COPPE/UFRJ

EXTRAÇÃO DE GRAMÁTICAS SINCRÔNICAS PARA TRADUÇÃO DE LINGUAGENS NATURAIS EM LINGUAGENS FORMAIS

Natália Giordani Silveira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro

Setembro/2010

Silveira, Natália Giordani

Extração de Gramáticas Sincrônicas para Tradução de Linguagens Naturais em Linguagens Fomais/ Natália Giordani Silveira. – Rio de Janeiro: UFRJ/COPPE, 2010.

IX, 103 p.: il.; 29,7 cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 99-103

1. Gramáticas sincrônicas. 2. Formalização de linguagem natural. 3. Análise semântica. 4. Tradução automática. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Agradeço imensamente à minha família, em especial minha irmã Marina, pelo apoio contínuo e incondicional que me ofereceram generosamente ao longo desse período de intenso trabalho, bem como em toda minha vida. Agradeço a meu orientador, Geraldo Xexéo, pela atenção e o interesse que sempre me dedicou, e pelo salto de fé ao aceitar uma aluna com formação e perspectivas completamente diferentes das suas para ensinar e orientar. Agradeço aos amigos que tornaram minha experiência na COPPE/UFRJ tão difícil de deixar para trás, e aos alunos e professores que tive na instituição, que me enriqueceram imensamente com seu conhecimento, suas ideias e perguntas. Agradeço ainda aos membros da banca, Adriana Vivacqua e Geraldo Zimbrão, que gentilmente aceitaram me trazer seus julgamentos e pontos de vista sobre esta dissertação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

EXTRAÇÃO DE GRAMÁTICAS SINCRÔNICAS PARA TRADUÇÃO DE LINGUAGENS NATURAIS EM LINGUAGENS FORMAIS

Natália Giordani Silveira

Setembro/2010

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Este trabalho propõe uma abordagem ao problema de encontrar uma representação semântica em uma linguagem formal para um enunciado em linguagem natural. Uma tal representação pode ser obtida pelo uso de uma gramática sincrônica que recubra a correspondência entre as duas linguagens. Apresenta-se, portanto, um procedimento para aprendizado automático de uma gramática sincrônica para duas linguagens dadas, uma natural e uma formal e livre de contexto, e de uma parametrização da gramática que permita escolher a melhor derivação onde houver ambiguidade. Além disso, é investigada a influência do uso de filtros de palavras e de transformações na gramática sobre o resultado final.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

EXTRACTION OF SYNCHRONIC GRAMMARS FOR THE TRANSLATION OF NATURAL LANGUAGES INTO FORMAL LANGUAGES

Natália Giordani Silveira

Setembro/2010

Advisor: Geraldo Bonorino Xexéo

Department: Computing and Systems Engineering

This work presents an approach to the problem of finding a semantic representation in a formal language for the natural language utterance. Such a representation may be obtained by the use of a synchronous grammar that covers the correspondence between the two languages. A procedure is thus presented for the automatic learning of a synchronous grammar for two given languages, one natural and one formal and context-free, and a parametrization of the grammar that permits an analyzer to choose the best derivation where ambiguity is found. Furthermore, the influence of the use of filters and transformations on the grammar is investigated.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1. Introdução

Esta dissertação explora uma solução para um problema tradicional da área de Processamento de Linguagem Natural: a análise do conteúdo semântico de enunciados em linguagem natural, neste trabalho entendida como sua tradução para uma linguagem formal adequada ao processamento automático.

Como anuncia a abertura de um dos principais livros-texto da área, "A ideia de dar ao computador a capacidade de processar a linguagem humana é tão antiga quanto a própria ideia de um computador." (JURAFSKY e MARTIN, 2008, p. 1) O estudo desse processamento tem como objetivo principal desenvolver modelos e algoritmos que permitam automatizar tarefas relacionadas às linguagens naturais usadas na comunicação entre humanos. Isso inclui um amplo leque de problemas, que se relacionam com essas línguas de diferentes formas. Algumas das aplicações clássicas da área são a construção de agentes conversacionais (capazes de interagir com o usuário por meio da linguagem natural e usar a interação para tomar decisões) (JOHNSTON *et al*., 2002; ACOMB *et al*., 2007), sistemas de resposta a perguntas (que realizem a análise tanto da pergunta quanto de extensas coleções de texto contendo as informações necessárias para respondê-la, geralmente exigindo inferências) (CHIANG *et al*., 2009; LI *et al*., 2009), sumarização automática (compreendendo tanto a seleção da informação relevante quanto a seleção ou geração do texto do resumo) (VANDERWENDE *et al*., 2007; SIDDHARTHAN e COPESTAKE, 2008; FATTAH e REN, 2009), extração de informações (ou seja, identificação em texto livre de dados pertencentes a categorias estruturadas) (BETHARD e MARTIN, 2007; PANG e LEE, 2008; PRASAD e PAEPCKE, 2008; SONG *et al*., 2009) e — possivelmente o mais

tradicional — a tradução automática (KOEHN *et al*., 2003; CHIANG, 2005; KOEHN *et al*., 2007).

O comportamento linguístico complexo típico das interações entre humanos requer conhecimento dos vários níveis de estruturação da linguagem natural. Um ser humano é capaz de reconhecer, distinguir e produzir os sons de sua língua nativa; ele conhece os significados de unidades lexicais e é capaz de compreender unidades desconhecidas pelo reconhecimento e composição de suas partes; consegue reconhecer e produzir um ordenamento lícito das palavras em uma frase, e o que este diz sobre a forma como estão relacionadas; dessas relações e de seus significados básicos, compõe proposições que se referem ao mundo extralinguístico; entende como essas proposições se relacionam às intenções daquele que as enuncia, em diferentes contextos; e é capaz de processar enunciados linguísticos longos e complexos, compreendendo relações estruturais e de significado que vão muito além do domínio da frase. Esses níveis de conhecimento são tipicamente definidos como fonética e fonologia (o conhecimento dos sons de uma língua e suas combinações), morfologia (das unidades básicas de significado, ou morfemas, e sua composição), sintaxe (das articulações entre as palavras em estruturas linguísticas), semântica (do significado de palavras e estruturas), pragmática (da relação entre o enunciado linguístico, os participantes da interação e o contexto) e discurso (das interações entre estruturas linguísticas em macroestruturas).

Um outro conjunto de tarefas do processamento de linguagem natural se relaciona portanto com a análise da linguagem nesses vários níveis, e compreende subproblemas cuja resolução dá suporte às aplicações citadas; é o caso, entre outros, da análise morfossintática (BRILL, 1995; NARADOWSKY e GOLDWATER, 2009), da análise sintática (COLLINS, 1997a; SMITH e EISNER, 2007), e da análise semântica.

