Uma arquitetura para o componente de tradução automática do Sistema Falibras, tradutor do Português para Língua de Sinais

Marcelo Nunes Ribeiro[1], Luis Cláudius Coradine[1], João Roberto Santos Júnior[1], Ivo Augusto Andrade Rocha Calado[1], Orivaldo de Lira Tavares[2], Fabio Cunha de Albuquerque[1]

[1] Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Campus A. C. Simões, Rod. BR 104 - N, Km 97, Tabuleiro dos Martins - Maceió - AL, CEP 57072-970. Fone 82 - 32141401
[2] Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Campus de Goiabeiras, Av. Fernando Ferrari s/n, Goiabeiras –Vitória – ES, CEP 29060-970. Fone 27 - 33352654

lccoral@gmail.com coradine@tci.ufal.br tavares@inf.ufes.br mnr@tci.ufal.br

Resumo. *Este artigo apresenta o componente de tradução automática, para um tradutor do português oral para a língua de sinais (Libras) na forma visual, usando o método de transferência sintática. Sua arquitetura é discutida, onde o analisador léxico possui um módulo de eliminação de ambigüidade utilizando classificação bayesiana e o analisador sintático usa o algoritmo de Earley. A tradução por transferência sintática baseia-se em uma adaptação de uma gramática de adjunção de árvores.*

Abstract. *This article presents the component of automatic translation, for a translator of the verbal Portuguese for the Brazilian language of signals in the visual form, implemented using the method of syntactic transference. The architecture is argued, where the lexical analyzer has a module of ambiguity elimination using a naïve Bayes classifier and the syntactic analyzer uses the Earley's algorithm. The translation for syntactic transference is based on an adaptation of the Tree Adjoining Grammars.*

Palavras-Chave. *Tradução automática, transferência sintática, ambigüidade léxica*

1. Introdução

A tradução automatizada de textos envolve problemas complexos e tem se constituído em assunto de abrangentes estudos pela comunidade científica [Russell and Norvig 2004]. Dentre esses problemas destacam-se a ambigüidade [Gale et al. 1992] e a análise sintática [Earley 1970].

A tradução do português para a língua brasileira de sinais (Libras) pressupõe algumas facilidades, nesse contexto, posto ser Libras uma língua com poucas classes gramaticais. Alguns trabalhos têm sido desenvolvidos nesse sentido, destacando-se o pioneirismo do Falibras [Coradine 2006] e [Tavares 2005] e o Tlibras [Martins 2005].

Este artigo apresenta uma arquitetura modular de um tradutor automático do português para Libras, onde os analisadores léxico e sintático são implementados como componentes de *software.* As etapas desse processo são apresentadas na Figura 1.

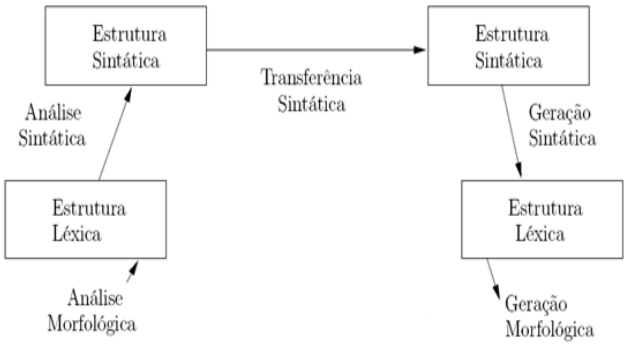

Figura 1. Arquitetura de tradução

O processo de tradução de textos inicia com uma Análise Léxico-Morfológica do texto de entrada. O resultado dessa etapa é a lista de itens léxicos do texto com os seus respectivos atributos léxicos. A etapa da Análise Sintática processa os itens léxicos para montar as estruturas sintáticas que compõem o texto de entrada. A partir dessas estruturas sintáticas, pelo método de transferência sintática, o tradutor gera as estruturas sintáticas equivalentes do texto traduzido. As últimas etapas da tradução são as gerações das estruturas léxicas e morfológicas constituintes do texto alvo.

A seção 2 apresenta o processo de eliminação da ambigüidade, a partir da lista dos itens léxicos [Gale et al. 1992]. O analisador sintático que implementa o algoritmo de Earley [Earley 1970] para gerar uma árvore de derivação na língua fonte é mostrado na seção 3. Na seção 4, é exibida uma adaptação da gramática de adjunção de árvores [Vijay-Shanker et al. 1986], aplicada à transferência sintática, segundo uma aproximação da idéia de LTAGs (*Lexicalized Tree Adjoining Grammars*) proposta em [Abeillé et al. 1990]. As considerações finais são apresentadas na seção 5.

