SV seguido de SN)

ou seja, uma sentença pode tanto ser um SN como um SV+SN.

O indeterminismo aumenta consideravelmente o tempo que o Analisador Sintático dispende para analisar a frase.

Por exemplo, na gramática:
a) SN → nome + SPrep
b) SN → nome + SAdj
c) SN → nome

para provar SN, ou se prova a primeira regra, ou a segunda, ou ainda, a terceira. Quando tentar provar (a) se verificará se a palavra atual na frase é um nome, ou seja, pedirá ao Léxico para consultar a Taxonomia. Caso não se trate de um nome a regra (a) falha e a máquina de inferência tentará provar (b). Para provar (b) novamente será consultado o léxico, a procura de um "nome", realizando uma tarefa já feita. A mesma tarefa desnecessária acontece também na prova de (c).

Se a gramática acima for rescrita para:

a) SN → nome + SComp
b) SComp → SPrep
c) SComp → SAdj
d) SComp → null

que é equivalente a anterior, a máquina de inferência pesquisará por nome apenas uma vez. Assim, as regras da gramática devem ser convenientemente arranjadas para garantir a eficiência da inferência.

Outra otimização utilizada na gramática definida ocorre na seguinte parte:

a) F → S
b) F → pron. inter. + S
c) F → F + SC
d) SC→ conj. cond. + S

a regra (c) diz que uma frase pode ser uma frase seguida de um sintagma condicional. Como a gramática acima

definida tem problemas para ser implementada, devido a recursividade à esquerda da regra (c), ela deve rescrita para:

a) F → S
b) F → pron. inter. + S
c) F → S + SC
c') F → pron. inter. + S + SC
d) SC→ conj. cond. + S

onde, F do lado direito de (c) foi primeiramente substituído pela regra (a) -(S)- gerando um novo (c) e em seguida foi criada uma nova regra para a regra (b) (pron.inter.+S) gerando (c').

com esta transformação ocorreu o problema descrito acima (no caso do SN), pois F pode começar com S em duas regras. Este problema é resolvido de forma similar. Gerando a gramática:

a) F → S + SC
b) F → pron. inter. + S + SC
d) SC→ conj. cond. + S
e) SC→ null

Exemplo 3.9: a Análise sintática completa para a frase:

O que acontece com o cf se o estoque aumentar 10 %

Regra	N	Regra	Árvore
c	F	pron. inter. SV SC *o que*	sv(acontece,sPrep(sn(cf))) se s(sn(estoque), sv(aumentar,num(10,%)))
l	SV	verbo SPrep *acontece*	sv(acontece,sPrep(sn(cf)))
r	SPrep	preposição SN *com*	sPrep(sn(cf))
g	SN	artigo[G,N] nome[G,N] *o[m,s] cf[m,s]*	sn(cf)
m	SC	conj. condicional S *se*	
e	S	SN SV	s(sn(estoque), sv(aumentar,num(10,%))
g	SN	artigo[G,N] nome[G,N] *o[m,s] estoque[m,s]*	sn(estoque)
k	SV	verbo SN *aumentar*	sv(aumentar,num(10,%))

i	SN	SNum	num(10,%)
v	SNUm	numeral unidade *10* %	num(10,%)

Ou na forma de árvore:

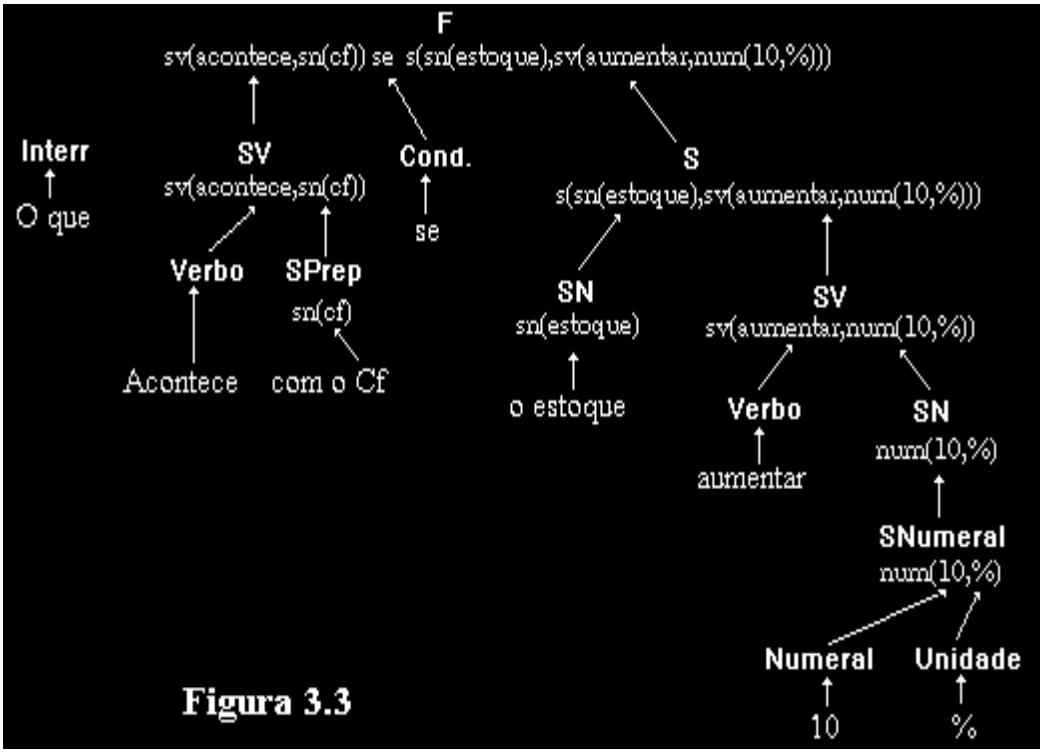


Figura 3.3

3.2.4. Analisador Semântico e Dicionário

Ao Analisador Semântico, juntamente com o Dicionário, cabe a tarefa de converter a árvore sintática enviada pelo Analisador Sintático num comando que possa ser executado pelo Simulador.

Para a realização desta tarefa faz-se uso dos conceitos da gramática transformacional. Assim, a árvore sintática é transformada até formar um comando. Para estas transformações são utilizadas regras armazenadas no Dicionário e que serão processadas pela máquina de inferência.

Exemplo 3.10

Algumas das regras armazenadas no Analisador Semântico:

a) q(sn(Nome)) ⇒ mostre(Nome).

esta regra significa: uma árvore no formato: q(sn(N)) - questão(sintagma_nominal(Nome)) - será transformada para o comando: mostre(N).

b) q(A) ⇒ C → transf(A,C)

leia-se: uma frase q(A) (onde A é outra árvore sintática qualquer) será transformada no comando C, somente se a árvore A puder ser transformada no comando C.
Por exemplo: se existe a regra de transformação:
trans(sv(é(sn(X))),mostre(X))
então a frase:
qual é o cf
com árvore:
A = q(sv(é,sn(cf)))
será transformada conforme:
q(sv(é,sn(cf))) ⇒ mostre(cf)
→ transf(sv(é,sn(cf)),mostre(cf))
onde X é cf.

c) Sa se Sb ⇒ C1 se C2 → transf(Sa,C1) e transf(Sb,C2).

esta regra deve ser lida da seguinte forma: Uma árvore na forma: SentençaA se SetençaB, será transformada para Comando1 se Comando2, somente se SentençaA for

transformada em Comando1 e SentençaB for transformada em Comando2.

d) s(sn(N),sv(V)) ⇒ C → transf(sv(V,sn(N)),C)

e) sv(V) ⇒ V.

f) sv(V,sn(N)) ⇒ V(N)

```
A frase                      : aumente DR
tem árvore                   : sv(aumente,sn(DR))
transformada pela regra (e)  : aumente(DR).

A frase                      : As DR aumentaram
tem árvore                   : s(sn(DR),sv(aumentar))
transformada pela regra (d)  : sv(aumentar,sn(DR))
transformada pela regra (e)  : aumente(DR).

A frase:
    Qual o cf se o estoque diminuir
tem árvore:
    q(sn(cf)) se s(sn(Estoque),sv(diminuir))
utilizando a regra (c), há duas sub-árvores que precisam
ser transformadas:
    1ª) q(sn(cf))
que é transformada pela regra (a) em: mostre(cf)
    2ª) s(sn(dr),sv(aumentar))
que é transformada por (d) em: sv(diminuir,sn(Estoque))
e pela regra (e) em: diminuir(Estoque).
voltando a regra (c) - que "chamou" as outras duas,
temos:
    mostre(cf) se diminuir(Estoque)
```

Deve ser observado que o Analisador Semântico deve ser congruente com o Analisador Sintático e com o Simulador, pois deve reconhecer todos os tipos de árvore enviada, bem como gerar comandos que o Simulador compreenda.