Nesta dissertação, preocupamo-nos com a terceira. Apresentamos a análise semântica como um problema de codificação do significado de fragmentos de linguagem natural em uma linguagem formal, e investigamos um método para aprendizado de regras de tradução entre uma linguagem natural e uma linguagem formal livre de contexto. As áreas de aplicação são várias; a obtenção da representação formal é um passo de processamento desejável no suporte a várias tarefas de alto nível do Processamento de Linguagem Natural, como a resposta a perguntas e a sumarização automática. Além disso, a tradução entre linguagens naturais e linguagens formais tem um papel importante na construção de interfaces baseadas em linguagem natural, como em interfaces a bancos de dados ou ficção interativa, além de ser um componente da tradução baseada em interlíngua. O método investigado alia ferramentas da tradução automática e da teoria de compiladores.

No Capítulo 2, contextualizamos a abordagem ao problema apresentando brevemente a área de tradução automática. No Capítulo 3, detalhamos as ferramentas teóricas utilizadas na solução. No Capitulo 4, mostramos uma série de trabalhos relacionados, introduzindo os algoritmos propostos e os resultados obtidos. No Capítulo 5, propomos um procedimento passo-a-passo para o aprendizado de tradutores automáticos entre pares de linguagem natural e linguagem formal, mostrando os algoritmos testados e os desafios encontrados, bem como os resultados de suas avaliações.

Capítulo 2. Descrição do problema

2.1. Representações do significado

Baseamo-nos em uma abordagem representacional à semântica, tomando como premissa que o significado de um enunciado linguístico pode ser capturado em uma estrutura formal, a que nos referimos como **representação do significado**.[1] Uma tal representação deve ser expressa em uma linguagem cujas expressões podem ser mapeadas de forma sistemática para um **modelo**, ou seja, uma estrutura formal que represente certos objetos, suas propriedades e as relações entre eles. Esse mapeamento é a função de **interpretação** da linguagem. O conjunto de objetos representados no modelo é seu **domínio**, e a extensão de uma expressão da linguagem é sua **denotação**.

Uma representação semântica adequada a aplicações computacionais deve ser expressa em um formalismo que tenha as seguintes propriedades: **verificabilidade** — deve haver no sistema uma forma de verificar o valor-verdade da expressão; **não ambiguidade** — a expressão não deve permitir mais de uma interpretação; **forma canônica** — reciprocamente, o mesmo significado deve ser representado sempre pela mesma expressão; **suporte a inferências e variáveis** — as expressões devem se prestar à manipulação para avaliação do valor-verdade de proposições que não estão explicitamente representadas; e **expressividade** adequada — o formalismo precisa ser suficientemente poderoso para veicular os conteúdos semânticos de interesse (JURAFSKY e MARTIN, 2008).

[1] Não nos estendemos sobre as implicações dessa premissa do ponto de vista da filosofia da linguagem. Para uma cobertura dessa questão, indicamos MARTINICH (1990) e LYCAN (2000).

2.2. Análise semântica

A análise semântica é a formalização de componentes semânticos de enunciados em linguagem natural em uma linguagem de representação de significado. Pode se dar em diferentes graus de detalhamento; a mera identificação de entidades em um texto e seus respectivos papéis nas relações que travam entre si (*semantic role labeling*) — papéis como 'agente', 'paciente', 'meio', 'objetivo', entre outros, que se pode atribuir aos participantes do estado de coisas construído por um fragmento de linguagem natural — é uma tarefa de análise semântica (GILDEA e JURAFSKY, 2002). Seu resultado, porém, não é uma representação completa do significado, e sim uma formalização de alguns de seus aspectos. Uma análise semântica mais poderosa deve capturar as especificidades das relações estabelecidas no estado de coisas construído, e, sobretudo, as inferências a que dão abertura.

Segundo MOONEY (2007), a **análise semântica profunda** pode ser entendida como o mapeamento de um enunciado em uma representação formal completa e correta de seu significado, de maneira a permitir que fragmentos de linguagem natural surtam certos efeitos sobre o sistema em que são enunciados.

Assim se estabelece uma relação entre a linguagem a mapear e a linguagem do mapeamento, e é pelo entendimento dessa relação que a análise semântica profunda pode, no contexto do Processamento de Linguagem Natural, ser entendida como fundamentalmente uma tarefa de tradução, de uma linguagem natural para uma linguagem formal. Essa abordagem permite que se explore um vasto conjunto de tecnologias pertencentes ao domínio da tradução automática, que, embora tradicionalmente aplicada a pares de linguagem natural-linguagem natural, pode contribuir para problemas de linguagem natural-linguagem formal.

Pode-se definir, portanto, a análise semântica como um mapeamento entre NL e MRL, tal que qualquer enunciado na linguagem formal MRL é o resultado da análise semântica (tradução) de pelo menos um enunciado na linguagem natural NL.

Definida a análise semântica profunda como um problema de tradução automática, obtém-se um espaço de soluções baseadas em abordagens ao problema da tradução: uma parte desse espaço corresponde ao domínio da tradução automática clássica, aplicada a pares de linguagens naturais; e outra, ao domínio da compilação, aplicada a pares de linguagens formais. A solução apresentada nesta dissertação empresta elementos desses dois universos; na próxima seção, apresentamos brevemente o da tradução automática.

2.3. Traduções automáticas

Abstratamente, pode-se entender uma tradução como um conjunto de pares de sentenças em que cada elemento pertence a uma linguagem diferente (AHO e ULLMAN, 1972). Sejam portanto L_1 e L_2 duas linguagens, e Σ e T seus respectivos alfabetos. Define-se a tradução τ como um conjunto de pares

$$\tau = \{(e, f) \mid e \in L_1, f \in L_2, x \text{ e } y \text{ são equivalentes traducionais}\}$$

Historicamente, tarefas de tradução automática em geral se concentram em pares de linguagens da mesma natureza: ambas naturais ou ambas formais. O primeiro caso é o objeto de estudo do campo da tradução automática em Processamento de Linguagem Natural; o segundo, do de Compiladores. A caracterização da relação de **equivalência traducional** varia de acordo com a aplicação; sua essência é que as sentenças são mapeadas para a mesma representação em um modelo, do ponto de vista semântico, e atendem adequadamente o mesmo conjunto de necessidades linguísticas, contextualmente definidas, do ponto de vista pragmático.

O campo da tradução automática das línguas humanas situa-se na origem histórica do próprio Processamento de Linguagem Natural, e mantém hoje sua posição como um dos mais problemas mais estudados na área. Como o objetivo do estudo da tradução automática se refere em geral a pares de linguagens naturais, nesses casos a relação de equivalência traducional se estabelece por uma mistura complexa de requisitos semânticos e pragmáticos (e em certos casos, também estilísticos), que, pela sua sofisticação e dificuldade de apreensão em regras, impõem grandes dificuldades.