A implementação do tradutor apresentado foi feita em C++ e está disponível no endereço <http://www.falibras.ic.ufal.br>.

2. Tratamento da Ambigüidade

No trabalho desenvolvido foi tratado o problema de eliminação de ambigüidade dentro do componente de análise léxica, o qual identifica as propriedades lexicais de cada palavra da frase. O componente de eliminação de ambigüidade utiliza um método de aprendizagem supervisionada por classificação bayesiana [Gale et al. 1992] e atua na lista de itens lexicais identificando o sentido de cada palavra do texto fonte.

O processo de eliminação de ambigüidade tem como meta determinar qual o sentido de uma palavra ambígua numa determinada frase e num determinado contexto. Essa escolha, também conhecida como *tagging*, é feita pela busca do sentido mais provável da palavra ambígua nas condições anteriores. Pode-se expressar a busca do sentido na forma:

$$arg\ max_{sentido}\ Pr(sentido \mid palavras, situacao)$$

Ou seja, deve-se determinar o sentido por meio das palavras que acompanham a palavra ambígua, dada uma certa situação. As palavras e a situação formam o contexto da frase. Quanto à representação de sentido, pode variar desde uma representação com traços sintáticos da palavra e sua tradução associada, até formas mais trabalhadas de representação do conhecimento.

Dessa forma, foi implementada a proposta de um modelo probabilístico para tratar a ambigüidade com base na classificação bayesiana [Gale et al. 1992]. O treinamento deve ser realizado sobre um *corpus* (grande quantidade de texto), onde toda palavra ambígua tem seu sentido correto apontado, caracterizando um problema de aprendizagem supervisionada. Assim, definindo-se [Manning and Schütze 1999]:

- w, uma palavra ambígua;
- $s_1,\ldots, s_k$, sentidos de w;
- $c_1,\ldots,c_i$, contextos de w em um *corpus*;
- $v_1,\ldots,v_j$, palavras usadas como formas contextuais para eliminação de ambigüidade.

A escolha ótima do sentido *s'* é feita como a seguir:

$$s' = arg\,max_{s_k}\ Pr(s_k|c)$$

$$s' = arg\,max_{s_k}\ \frac{Pr(c|s_k)}{Pr(c)}\ Pr(s_k)$$

$$s' = arg\,max_{s_k}\ Pr(c|s_k)\ Pr(s_k)$$

considerando a regra de Bayes e o fato de $Pr(c)$ é ser constante para todos os sentidos.

Para se determinar $Pr(c \mid s_k)$ foi tomada a suposição *naïve* de Bayes, onde os atributos usados para descrição são considerados independentes. Além disso, considerando que v_j, representa uma palavra inserida num contexto c, pode-se escrever:

$$Pr(c|s_k) = Pr(\{v_j|v_j \in c\} \mid s_k) = \prod_{v_j \in c} Pr(v_j|s_k)$$

onde, se ajustando para diminuição do custo computacional, implica em:

$$s' = arg\,max_{s_k}\left[\log Pr(s_k) + \sum_{v_j \in c} \log Pr(v_j|s_k)\right]$$

O cálculo das probabilidades é feito por Estimação de Máxima Verossimilhança [Gale et al. 1992], de forma que:

$$Pr(v_j|s_k) = \frac{C(v_j, s_k)}{\sum_t C(v_t, s_k)}$$

$$Pr(s_k) = \frac{C(s_k)}{C(w)}$$

sendo:

- $C(v_j\ ,\ s_k)$, número de ocorrências de v_j em um contexto de sentido s_k no *corpus* de treinamento;
- $C(s_k)$, número de ocorrências de s_k no *corpus* de treinamento;
- $C(w)$, número total de ocorrências da palavra w.

Os Algoritmos I e II mostram o treinamento e a eliminação de ambigüidade.