No dicionário estão armazenadas as ações que o simulador deve realizar quando determinada palavra (especialmente os verbos) é encontrada na frase.

```
Por exemplo: o verbo
  passar tem como ação: atualize o valor de X em Y
  atualizar tem como ação: atualize o valor de X em Y
  mostrar : mostre X
```

onde X e Y são outros termos da frase que o analisador sintático encontrou.

3.2.5. Simulador

O objeto Simulador tem como serviço executar o comando a ele enviado pelo Analisador Semântico e enviar o resultado da execução à Interface.

O objetivo do Simulador é informar ao usuário o que acontece com o **C**iclo **E**conómico **F**inanceiro (e outros indicadores) se as variáveis que o determinam forem alteradas.

A definição do Simulador, ou seja, as suposições, as fórmulas e as regras de implicação foram definidas por Alejandro Martins e Paulo de Tarso Mendes Luna. Estas informações caracterizam o Simulador como um sistema especialista onde o conhecimento de dois especialistas foi transformado em regras e fórmulas.

Suposições para o Simulador:

- A empresa simulada é um empresa comercial (uma loja ou super mercado);
- Por conseqüência não há produto em processo, assim:
  Estoque=Estoque de Produto Acabado;
- As vendas à vista levam dois dias para transformarem-se em dinheiro;
- O período considerado é de um mês e as variáveis relacionadas são dadas por mês. (ex.: vendas brutas no mes, etc);
- Não serão consideradas Despesas Antecipadas;
- O fornecimento de água e luz não é considerado;
- Considerou-se que a despesa operacional é formada por uma despesa fixa mais uma parte variável (definida neste caso como um percentual de 8% das vendas brutas);
- Considera-se que não há concorrência perfeita, e que os consumidores não são bem informados sobre os preços.

As variáveis manipulados pelo simulador são:

sigla	descrição Fórmula de cálculo outras variáveis que também devem ser alteradas se esta for.
pmr	prazo médio dos recebimentos ((pvp*prz)+((1-pvp)*2)) prz,dr,gdr
vb	vendas brutas (dr/pmr) dr,if,of,do,cmv,com,for,pme,ge
pvp	percentual de vendas a prazo pmr,dr,gdr
prz	prazo pmr,dr,gdr
dr	duplicatas a receber (vb*pmr) pmr,prz,gdr
gdr	giro de duplicatas a receber (30/pmr) pmr,prz,dr
mu	mark-up vb,dr,if,of,do
cmv	custo da mercadoria vendida ((1/(1+mu))*bv) com,for,pme,ge
e	estoque ((ei+ef)/2) ef,com,for,pme,ge
ei	estoque inicial e,pme,ge,com,for
ef	estoque final

```
          ((ei+com)-cmv)
          e,pme,ge,com,for

ge        giro dos estoques
          (30/pme)
          pme,e,ef,com,for

pi        percentual de imposto
          if,of

if        imposto faturado
          (pi*vb)
          pi,of

of        obrigações fiscais
          ((if/30)*pmi)
          pmi

pmi       prazo médio de recolhimento de impostos
          of

for       fornecedores
          ((com/30)*pmc)
          pmc

com       compras
          ((cmv-ei)+ef)
          ef,e,pme,ge,for

pmc       prazo médio de compras
          for

fp        folha de pagamento
          sal,enc

sal       salários
          ((fp/30)*pms)
          fp,enc

pms       prazo médio de pagamento de salário
          sal

penc      percentual de encargos
          enc
```

```
enc       encargos
          penc*((fp/30)*pmenc)
          pmenc


pmenc     prazo médio de pagamento de encargos
          enc

do        despesas operacionais
          ((dof+0.08)*vb)
          dof

dof       despesas operacionais fixas
          do

cf        ciclo financeiro

pme       prazo médio do estoque
          (e/cmv)
```

Ou seja, se o valor de pmr for alterado, também deve ser alterado o valor de prz (ou pvp, mas esta hipótese não será considerada), de dr e de gdr.