As abordagens clássicas à tradução automática se baseiam no estabelecimento de correspondências determinísticas entre estruturas na linguagem-origem e na linguagem-alvo; com a evolução dos algoritmos, essas correspondências passaram a ser procuradas em níveis cada vez mais sofisticados. Um diagrama conhecido como Triângulo de Vanquois (Figura 1; JURAFSKY e MARTIN, 2008) ilustra os diversos níveis e as relações entre as abordagens que enfocam cada um deles.

Na **tradução direta**, a abordagem mais ingênua, o texto-fonte é traduzido palavra por palavra, segundo um dicionário bilíngue, para a linguagem-alvo. Após a tradução de todas as palavras, um reordenamento local pode ser aplicado ao texto-alvo. Como não há nenhum conhecimento da estrutura sintática, não se reordenam constituintes, apenas palavras. Esta abordagem não é mais usada diretamente em sistemas de tradução automática, mas a intuição original da transformação passo a passo de um texto em outro permanece subjacente aos sistemas modernos (JURAFSKY e MARTIN, 2008).

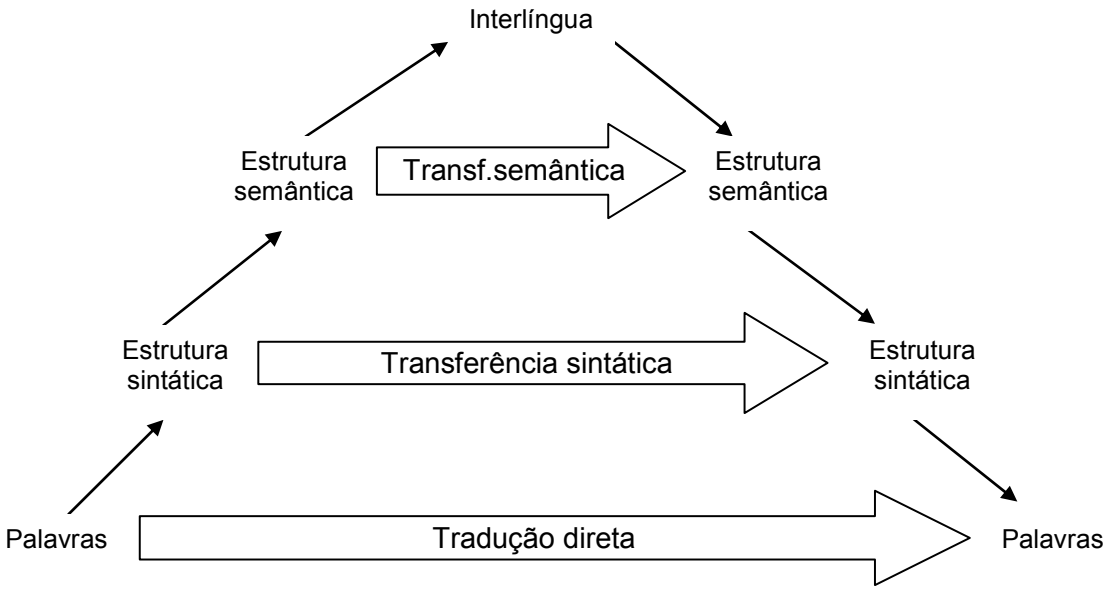

Figura 1. Triângulo de Vanquois.

A abordagem de **transferência** procura levar em conta as diferenças sistemáticas entre as estruturas sintáticas de duas linguagens e parte da análise sintática para um mapeamento mais sofisticado, no nível dos constituintes, e não das palavras. Continua existindo um conjunto de operações determinísticas que compõem a transformação, mas tais regras não se referem mais somente a itens lexicais isolados, mas também a estruturas sintáticas, como fragmentos de árvore. Mapeamentos entre estruturas sintáticas são combinados a mapeamentos lexicais.

Em um nível maior de complexidade, abordagens em aplicações reais combinam aspectos da tradução direta e da transferência, com análise morfossintática, dicionários ricos que mapeiam expressões multiverbais, reordenamento de fragmentos sintáticos, e geração de morfemas.

Todas essas abordagens se baseiam na construção de recursos para pares específicos de linguagens, o que obviamente levanta um problema em contextos multilíngues. Em grande parte por isso, surgiu outra perspectiva sobre o problema da tradução automática. Se fosse possível usar uma única linguagem suficientemente

expressiva como intermediária de todos os pares, os sistemas de tradução poderiam ser divididos em dois subsistemas: tradução da linguagem-fonte para a linguagem intermediária, e da linguagem intermediária para a linguagem-alvo. Isso faria com que a quantidade de recursos necessários crescesse linearmente em relação ao número de línguas em sistemas multilíngues, em vez de quadraticamente, pois seria necessário somente construir tradutores para uma única intermediária canônica. Esse é o esquema básico da tradução baseada em uma **interlíngua**.

Esse esquema pressupõe, naturalmente, o uso de uma interlíngua (a linguagem intermediária) apropriada, suficientemente expressiva e rigorosa para capturar em uma mesma forma canônica a expressão de uma determinada proposição em qualquer outra linguagem. Volta-se com isso ao problema das representações de significado, já mencionado; escolhida uma representação adequada, surge então novamente o problema que se propõe nesta dissertação: a tradução da linguagem natural para a linguagem formal da representação.

Essas três abordagens tradicionais vêm sendo suplantadas nos últimos anos pela **tradução automática estatística** (LOPEZ, 2008). Nesse outro paradigma, o conhecimento da estrutura das linguagens é relegado a segundo plano, e o problema da tradução passa a ser encarado como um problema de aprendizado de máquina. Algoritmos de aprendizado são aplicados a grandes volumes de textos "paralelos", ou seja, divididos em unidades-fonte mapeadas em unidades-alvo por uma bijeção.