Algoritmo I Algoritmo de treinamento para eliminação de ambigüidade

for todos os sentidos s_k de w **do**
  for todas as palavras v_j no vocabulário **do**

$$Pr(v_j|s_k) = \frac{C(v_j, s_k)}{\sum_t C(v_t, s_k)}$$

  end for
end for
for todos os sentidos s_k de w **do**
  for todas as palavras v_j no vocabulário **do**

$$Pr(s_k) = \frac{C(s_k)}{C(w)}$$

  end for
end for

Algoritmo II Algoritmo para eliminação de ambigüidade

for todos os sentidos s_k de w **do**
  score(s_k) = log Pr(s_k)
  for todas as palavras na janela de contexto c **do**
    score(s_k) = score(s_k) + log Pr(v_j | s_k)
  end for
end for
escolha s' = arg $\max_{sk}$ score(s_k)

2.1 O componente de eliminação de ambigüidade implementado

O componente de eliminação de ambigüidade do tradutor é formado por três classes que abstraem os conceitos de palavra, frase e *corpus* que implementam o algoritmo. O diagrama de classes dessa estrutura é apresentado na Figura 2.

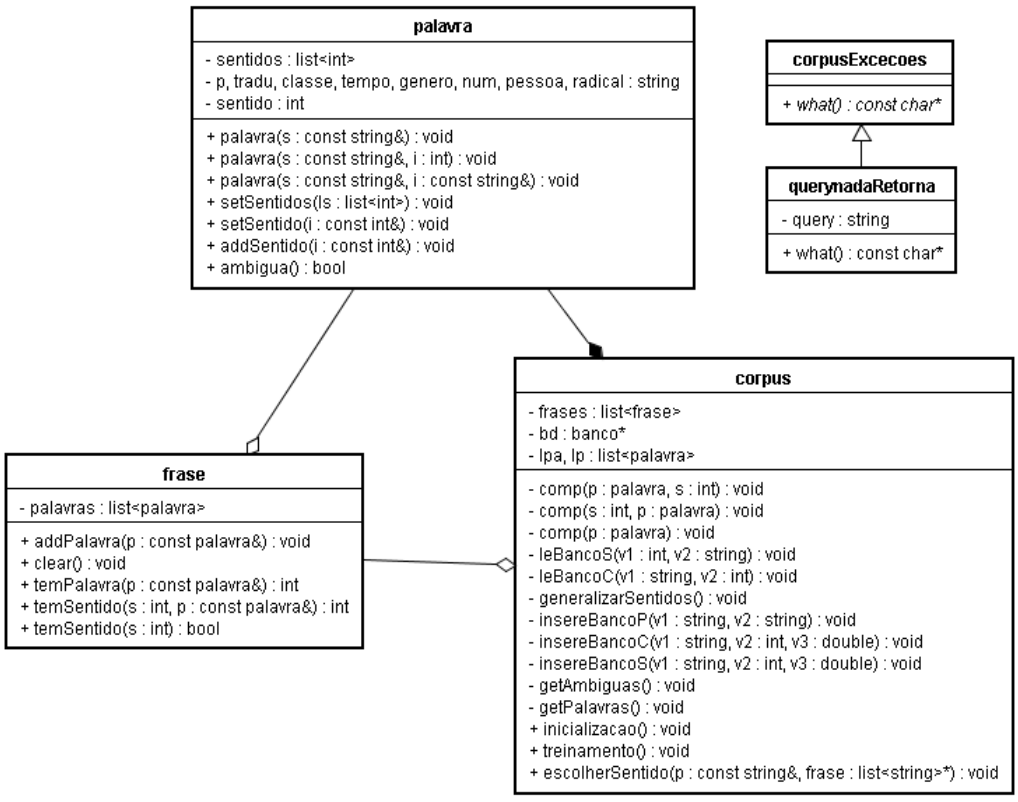

Figura 2. Diagrama de classes do componente de eliminação de ambigüidade

A classe **palavra** possui como atributos uma lista de todos os sentidos possíveis (representados por inteiros que são a chave primária da tabela de sentidos do banco), um inteiro que referencia o sentido da palavra e uma seqüência de *strings* contendo palavra e informação léxica da mesma. Fornece ainda uma seqüência de métodos triviais.

A classe **frase** guarda uma lista de **palavra**, com operações, tais como:

- **temPalavra:** Retorna um inteiro resultante da contagem das ocorrências da palavra **p** na frase.
- **temSentido:** A sobrecarga do método com dois parâmetros conta o número de ocorrências da palavra p com sentido s na frase.

A classe **corpus** guarda toda a complexidade do componente. Possui uma lista de **frase** (que representa o *corpus*), um ponteiro para conexão com o banco de dados e duas listas de **palavra** que guardam as palavras que ocorrem no *corpus*: **lpa** que possui todas as palavras ambíguas e **lp** que possui todas as palavras.