Se o valor de vb for alterado, deve-se alterar também o valor de: dr, if->of, do, cmv->com->for, cmv->pme->ge.

...

A partir destas variáveis são calculados indicadores que, por sua vez, irão determinar o **C**iclo **F**inanceiro (CF).

Algumas das variáveis acima poderão ser melhor compreendidas a partir da seguintes tabela:

Aspectos Financeiros	Aspectos Económicos	(Em dias de Venda)
Prazo médio dos Recebimentos (PMR) DR --- Vendas a Prazo	% Vendas a prazo Vendas a Prazo --- Vendas Brutas	= DR (+)
Prazo médio dos Estoques (PME) Estoque --- CPV ou CMV	Composição do Custo CPV ou CMV --- Vendas Britas	= Estoque (+)
Prazo médio de Recolhimento das Obrigações Fiscais Obrigações Fiscais --- Imposto Faturado	% Imposto Imposto Faturado --- Vendas Brutas	= Obrigações Fiscais (-)
Prazo médio de Compras Fornecedores --- Compras	% Compras Compras --- Vendas Brutas	= Fornecedores (-)
Prazo médio de pagamento de salários e Encargos Salários e Encargos --- CPV + Desp. Operac.	Composição dos Custos e Despesas CPV + Desp. Operac. --- Vendas Brutas	= Salários e Encargos Sociais (-)

Tabela 3.1

$$Cf = \Sigma \text{ 3ª coluna}$$

$$\text{Giro dos Estoques} = \frac{360}{PME}$$

$$\text{Giro das Dup. Receber} = \frac{360}{PMR}$$

Por exemplo, se as variáveis possuírem os valores:

Variável	**Valor**
Duplicatas a Receber	4608,00
Vendas a Prazo	3465,00
Vendas Brutas	6300,00
Estoque	2066,00
CPV	2645,00
Compras	1488,00
Fornecedores	1977,00
Obrigações Fiscais	1785,00
Imposto Faturado	1075,00
Salários e Encargos Sociais	729,00
Despesas Operacionais	3561,00

Teremos a tabela:

Aspectos Financeiros (dias)	Aspectos Económicos	(Em dias de Venda)
PMR 4608/3465 * 30 = 40	% Vendas a prazo 3465/6300 = 0,55	= 22 (+)
PME 2066/2645 * 30 = 23	Composição do Custo 2645/6300 = 0,42	= 10 (+)
PM Orig. Fisc. 1785/1075 * 30 = 50	% Imposto 1075/6300 = 0,17	= 9 (-)
PMC 1977/1488 * 30 = 40	% Compras 1488/6300 = 0,24	= 9 (-)
PM Salários 729/3561 *30 = 50	Comp.Cutos/Despesas 3561/6300 = 0,57	= 3 (-)

CF = 11 dias

Durante a utilização do protótipo o usuário pode alterar os valores das variáveis e pode acompanhar as conseqüencias destas alterações no Ciclo Económico Financeiro.

Por exemplo, para frase:

O que acontece com o ciclo financeiro se as duplicatas a receber aumentarem 40 %.

o Simulador irá executar os seguintes comandos (vindos do Analisador semântico):

guardar o valor de DR;
aumenta-lo em 40%;
calcular o CF;
mostrar o cf; e
atualizar DR com o valor original.

3.3. Exemplos

Algumas das frases reconhecidas:

- Passar as Duplicatas a receber para 40 meses
- As dr passaram para quarenta
- Mostre o valor o ciclo financeiro
- Qual o CF
- Qual é o CF
- Qual é o valor do CF
- Mostre o CF
- Qual é o PME
- Atualize o estoque
- Atualize o estoque em 1 mil
- Aumente o estoque em 30%
- Diminua as dr em 5%
- Qual o cf se as dr passarem para 400
- O que acontece com o pme se o estoque aumentar

Digamos que o usuário entre com a seguinte frase:

Qual o Ciclo Financeiro ?