Um modelo possível do objetivo de uma tradução entre linguagens naturais é, segundo JURAFSKY e MARTIN (2008), "a produção de um resultado que maximiza alguma função que representa a importância tanto da fidelidade quanto da fluência." Essa caracterização em termos de fidelidade e fluência reflete um desafio típico da tradução de línguas humanas: o desafio de acomodar o sentido literal do enunciado-

fonte em um enunciado na língua-alvo não apenas correto, mas que soe natural e plausível. A tradução automática estatística é um conjunto de abordagens que buscam obter esse resultado através da construção de modelos probabilísticos de fidelidade e fluência. Considerando, portanto, uma sentença na língua-fonte, *f*, e um conjunto de sentenças possivelmente equivalentes como tradução na língua-alvo, *e*, pode-se dizer que a melhor tradução $\hat{e}$ é dada por:

$$\hat{e} = \arg\max_{e} P(e \mid f)$$

Pelo Teorema de Bayes:

$$\hat{e} = \arg\max_{e} \frac{P(f \mid e)P(e)}{P(f)}$$

Essa redução permite ignorar *P*(*f*), pois o resultado não é afetado por esse fator. Conceitualmente, *P*(*f*) se refere à probabilidade da sentença-fonte em um modelo da linguagem-fonte, que é irrelevante porque a sentença-fonte é dada. Resta assim a Equação Fundamental da Tradução Automática:

$$\hat{f} = \arg\max_{f} P(f \mid e)P(e)$$

Os dois componentes da equação, $P(f \mid e)$, o **modelo da tradução**, e *P*(*e*), o **modelo da linguagem** (que informa a probabilidade de um enunciado em uma determinada linguagem) se referem aos dois parâmetros previamente citados: fidelidade (no modelo da tradução) e fluência (no modelo da linguagem). Os três problemas básicos dos sistemas de tradução automática, portanto, são as estimativas desses modelos, e a determinação das variáveis que são a fonte subjacente das observações.

Capítulo 3. Ferramentas teóricas

Este capítulo introduz definições de conceitos e estruturas sobre os quais se alicerça a solução explorada para o problema apresentado no Capítulo 2.

3.1. Gramáticas

Para a discussão sobre *parsers* e compiladores, apresentamos algumas definições formais, recorrentes na literatura, dos principais conceitos em questão.

Seja Σ um conjunto de **palavras**[2] denominado **alfabeto** ou **léxico**. Seguindo HOPCROFT e ULLMAN (1979), apresentamos "palavra" como um conceito primitivo, sem uma definição formal. Uma sequência de palavras é uma **sentença** ou ***string***. Uma **linguagem formal** *L* é um conjunto de **sentenças**. Um dos artifícios formais de especificação de uma linguagem é a **gramática**[3].

Uma gramática G_L da linguagem *L* é uma tupla (*N*, Σ, R, *S*) onde:

(1) *N* é um conjunto finito de símbolos não-terminais;

(2) Σ é um conjunto finito de palavras tal que $N \cap \Sigma = \varnothing$;

(3) R é um conjunto finito de **regras de reescrita**, ou seja, pares ordenados $<\alpha, \beta>$ (chamados, respectivamente, de *LHS* e *RHS*; doravante notados como $\alpha \rightarrow \beta$), onde α, β são sequências de elementos de *N* ou Σ;

(4) *S* é um elemento distinto de *N* chamado **símbolo inicial**.

---

[2] Usamos aqui o termo 'palavra' onde alguns autores em Ciência da Computação usam 'símbolo'. Em inglês, é comum designar um elemento do alfabeto como '*symbol'* e uma sequência bem-formada (ou seja, pertencente à linguagem) desses símbolos como '*word'*. Aqui chamamos de 'palavras' os elementos do alfabeto, e de 'sentenças', ou *'strings'* as sequências bem-formadas de palavras. A escolha se dá para acomodar, dado que nos ocupamos de linguagens naturais, o sentido de 'palavra' na Linguística como uma unidade básica dessas linguagens, mais próximo portanto da noção de elemento do alfabeto do que de sequência bem-formada.

[3] Na literatura linguística, esse tipo de gramática é conhecido como **gramática de estrutura de constituintes**, por atribuir uma hierarquia de constituintes sintáticos à sentença.

Dada uma gramática $G = (N, \Sigma, R, S)$, a gramática $G(X)$ é definida por (N, Σ, R, X), onde $X \in N$.

G é uma **gramática livre de contexto** se, para todos os pares $<\alpha, \beta>$ em R, α contém um único elemento de N. L é uma **linguagem livre de contexto** se existe alguma gramática livre de contexto que defina L.

Uma **regra-ε** é uma regra da forma $X \rightarrow \varepsilon$.

Para compreender a definição de linguagens pelo uso de gramáticas, é necessária a noção de **derivação**. Seja $\alpha \Rightarrow_G \beta$ ("α **deriva diretamente** β em G") uma relação entre sequências de elementos de $N \cup \Sigma$. Sejam $\alpha X \beta$ e $\alpha\gamma\beta$ duas tais sequências; $\alpha X \beta \Rightarrow_G \alpha\gamma\beta$ se $R(G)$ contém uma regra tal que $X \rightarrow \gamma$. Já a relação $\alpha \underset{G}{\Rightarrow}^{*} \beta$ ("α **deriva** β em G") é o fechamento reflexivo-transitivo da relação anterior ($\alpha \underset{G}{\Rightarrow}^{+} \beta$ e $\alpha \underset{G}{\Rightarrow}^{k} \beta$ são, respectivamente, o fechamento transitivo e o produto k-tuplo da relação). Uma sequência de terminais e não-terminais α é chamada uma **forma sentencial** de G se

$$S \underset{G}{\Rightarrow}^{*} \alpha .$$

Assim definidas essas relações, pode-se afirmar que

$$L(G) = \left\{ w \mid w \in \Sigma, S \underset{G}{\Rightarrow}^{*} w \right\}$$

Uma gramática é **livre de ciclos** se não existe nenhuma derivação da forma $X \underset{G}{\Rightarrow}^{+} X$ para nenhum X em G.

Duas gramáticas G_1 e G_2 são ditas **fracamente equivalentes** se $L(G_1) = L(G_2)$, e **fortemente equivalentes** se, além disso, para toda *string* S pertencente a $L(G_1)$ e $L(G_2)$, o conjunto de possíveis derivações de S for igual para ambas as gramáticas.

3.1.1. Árvores de derivação

Uma derivação pode ser apresentada na forma de uma **árvore ordenada e rotulada** (nesse contexto geralmente chamada **árvore de derivação**).

$$A = (N, V, R)$$

onde N é um conjunto de nós rotulados, V é um conjunto de pares ordenados denotando arestas direcionadas entre os membros de N, e R é a raiz. A é uma árvore de derivação em $G(\mathrm{M}) = (N, \Sigma, R, X)$ se

(1) a raiz de A tiver rótulo X;

(2) se F_1, ..., F_i são subárvores enraizadas nos filhos da raiz de A e o rótulo de F_i é X_i , então $A \rightarrow X_1 \ldots X_k \in R$. F_i deve ser uma árvore de derivação em $G(\mathrm{F}_i)$ se $X_i \in N$, ou um único nó se $X_i \in \Sigma$. Caso a raiz de A tenha um único filho de rótulo X, analogamente deve haver uma regra $A \rightarrow X$ em R.