Numa breve descrição dos métodos, desde a inicialização até a escolha do sentido, pode-se verificar que a análise léxica é simplificada a um percurso de frase.

3. Análise sintática

O algoritmo de Earley [Earley 1970] apresenta uma forma de análise sintática eficiente que mistura os métodos das análises *top-down* e *bottom-up*. Ele não é exatamente um analisador, uma vez que não provê a construção de árvores de análise, mas um reconhecedor. Ou seja, para uma dada cadeia de entrada, o algoritmo diz se ela pode ser ou não gerada pela gramática. Dentre suas características estão que:

- o funcionamento se dá sobre qualquer gramática livre de contexto;
- ele é executado em um tempo de $O(n^3)$, no pior caso;
- trata recursão à esquerda, naturalmente;
- tem comportamento *LR* (*Left Most*), ou seja, estende, preferencialmente, os símbolos não-terminais mais à esquerda de uma cadeia.

Todo o trabalho do algoritmo é feito por meio de conjuntos de estados. Os estados são entidades que representam possíveis gerações para a cadeia de entrada. Por exemplo, o estado:

$$S \rightarrow SN \bullet SV \quad [0]$$

significa que o algoritmo encontrou um *SN* e se encontrar um *SV*, formará um *S*. O símbolo • (ponto) indica até onde a cadeia foi analisada e o número entre colchetes designa o conjunto onde o respectivo estado foi criado. Enquanto cada *token* da cadeia de entrada é analisado, um conjunto de estados deverá ser criado. Se uma cadeia $x_1x_2...x_n$, é analisada, tem-se, portanto, $n+1$ conjuntos de estados.

O algoritmo baseia-se em três métodos: *predict*, *scanner* e o *completer*. Esses métodos tratam exclusivamente de um estado, de acordo com sua configuração. A seguir, estão descritas suas características e funções:

- **predict:** tem comportamento *top-down*. É executado somente quando o estado que está sendo analisado tem o **ponto** à esquerda de um não-terminal;
- **scanner:** tem comportamento *bottom-up*. Deve ser executado somente quando o estado corrente tiver o **ponto** à esquerda de um terminal;

- **completer:** possui características *bottom-up*. Deve ser chamado somente quando o estado corrente estiver com o ponto no final da cadeia.

A implementação do algoritmo foi feita totalmente na linguagem de programação C++, com o auxílio das bibliotecas *boost* (ver www.boost.org), mais especificamente, com o uso a classe contêiner *tokenizer.hpp*.

A gramática que o sistema trabalha é armazenada em um arquivo de configuração. Isso dá mais flexibilidade ao sistema, pois, se houver necessidade de alteração, basta apenas modificar o arquivo e não recompilar todo o sistema. Foi usada para teste do tradutor uma gramática proposta por [Ralha et al. 2004].

A Figura 3 apresenta o diagrama UML (*Unified Modeling Languange*) das quatro classes que implementam o algoritmo do analisador.