A Interface inicialmente irá converter a frase numa lista e em seguida trocará as abreviaturas:

```
[qual,o,ciclo,finenceiro]
[qual,o,cf]
```

esta lista será enviada ao Analisador Sintático. Este, por sua vez, irá consultar a gramática definida (ver item 3.23) que tem F (**F**rase) como regra inicial (S_O). F pode assumir três formas:

```
a) F     → S SC
b) F     → pronome interrogativo[G,N] S[G,N] SC
c) F     → o que SV SC
m) SC    → conjunção condicional S
n) SC    → null
```

Primeiramente o Analisador Sintático, através de sua máquina de inferência, tentará provar F com a regra (a), para tanto, precisa provar S, que é definida por:

```
d) S     → SN
e) S     → SN[N] SV[N]
f) S     → SV
```

Como a primeira palavra da frase não "casa" com nenhuma destas regras, a tentativa de provar S falha, e por conseguinte, falha (a). Tenta-se, então, a regra (b). Esta regra começa com um pronome interrogativo, como a frase também começa com pronome interrogativo, (b) tem chance, mas deve ser seguido de S e SC. SC é opcional, pois pode ser null. S é definida conforme (d,e,f), tenta-se a primeira hipótese de S:(d). (d) deve ser um SN:

```
g) SN    → artigo[G,N] nome[G,N] SComp
h) SN    → nome SComp
i) SN    → SNum
o) SComp → SAdj
p) SComp → SPrep
q) SComp → null
```

Para provar SN, pela regra (g), a frase deve ter um artigo e um nome, ambos concordando em gênero e numero; Scomp é opcional, pois pode ser null. Como o restante da frase ("o cf") é justamente um artigo e um nome a regra (g) é selecionada, conseqüentemente (d) e (b) também são

selecionadas, e a frase é aceita como pertencendo à língua.

Por fim, temos a seguinte árvore sintática para a frase:

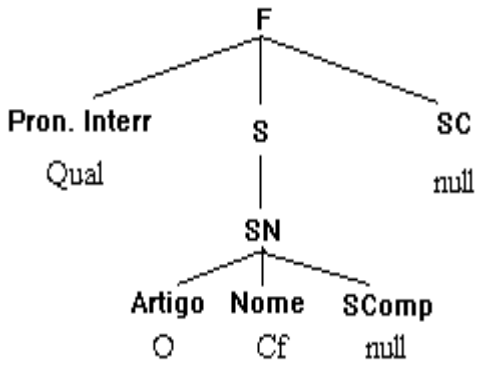


Figura 3.4

A representação clausal é montada a partir das regras utilizadas (b,d,g), onde cada regra cria uma parte da cláusula.

```
b) F    → pronome interrogativo[G,N] S[G,N] SC
        ← q(S) se SC
d) S    → SN
        ← SN
g) SN   → artigo[G,N] nome[G,N] SComp
        ← sn(nome cple SComp)
```

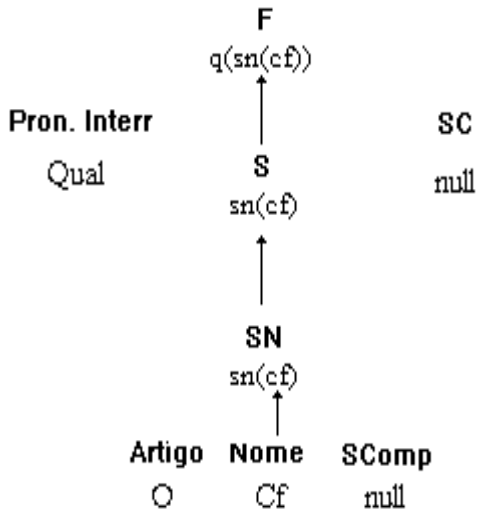


Figura 3.5

Esta cláusula (*q(sn(cf))*) é então enviada ao Analisador Semântico para montar o comando. Que utiliza a regra (a) de exemplo 3.10.

a) q(sn(Nome)) ⇒ mostre(Nome).

criando o comando:

mostre(cf)

O Simulador, ao receber este comando irá executa-lo, pegando o valor de cf e enviando-o à Interface para que o usuário veja.