Um **caminho** em A é uma sequência de nós $X_1,\ldots, X_k$ tal que, para qualquer $1 \leq i \geq k$, existe um aresta $<X_i, X_{i+1}>$ em A.

O nó X **domina** o nó Y se $X \neq Y$ e X faz parte do caminho da raiz da árvore até Y.

Entre um par de nós X e Y tal que nenhum caminho em A contém X e Y, existe a relação de **precedência**. Se X e Y são irmãos, X precede Y se seus rótulos são x_i e x_j pertencentes a uma regra $\mathrm{A} \rightarrow x_1 \ldots x_k \in \mathrm{R}(G)$ e $i < j$. A relação é estendida para pares de nós com pais diferentes pela afirmação de que, se X precede Y, todos os nós dominados por X precedem todos os nós dominados por Y.

Um **corte** de uma árvore A é um conjunto C de nós de A tais que:

(1) não há dois nós X e Y em C tais que X domina Y;

(2) nenhum nó pode ser adicionado a C sem violar a condição (1).

Seja a concatenação dos rótulos dos nós de um corte da árvore, em ordem de precedência, uma sequência chamada **fronteira interna**. Em uma árvore de derivação

em uma gramática livre de contexto, α é uma fronteira interna se e somente se α é uma forma sentencial, ou seja, se $S \underset{G}{\Rightarrow}^{*} \alpha$ (AHO e ULLMAN, 1972). Uma fronteira interna α da subárvore enraizada por *X* será designada uma **fronteira interna de *X***; em uma árvore que represente uma derivação livre de contexto, a fronteira interna de qualquer nó é uma *substring* da fronteira interna da árvore. Observa-se que, uma vez que $X \underset{G}{\Rightarrow}^{*} \alpha$, a introdução da regra $X \rightarrow \alpha$ na gramática não altera a linguagem reconhecida por esta.

Definimos uma terceira ordem parcial sobre o conjunto *N*(*A*) de nós da árvore sintática: **precedência sintática**.

Um nó terminal *t* **precede sintaticamente** outro nó terminal *u* se a palavra que rotula *t* ocupa na sentença derivada uma posição inferior à da palavra que rotula *u*. Para conceituar a relação de precedência sintática envolvendo nós não-terminais, definimos duas funções: **fronteira** e **canto esquerdo**. A fronteira de um nó é a concatenação dos rótulos de todos os nós terminais da maior subárvore de derivação nele enraizada, na ordem de sua posição na sentença derivada. O canto esquerdo de um nó é o nó rotulado com a primeira palavra de sua fronteira, ou seja, o terminal cujo rótulo tem a menor posição na sentença derivada. Podemos dizer então que um nó *X* precede sintaticamente outro nó *Y* se o canto esquerdo de *X* precede sintaticamente o canto esquerdo de *Y*.

Em uma árvore que represente uma derivação livre de contexto, se o nó *X* precede sintaticamente o nó *Y*, obrigatoriamente todos os nós precedidos sintaticamente por *X* precedem sintaticamente todos os nós precedidos sintaticamente por *Y*. Uma árvore de derivação em que não se verifica essa propriedade não pode ser gerada por uma gramática livre de contexto, pois, dado um corte α*X*β da árvore, a fronteira t_i^j de *X* é tal que $X \underset{G}{\Rightarrow}^{*} t_i^j$, e portanto deve ser um fator de $\alpha t_i^j \beta$; se existe um nó dominado por

X que é precedido sintaticamente por um nó dominado por *Y*, a fronteira de *X* não pode ser contínua.

Duas árvores de derivação A_1 e A_2 são **isomórficas** se existe uma função bijetora $f: A_1 \rightarrow A_2$ tal que

(1) se $<X, Y>$ é uma aresta em A_1, $<f(X), f(Y)>$ é uma aresta em A_2;

(2) se $<X, Y>$ não é uma aresta em A_1, $<f(X), f(Y)>$ não é uma aresta em A_2;

(3) se *X* é a raiz de A_1, $f(X)$ é a raiz de A_2.

A subárvore de A que contém todos, e apenas, os nós rotulados com símbolos não-terminais de *G* será chamada a **subárvore não-terminal** de A.

Chama-se de **árvore alinhada** uma árvore de derivação anotada com alinhamentos entre seus nós e pares (e_i, i) de palavras de uma sentença *e* em outra linguagem e sua posição *i*. Em uma árvore alinhada, o **escopo** de um nó *X* (denotado por e(*X*)) é a subsequência das palavras de *e* alinhadas a *X* e aos nós dominados por *X*. O **fechamento do escopo** de *X* (denotado por $\xi(X)$) é a menor *substring* (ou seja, subsequência contínua) de *S*' que contém todos os elementos de seu escopo.

Uma **derivação mais à esquerda** (*leftmost derivation*) é uma derivação em que, para qualquer subsequência de seus *k* passos $<X_1, \beta_1>, <X_2, \beta_2> \ ... <X_k, \beta_k>$, não há nenhum par (X_i, X_j) tal que $i < j$ e X_j precede X_i.

Uma gramática que atribui mais de uma árvore de derivação à mesma sentença é considerada **ambígua**.

3.1.2. Gramáticas livres de contexto ponderadas

Quando uma gramática é ambígua, é útil ter alguma forma de decidir qual é a melhor derivação, entre as possíveis, de uma determinada sentença. Para esse fim, as regras de uma gramática livre de contexto podem ser estendidas para incluir uma ponderação que informa qual é a melhor escolha entre as possíveis. Temos portanto uma

gramática (N, Σ, R, S) em que R contém triplas <A, α, p> (doravante A → α [p]), onde p expressa algum valor de ponderação que um analisador sintático adequado poderá utilizar em uma função de decisão. Um tipo comum de gramática ponderada são as gramáticas probabilísticas, nas quais o peso de uma regra é a sua probabilidade de ocorrência em uma derivação. Nesse caso, $0 \geq p \leq 1$, e seu valor expressa P(β | A) (doravante P($N \rightarrow \beta$)). A soma das probabilidades de todas as regras com o mesmo lado esquerdo deve ser 1:

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

Essa extensão permite lidar com derivações ambíguas porque o uso das probabilidades das regras permite atribuir uma probabilidade às derivações que estas compõem. A probabilidade de uma derivação D para a sentença S é o produtório das N regras nela usadas:

$$P(D,S) = \prod_{i=1}^{n} P(A_i \rightarrow \alpha_i)$$

A derivação mais provável é dada por:

$$\hat{D} = \arg\max_{D} \prod_{i=1}^{n} P(A_i \rightarrow \alpha_i)$$

Note-se que a natureza livre de contexto das regras impõe uma presunção de independência a essa estimativa (JURAFSKY e MARTIN, 2008), o que não reflete adequadamente a forma como se articulam dependências sintáticas estruturais compreendendo vários constituintes da frase.