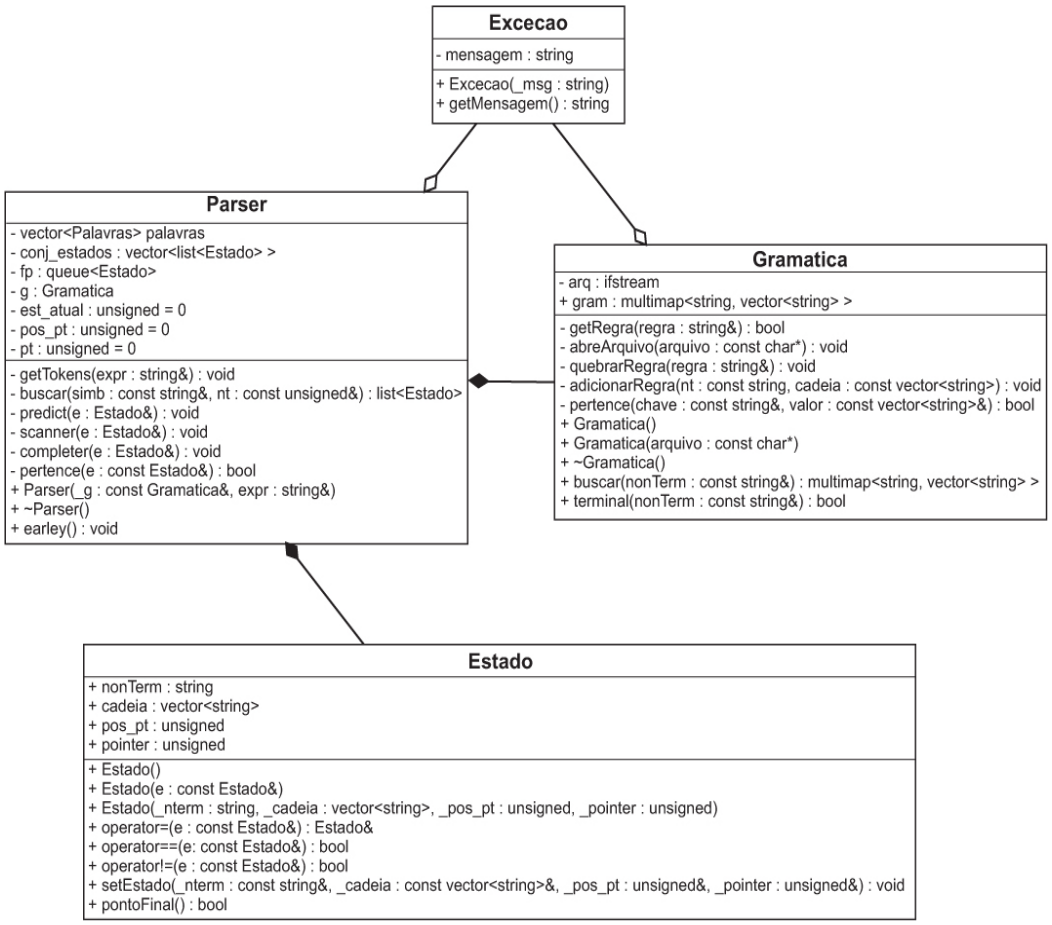

Figura 3. Diagrama de classes do analisador sintático

A classe **Excecao** trata todas as ocorrências de erros ou eventos inesperados em tempo de execução do sistema.

A classe **Gramatica** define o objeto que abstrai a gramática dos idiomas.

A classe **Estado** abstrai os estados descritos no algoritmo de Earley. Possui métodos que, por exemplo, verificam se o ponto está no final da cadeia de um determinado estado (**pontoFinal**) ou se dois estados são iguais ou não.

A classe **Parser** é a principal do módulo. Os métodos *predict*, *scanner* e *completer* executam exatamente as três operações básicas do algoritmo.

4. Transferência sintática

4.1. Gramática de Adjunção de Árvores (TAG)

Para um melhor entendimento do processo de transferência sintática do tradutor (português-libras), é necessário uma introdução às TAGs (*Tree Adjoining Grammars*) [Vijay-Shanker et al. 1986]. Uma variante desse formalismo, as *Synchronous TAGs*) [Shieber and Schabes 1990], é de extrema importância para o sistema desenvolvido, pois a transferência é atualmente realizada a partir de uma aproximação do formalismo.

A TAG é uma quíntupla *G = (Vn, Vt, S, I, A)*, [Vijay-Shanker et al. 1986], onde:

- V_n é um conjunto finito de não-terminais;
- V_t é um conjunto finito de terminais;
- *S* é o símbolo inicial;
- *I* é um conjunto finito de árvores iniciais;
- *A* é um conjunto finito de árvores auxiliares.

Uma árvore inicial, como mostrada na Figura 4 [Vijay-Shanker et al. 1986], é aquela em que o não-terminal no topo é um nó de **substituição**. Ou seja, pode ser anexada a uma árvore auxiliar através de uma operação de substituição. A árvore auxiliar tem um nó raiz marcado com um não-terminal X e um nó filho também marcado com X, onde se define a operação de **adjunção**. Os nós filhos também podem conter símbolos terminais, como mostrado na Figura 5 [Vijay-Shanker et al. 1986].

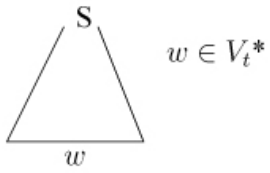

Figura 4. Árvore inicial TAG

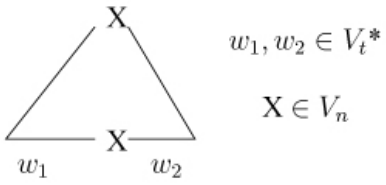

Figura 5. Árvore auxiliar TAG

A operação de **adjunção** é dada da maneira mostrada na Figura 6 [Vijay-Shanker et al. 1986], onde η é um nó marcado com X na árvore γ. Sendo β uma árvore auxiliar, a árvore obtida por adjunção de β em η é a árvore γ'.