4. Conclusão

Concluíu-se com este trabalho, e principalmente com o resultado da implementação, que é possível dar aos sistemas interfaces na linguagem dos seus usuários. Embora deva ser ressaltado que a experiência foi realizada em um domínio específico e com um vocabulário restrito, e que uma interpretação genérica - livre de domínio - não implica somente na ampliação do dicionário, neste caso, surgem outros problemas, principalmente de ordem semântica.

Futuras implementações devem ser mais modulares e genéricas, permitindo que o dicionário e as regras de sintaxe possam ser alterados pelo próprio usuário. Tornando assim, o protótipo ainda mais próximo da linguagem do usuário.

Convém ressaltar que a interdisciplinariedade, principalmente na área de linguística, é uma pré-requisito para que bons trabalhos possam ser realizados em CLN.

Apesar das dificuldades ficou evidente que o desenvolvimento de interfaces em LN para domínios restritos é perfeitamente viável é muito útil.

5. Bibliografia

[ARAR99] ARARIBÓIA, G. Inteligência artificial. Rio de Janeiro: LTC, 1988.

[BRAT90] BRATKO, Ivan. PROLOG: programing for artificial intelligence. 2. ed. [S.l.]: Addison-Wesley, 1990.

[CÂMA91] CÂMARA, Luiz Eduardo Saraiva e FONSECA, Décio. Um sistema de interface cooperativo para modelagem em banco de dados. In: Simpósio Brasileiro de Banco de Dados, Manaus, Anais..., mai., 1991.

[CAUD92] CAUDILL, Maureen. Kinder, gentler computing. BYTE, Natural I/O. 17(4):135-50, abr. 1992.

[CARR87] CARRAHER, David William, MUTCHNIK, Valdyr e ALBUQUERQUE, Cid C. A análise morfológica e sintática do português. In: Simpósio Brasileiro de Inteligência Artificial, Anais..., 1987.

[CARV82] CARVALHO, Castelar de. Para compreender Saussure: fundamento e visão crítica. 3° ed. rev. aum. Rio de Janeiro: Rio, 1982. 192 p.; 21cm.

[CASA87] CASANOVA, Marco A., GIORNO, Fernando A.C. e FURTADO, Antonio L. Programação em lógica e a linguagem PROLOG. São Paulo: Edgard Blücher, 1987.

[CHOM65] CHOMSKY, Noam. Aspects of the theory of syntax. Cambridge : MIT, 1965.

[CHOM71] CHOMSKY, Noam. Linguagem e pensamento. 2°ed. Petrópolis: Vozes, 1971.

[COAD92] COAD, Peter e YOURDON, Edward. Análise baseada em objetos. Rio de Janeiro: Campus, 1992.

[CUNH92] CUNHA, Celso Ferreira da. Gramática da língua portuguesa. 12° ed, 3ª tir. Rio de Janeiro: FAE, 1992. 655 p.; 24 cm.

[SHAP87] SHAPIRO, Suant C. editor. Enciclopedia of artificial intelligence. New York: John Wiley & Sons, 1987.

[FEIG81] FEIGENBAUM, Edward e BARR, Avrom. The handbook of artificial intelligence. [s.l.]:Heuristech Press, 1981.

[HOUA91] HOUAISS, Antonio. O que é língua. 2° ed. São Paulo: Brasiliense, 1991. Coleção Primeiros Passos, n° 239.

[KRUL91] KRULEE, K. Gilbert. Computer processing of natural language. New Jersey: Prentice-Hall, 1991.

[MONT91] MONTENEGRO, Fernando Borges. Uma interface de linguagem natural para um sistema de administração de capital de giro. Dissertação de Mestrado. Florianópolis, 1991.

[ORLA90] ORLANDI, Eni Pulcinelli. O que é lingüística. 4° ed. São Paulo: Ed. Brasiliense, 1990. Coleção Primeiros Passos, n° 184.

[SAVA87] SAVADOVSKY, Pedro. Introdução ao projeto de interfaces em linguagem natural. São Paulo: SID Informática, 1987.

[TOWN90] TOWNSEND, Carl. Técnicas avançadas em turbo PROLOG. Rio de Janeiro: Campus, 1990.

Assinaturas

_____

Jomi Fred Hübner
(Aluno)

_____

Paulo de Tarso M. Luna
(Orientador)

6. Anexos

- Exemplo de interação e telas
- Alguns fontes do protótipo