As probabilidades atribuídas às regras podem ser aprendidas de diferentes maneiras. Se existe disponível um corpus anotado com a derivação correta de cada sentença, é possível simplesmente contar o número de vezes em que cada não-terminal

tem uma determinada expansão e então dividi-lo pelo número de ocorrências do não-terminal:

$$P(\alpha \to \beta|\alpha) = \frac{Count(\alpha \to \beta)}{\sum_{\gamma} Count(\alpha \to \beta)} = \frac{Count(\alpha \to \beta)}{Count(\alpha)}$$

A situação mais comum, porém, é que um tal corpus não esteja disponível, caso em que é preciso estimar essas probabilidades. Tendo-se um parser não probabilístico, é possível computar todas as derivações possíveis de cada sentença; para cada uma delas, então, pode-se manter uma contagem das ocorrências de cada regra ali usada. No entanto, como algumas derivações são menos prováveis do que outras, é preciso de alguma forma refletir esse desequilíbrio, de modo que as contagens devem ser pesadas pela probabilidade da derivação — o que em teoria exigiria que já se dispusesse desses valores.

A resolução desse problema se dá pelo uso de algoritmos de maximização de expectativas (MANNING e SCHÜTZE, 1999); para manter a continuidade da discussão sobre gramáticas, porém, adiamos a apresentação destes até outra seção do capítulo.

3.2. Analisadores sintáticos (parsers)

Um **analisador sintático** ou ***parser*** é um programa que verifica se uma determinada sentença pertence a uma determinada linguagem e produz alguma representação da(s) derivação(ões) da sentença (diferentemente de um **reconhecedor**, que não desempenha esta segunda tarefa). Segundo AHO e ULLMAN (1972), há três grandes tipos de *parsers*: os universais, que podem ser usados com qualquer gramática livre de contexto, mas são os mais ineficientes; os da-raiz-para-as-folhas (*top-down*), que processam a sentença comparando-a com expansões feitas *a priori* a partir da gramática; e os das-folhas-para-a-raiz (*bottom-up*), que processam a sentença buscando, *a posteriori*, as possíveis árvores sintáticas que podem recobri-lo.

```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE((γ → • S, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
    for each state in chart[i] do
      if INCOMPLETE?(state) and
            NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
      elseif INCOMPLETE?(state) and
            NEXT-CAT(state) is a part of speech then
        SCANNER(state)
      else
        COMPLETER(state)
    end
  end
  return(chart)

  procedure PREDICTOR((A → α • B  β, [i,j]))
      for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar) do
          ENQUEUE((B → • γ, [j,j]),chart[j])
      end

  procedure SCANNER((A → α • B  β, [i,j]))
      if B ⊂ PARTS-OF-SPEECH(word[j]) then
          ENQUEUE((B →  word[j], [j, j+1]), chart[j+1])

  procedure COMPLETER((B → γ •, [j,k]))
      for each (A → α • B  β, [i,j]) in chart[j] do
          ENQUEUE((A → α B •  β, [i,k]), chart[k])
      end

  procedure ENQUEUE(state, chart-entry)
      if state is not already in chart-entry then
          PUSH(state, chart-entry)
      end
```

Figura 2. O algoritmo de Earley. Adaptado de JURAFSKY e MARTIN (2008)

3.2.1. O algoritmo de Earley

O **algoritmo de Earley** (Figura 2) (EARLEY, 1970) é um algoritmo de programação dinâmica para o reconhecimento de linguagens livres de contexto. O algoritmo pode ser aplicado para qualquer linguagem dessa classe; o reconhecedor e o *parser* nele baseados são universais. A análise é feita percorrendo o input uma única vez e preenchendo um *array* que registra possíveis fragmentos de árvore, da-raiz-para-as-

folhas (ou seja, *top-down*). Mais amplo do que os algoritmos LALR e *shift-reduce*, geralmente usados em aplicações práticas voltadas para linguagens formais, o algoritmo de Earley é capaz de analisar toda a classe das linguagens livres de contexto, em tempo polinomial ($\mathbf{O}(n^3)$), ou quadrático ($\mathbf{O}(n^2)$) para gramáticas não ambíguas.

O componente central do algoritmo de Earley é uma tabela de estados, de tamanho $|w| + 1$, que armazena em cada uma de suas entradas uma lista de representações das subárvores sintáticas geradas até aquele momento. Essas representações, ou estados, são chamados **regras pontuadas**, por corresponderem a regras da gramática da linguagem analisada anotadas com um ponto que indica que parte da regra recobre o input encontrado. Cada regra pontuada tem a forma:

$$N \rightarrow \alpha.\beta[i, j]$$

N é um símbolo não-terminal da gramática; α e β são sequências de símbolos terminais e não-terminais, ou ainda a string vazia (ε). $N \rightarrow \alpha\ \beta$ é uma regra da gramática; *i* e *j* são índices denotando o início e o fim de uma subsequência do input. O ponto representa uma possível relação entre o input e a regra; os símbolos que se encontram à esquerda do ponto, ou seja, α, podem ser reescritos na subseqüência identificada por *i* e *j*. Já os símbolos à direita, β, são uma predição, baseada na gramática, do que se encontrará no resto da string. Se $\alpha = \varepsilon$, toda a regra é uma predição; se $\beta = \varepsilon$, diz-se que a regra está completa, e que o símbolo *N* pode ser reescrito como a *substring* (*i*, *j*). Um estado de escopo (*i*, *j*) se encontra necessariamente na entrada *j* da tabela (pois o parser ainda não processou os *tokens* seguintes).

O parser percorre toda a tabela da esquerda para a direita, processando os estados. Uma entre três operações possíveis é aplicada a cada estado examinado. O resultado é sempre a adição de estados na atual ou na próxima entrada da tabela. Nenhum estado é removido.

A inicialização do algoritmo se dá pela função **preditor**. O preditor é chamado quando se processa um estado com um não-terminal à direita do ponto. A função é responsável por inserir na tabela todas as possíveis expansões do não-terminal que ainda não estiverem no atual estado da tabela. Quando o parser encontra uma regra pontuada com um não-terminal à direita do ponto, o preditor recupera na gramática todas as regras encabeçadas pelo não-terminal, que representam suas possíveis expansões, e insere novos estados na tabela com as ditas expansões à direita do ponto. À medida que o parser processa os estados de uma entrada da tabela, esses estados inseridos pelo preditor serão eles mesmos desenvolvidos em novos estados.