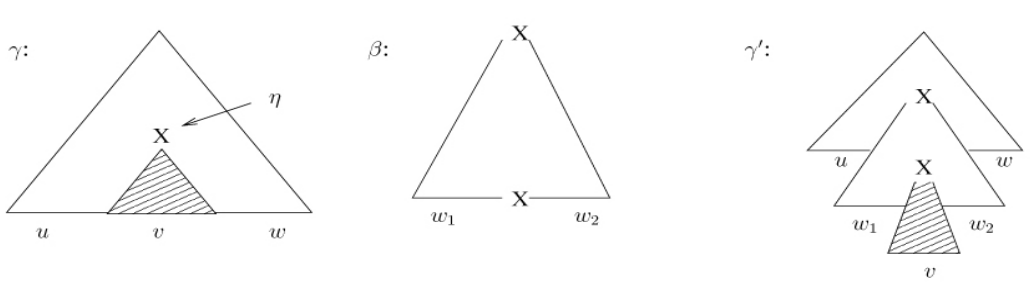

Figura 6. Adjunção em TAGs

Synchronous TAG [Shieber and Schabes 1990] é uma variante das TAGs que especifica correspondências entre linguagens, tomando um par de árvores, uma para a língua fonte, outra para a língua destino, onde nós de uma podem ser ligados a nós de outra. A derivação em uma *Synchronous TAG* inicia-se escolhendo um par de árvores (α_1, α_2) de uma TAG e seguem-se os passos:

1. Escolher uma ligação conectando dois nós no par de árvores (η_1 em α_1 e η_2 em α_2);
2. Escolher um par de árvores (β_1, β_2);
3. Construir o par resultante $<\beta_1(\alpha_1, \eta_1), \beta_2(\alpha_2, \eta_2)>$, onde $\beta(\alpha, \eta)$ é o resultado da operação de substituição ou adjunção de β em α no nó η.

Synchronous TAG mostra-se um formalismo atrativo para transferência sintática. Para tanto, é encarada como um **léxico de transferência**, como mostrada na Figura 7 adaptada de [Abeillé et al. 1990], no contexto do tradutor do Português para Libras. O processo de transferência é a própria derivação e as **regras de transferência** são as ligações entre as árvores num par. O processo de transferência é ilustrado na Figura 8 [Abeillé et al. 1990].

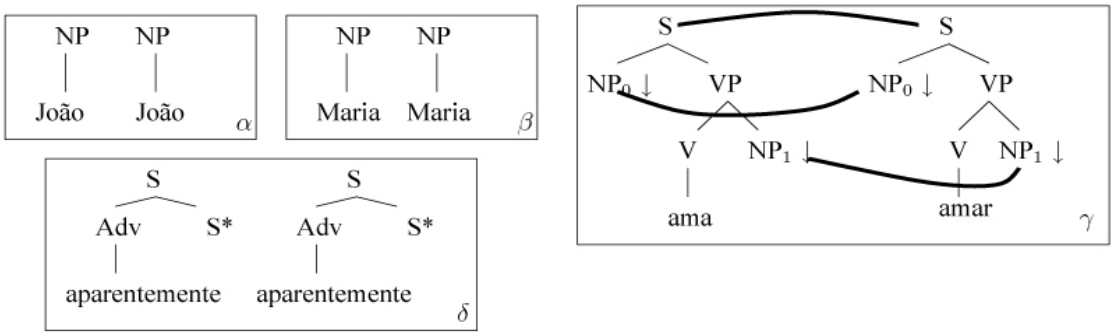

Figura 7. Um léxico de transferência

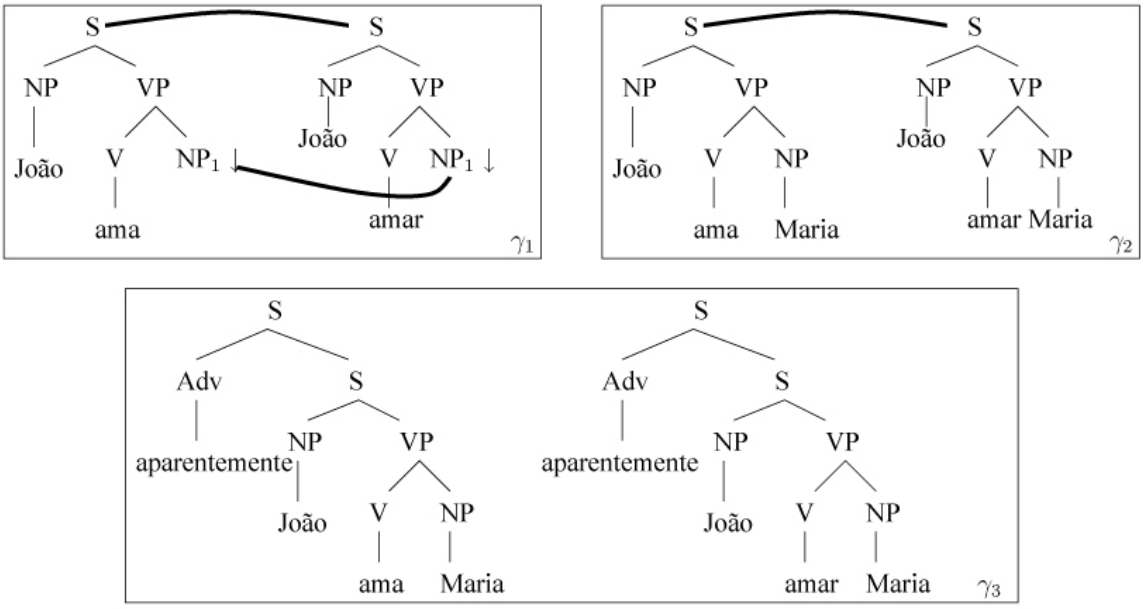

Figura 8. Derivação do léxico de transferência

A derivação utilizando o léxico de transferência ocorre da mesma maneira que a derivação com *Synchronous TAGs*. No exemplo da Figura 8, o par α opera na ligação NP_0 - NP_0 em γ, gerando γ_1. Depois, o par β opera na ligação NP_1 - NP_1 de γ_1, gerando γ_2. Por último, o par δ opera na ligação S-S em γ_2, gerando γ_3. Note que a última operação foi uma adjunção, não uma substituição. Percorrendo os terminais da árvore à direita em γ_3, tem-se a tradução em Libras "Aparentemente João amar Maria".

A transferência trata muitos aspectos lingüísticos da geração, como seleção léxica e seleção da estrutura sintática (que é o objetivo da derivação utilizando *Synchronous TAGs*). A atividade de geração com o uso de *Synchronous TAGs* somente lê os símbolos terminais da árvore, determinando a flexão correta da palavra baseada em informação léxica contida numa LTAG.

4.2 Transferência sintática no tradutor

A transferência sintática atualmente implementada no sistema se constitui de uma aproximação da idéia utilizada em *Synchronous TAGs*. Na análise sintática, a árvore de análise é "quebrada" em componentes triviais (tais como os sintagmas nominais) e uma árvore principal derivada da árvore original é escolhida, geralmente a que possui o verbo. A árvore principal equivale ao par de árvores, onde ocorrem as derivações em *Synchronous TAGs*, sendo que aqui não é definida uma ligação entre as árvores (onde poderiam ocorrer operações de substituição e adjunção) por não existir uma árvore destino.

Continuando a transferência, o ponto chave dessa aproximação é a criação de um identificador para a árvore principal. O identificador é uma *string* contendo uma representação linear da árvore principal. Este identificador é utilizado para a escolha do método (regra) de transferência. O método de transferência deve então:

1. Construir a árvore destino de maneira a obedecer a gramática da língua destino, rearranjando a árvore principal;
2. Realizar a operação de substituição nos nós da árvore principal com os componentes triviais gerados na análise.

Com isso, a geração da tradução se torna um simples percurso das folhas da árvore destino. Estas folhas devem possuir bastante informação léxica para se efetuar a geração. Uma possível atividade da geração é a fixação verbal, que necessária em Libras. Essa tarefa é realizada dados tempo, modo, pessoa e número do verbo.