Seja a gramática livre de contexto *G*:

$S \rightarrow$ A B

A $\rightarrow$ A a

A $\rightarrow$ a

B $\rightarrow$ B b

B $\rightarrow$ b

A inicialização produziria o seguinte estado:

$S \rightarrow$. A B [0, 0]

Por causa da presença do não-terminal à esquerda do ponto, o processamento desse estado exigiria a aplicação da função preditor, que adicionaria à fila os seguintes estados:

A $\rightarrow$. A a [0, 0]

A $\rightarrow$. a [0, 0]

Já a função ***scanner*** é aplicada quando se encontra um terminal à direita do ponto. Essa função consulta o próximo *token* da string e o compara com o terminal

encontrado no estado em processamento. Se houver uma coincidência, o *scanner* fará uma cópia do estado em processamento, avançará o ponto (indicando que o terminal avaliado foi encontrado no input) e aumentará o escopo do estado. Ao processar o estado

$$A \rightarrow . \ a \qquad [0, 0]$$

o *scanner* produziria, para a string *aabb*:

$$A \rightarrow a \ . \qquad [0, 1]$$

A função **completor**, por sua vez, é chamada quando não há símbolos à direita do ponto no estado em processamento. Essa função é responsável por encontrar estados anteriores que tinham à direita do ponto o não-terminal que encabeça o estado em processamento. O completor recupera a entrada da tabela onde estão os estados cujo escopo termina no mesmo ponto onde começa o escopo do estado em processamento; ou seja, os estados que recobriam a parte do input que precede aquela recoberta pelo atual estado. Como este está completo, o completor pode recuperar os estados antigos que "aguardavam" que se encontrasse aquela categoria e copiá-los para a atual entrada da tabela, avançando a posição do ponto e fazendo a união dos escopos. Portanto, ao encontrar o estado

$$A \rightarrow a \ . \qquad [0, 1]$$

o completor pode inserir na fila o estado

$$A \rightarrow A \ . \ a \qquad [0, 1].$$

A derivação completa da string *aabb*, portanto, com a indicação da função que inseriu o estado, seria:

$S \rightarrow . \ A \ B$ [0, 0] *inicializador*

$A \rightarrow . \ A \ a$ [0, 0] *preditor*

$A \rightarrow . \ a$ [0, 0] *preditor*

A → a .	[0, 1]	*scanner*
A → A . a	[0, 1]	*completor*
A → A a .	[0, 2]	*scanner*
S → A . B	[0, 2]	*completor*
A → A . a	[0, 2]	*completor*
B → . B b	[2, 2]	*preditor*
B → . b	[2, 2]	*preditor*
B → b .	[2, 3]	*scanner*
B → B . b	[2, 3]	*completor*
S → A B .	[0, 3]	*completor*
B → B b .	[2, 4]	*scanner*
S → A B .	[0, 4]	*completor*

Quando não há mais estados a processar, o parser verifica se foi encontrada uma derivação correta buscando, na última entrada da tabela, um estado encabeçado pelo símbolo inicial que tenha escopo sobre a totalidade da string. No exemplo, esse é o caso do último estado inserido.

3.3. Alinhamentos entre palavras

Sejam *e* e *f* duas sequências de palavras nas linguagens L_1 e L_2, respectivamente. Em modelos de tradução baseada em palavras, P(*f* | *e*) para um dado par de sentenças é calculado a partir de alinhamentos entre subsequências de *e* e *f*.

$$P(e \mid e) = \sum_{\Omega} P(f, a \mid e)$$

Embora alinhamentos entre subsequências de várias palavras sejam mais expressivos em termos do que se pode realmente considerar uma boa tradução, a escassez de dados sobre alinhamentos entre expressões equivalentes em diferentes

línguas e o fato de que a separação de palavras é um problema melhor entendido do que a delimitação de expressões de várias palavras não necessariamente equivalentes a constituintes sintáticos fazem com que se recorra a **alinhamentos entre palavras**.

Um alinhamento entre palavras é essencialmente um modelo da tradução de palavras da linguagem L_1 para palavras da linguagem L_2. O modelo se expressa como um mapeamento entre o vocabulário-fonte e o vocabulário-alvo. Sejam $e = e_1^l = \langle e_1,...,e_l \rangle$ e $f = f_1^m = \langle f_1,...,f_m \rangle$. Um alinhamento entre estas é denotado por $\Omega(e,f)$.

Enfocamos os modelos de alinhamento de palavras desenvolvidos por um grupo de pesquisadores da IBM (BROWN *et al.*, 1993), por isso conhecidos como Modelos IBM, que são amplamente usados na pesquisa na área. São cinco modelos, crescentemente complexos, que modelam a probabilidade P(*f* | *e*) como função de séries (crescentemente sofisticadas) de propriedades. Os Modelos IBM restringem as possibilidades de alinhamento àquelas em que cada palavra f_j está conectada a uma e somente uma palavra e_i. Com essa restrição, um alinhamento a pode ser representado como uma série de comprimento *m*:

$$\Omega = \langle a_1,....a_m \rangle \mid 0 \le a_j \le l \text{ para todo } j \text{ de } 1 \text{ a } m$$

onde a_j denota a posição da palavra de *e* a que f_j está alinhada. Se $a_j = 0$, f_j não está alinhada a nenhuma palavra de *e*.

O valor de P(*f* | *e*), nesse modelo, é o somatório das probabilidades dos possíveis alinhamentos:

$$P(f \mid e) = \sum_a P(f, a \mid e)$$

Nos Modelos 1 e 2, um processo gerativo cria diferentes possibilidades de alinhamento entre as sentenças *e* e *f*, dado que |*e*| = *L*. Seguem-se os seguintes passos:

(1) escolhe-se um comprimento *m* para a sentença *f*.

(2) Escolhe-se um alinhamento $\Omega = \langle a_1,....a_m \rangle$ entre as palavras de *e* e as posições de *f*.

(3) Escolhe-se uma palavra de F para cada posição *j* em *f*, traduzindo-se a palavra de E (e_{a_j}) à qual a posição se encontra alinhada.