Podem-se identificar duas dificuldades inerentes à transferência, através da arquitetura aqui proposta:

- Forte dependência entre a análise e transferência. Uma implementação baseada fortemente em *Synchronous TAGs* ajudaria a simplificar essa dependência, já que a atividade de derivação em *Synchronous TAGs* é a própria análise e transferência ocorrendo simultaneamente.
- O número de regras de transferência é enorme.

5. Conclusões e trabalhos futuros

No projeto do tradutor apresentado, já foram implementados os métodos aqui expostos de eliminação de ambigüidade, análise e transferência sintática. Das contribuições deste trabalho destacam-se os componentes C++ para eliminação de ambigüidade para análise sintática.

No prosseguimento deste projeto se deverá trabalhar na contextualização do que foi produzido até agora para o domínio de tradução Português-Libras, ou seja, produzir um *corpus* de treinamento para o componente de eliminação de ambigüidade, construir um léxico bilíngüe Português-Libras, uma gramática da língua portuguesa e identificar regras de transferência sintática para Libras. Pretende-se realizar uma implementação completa das *Synchronous TAGs* e utilizar métodos estatísticos de aquisição léxica propostos em [Manning and Schütze 1999] para alimentar as bases de conhecimento.

6. Referências

Abeillé, A., Schabes, Y., and Joshi, A. K. (1990). **Using Lexicalized TAGs for machine translation**. In Proceedings of the 13th conference on Computational Linguistics, pages 1–6, Morristown, NJ, USA. Association for Computational Linguistics.

Coradine, L. C.; Tavares, O. L.; Albuquerque, F. C.; Breda, W.L.; Silva, A.F.; Silva, C.B.; Ribeiro, M.N. (2006). **Interfaces Tradutoras do Sistema Falibras.** In: IV Congresso Ibero-Americano Sobre Tecnologias de Apoio a Portadores de Deficiência (Iberdiscap 2006). Vitória, ES, de 20 a 22 de fevereiro de 2006. Anais do IV Congresso Ibero-Americano Sobre Tecnologias de Apoio a Portadores de Deficiência (Iberdiscap 2006), pp., fev./2006. ISBN 84-96023-45-1.

Earley, J. (1970). **An efficient context-free parsing algorithm**. Commun. ACM, 13(2):94–102.

Gale, W., Church, K. W., and Yarowsky, D. (1992). **A method for disambiguating word senses in a large corpus**. Computers and the Humanities, 26:415–439.

Manning, C. D. and Schütze, H. (1999). **Foundations of Statistical Natural Language Processing**. The MIT Press, Cambridge.

Martins, R. T. ; Pelizzoni, J. M. ; Hasegawa, R. (2005). **Para um Sistema de Tradução Semi-Automática Português-Libras.** In: III Workshop em Tecnologia da Informação e da Linguagem Humana – TIL 2005 (XXV Congresso da Sociedade Brasileira de Computação). São Leopoldo, RS, de 25 à 29 de julho de 2005. Anais do XXV Congresso da Sociedade Brasileira de Computação – SBC 2005, jul./2005.

Ralha, J., Ralha, C., de Araújo, J., and da Silva Melônio, C. (2004). **Uma proposta para o desenvolvimento de um interpretador para a língua brasileira de sinais**. In Anais do XV Simpósio Brasileiro de Informática na Educação, pages 671–673, Manaus, AM. UFAM.

Russell, S. and Norvig, P. (2004). **Inteligência Artificial.** Elsevier, Rio de Janeiro, 2 edition.

Shieber, S. M. and Schabes, Y. (1990). **Synchronous tree-adjoining grammars**. In Proceedings of the 13th conference on Computational linguistics, pages 253–258, Morristown, NJ, USA. Association for Computational Linguistics.

Tavares, O. L.; Coradine, L. C.; Breda, W. L. (2005). **Falibras-MT – Autoria de tradutores automáticos de textos do português para Libras, na forma gestual animada: Uma abordagem com memória de tradução.** In: III Workshop em Tecnologia da Informação e da Linguagem Humana – TIL 2005 (XXV Congresso da Sociedade Brasileira de Computação). São Leopoldo, RS, de 25 à 29 de julho de 2005. Anais do XXV Congresso da Sociedade Brasileira de Computação – SBC 2005, pp. 2099-2107, jul./2005.

Vijay-Shanker, K.,Weir, D. J., and Joshi, A. K. (1986). **Tree adjoining and head wrapping**. In Proceedings of the 11th coference on Computational linguistics, pages 202–207, Morristown, NJ, USA. Association for Computational Linguistics.