A relação entre $P(f,a \mid e)$ e as escolhas descritas acima pode ser expressa na seguinte equação:

$$P(f,a \mid e) = P(m \mid e)\prod_{j=1}^{m} P(a_j \mid a_i^{j-1}, f_i^{j-1}, m, e)P(f_j \mid a_1^j, f_i^{j-1}, m, e)$$

No Modelo 1, presume-se que $P(m \mid e)$, ou seja, a probabilidade do comprimento da tradução *f*, é independente de *m* e *e*. Todos os comprimentos dentro de um certo espaço são considerados equiprováveis, de modo que $P(m \mid e)$ é estimado como uma pequena constante ι. Além disso, $P(a_j \mid a_i^{j-1}, f_i^{j-1}, m, e)$ é tratada como dependente apenas de *L*, o comprimento da sentença *e*. Todos os alinhamentos possíveis entre *e* e *f*, fixados seus comprimentos, são considerados equiprováveis, de maneira que nesse modelo a relação entre as posições das palavras alinhadas não influencia a probabilidade de seu alinhamento; esta é, reconhecidamente, uma presunção bastante problemática. Finalmente, $P(f_j \mid a_1^j, f_i^{j-1}, m, e)$ é tratada como dependente somente de f_j e a palavra a que está alinhada, e_{a_j}, sendo aproximada como a probabilidade de tradução de e_{a_j} em f_j, denotada $t(f_j \mid e_{a_j})$. Portanto:

$$P(f,a \mid e) = \frac{\iota}{(l+1)^m}\prod_{j=1}^{m} t(f_j \mid e_{a_j}).$$

$$P(f \mid e) = \sum_a \frac{\iota}{(l+1)^m}\prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

$$\hat{a} = \arg\max_{a} \frac{\iota}{(l+1)^m} \prod_{j=1}^{m} t(f_j \mid e_{a_j})$$

Como ι e L são fixos para um par e, f:

$$\hat{a} = \arg\max_{a} t(f_j \mid e_{a_j}) \qquad 1 < j < M$$

As probabilidades $t(f_j \mid e_{a_j})$ são estimadas usando o algoritmo de maximização de expectativas (MANNING e SCHÜTZE, 1999).

No Modelo 2, a ordem das palavras passa a ser levada em conta, e $P(a_j \mid a_i^{j-1}, f_i^{j-1}, m, e)$ é tratada como dependente de j, a_j, e m, além de L.

Nos Modelos 3, 4 e 5, a sequência f é construída se escolhendo, para cada palavra na sequência E, primeiro o número de palavras a que e estará conectada, uma propriedade conhecida como **fertilidade** que não é levado em conta nos Modelos 1 e 2; depois quais serão essas palavras; e finalmente em que posições serão realizadas. As diferenças entre os três modelos se concentram no último passo. Por estarem fora do escopo deste trabalho, não discutiremos os detalhes dos modelos mais sofisticados.

3.4. Gramáticas sincrônicas

O problema de determinar se um par pertence a τ é análogo ao problema de especificar se uma sentença pertence a uma linguagem infinita. As especificações finitas utilizadas são extensões das estruturas utilizadas no segundo caso. Assim como para o problema de pertinência a uma linguagem, o problema de pertinência a uma tradução pode, entre outros exemplos, ser resolvido com um autômato que recebe como *input* uma sentença x e produz uma sentença y tal que $(x, y) \in \tau$. É possível também utilizar uma extensão das gramáticas gerativas que definimos no início do capítulo. É nesse segundo formalismo que nos concentramos.

Gramáticas sincrônicas livres de contexto (WONG, 2007; doravante simplesmente **gramáticas sincrônicas**, pois não discutiremos estruturas não livres-de-contexto), também chamadas **esquemas de tradução orientada à sintaxe**, são essencialmente gramáticas livres de contexto cujas regras são anotadas com traduções para as palavras geradas. Onde as gramáticas antes descritas geram todas as sentenças pertencentes a uma linguagem, gramáticas sincrônicas geram de uma só vez pares ordenados de sentenças pertencentes a duas linguagens distintas, atribuindo-lhes estruturas sintáticas correlatas.

Formalmente, a gramática sincrônica GS<L_1, L_2> das linguagens L_1 e L_2 é uma tupla ordenada (N, Σ, Φ, R, S). Assim como nas gramáticas assíncronas, N é um conjunto finito de símbolos não-terminais, variáveis que representam pares de linguagens. Σ e Φ são dois conjuntos finitos de palavras, os alfabetos, respectivamente, de L_1 e L_2. N e Σ ∪ Φ são disjuntos. R é um conjunto finito de **regras de reescrita sincrônicas**, ou seja, triplas ordenadas <X, α, β> (doravante $X \rightarrow$ <α, β>), onde $X \in N$, $\alpha \in (N \cup \Sigma)^*$, e $\beta \in (N \cup \Delta)^*$, e os não-terminais de β são uma permuta dos não-terminais de α. Cada não-terminal em α está associado a um não-terminal de β. S, novamente, é um elemento distinto de N, chamado símbolo inicial.

A relação $\alpha \Rightarrow_G \beta$ e seus fechamentos discutidos têm generalizações simples para pares ordenados de sequências, que adotamos. A tradução τ denotada por T, GS<L_1, L_2>, portanto, é o conjunto

$$\tau\left(GS\left\langle L_1, L_2\right\rangle\right)^* = \left\{(x, y) \mid S \underset{GS}{\Rightarrow}^* (x, y), x \in \Sigma^*, y \in \Delta^*\right\}$$

Nesse caso, τ é uma **tradução orientada à sintaxe**.

Dada uma gramática sincrônica GS<L_1, L_2> = (N, Σ, Φ, R, S), a gramática G_1 tal que G_1 = (N, Σ, R_1, S), onde

```
function TRANSFORM-TREE(tree) returns newTree

  newTree .INSERTASROOT(PROCESS(root))
  return(newTree)

  procedure PROCESS(node)
    N → α, β = FIND-IN-GRAMMAR(node, children)
      for i ← from 0 to LENGTH(α)
        if TERMINAL?(n_i) then
          DELETE(n_i)
        else
          MOVE(n_i, j)
          where j | ASSOCIATED(n_i) = β[j]
        end
      end
      for j ← from 0 to LENGTH(β)
        INSERT(m_j, j)
      end
      for each nonterminalNode in children
        PROCESS(nonterminalNode)
      end
    return node
```

Figura 3. Algoritmo de transformação de tradução baseada em árvores com gramáticas sincrônicas. (AHO e ULLMAN, 1972)

$$R_1 = \{n \to \alpha \mid A \to \langle \alpha, \beta \rangle \in R\}$$

será chamada a **gramática-fonte**. Analogamente, a gramática G_2 que $G_2 = (N, \Sigma, R_2, S)$, onde

$$R_2 = \{n \to \beta \mid A \to \langle \alpha, \beta \rangle \in R\}$$

será chamada a **gramática-alvo**.

Como demonstram AHO e ULLMAN (1972), uma tradução orientada à sintaxe pode ser usada em um algoritmo de transformação de árvores de derivação da gramática-fonte, delineado na Figura 3.

A gramática sincrônica produzirá pares ordenados de derivações, e portanto pares de árvores de derivação. Esses pares de árvores $<A_1, A_2>$ têm as seguintes propriedades:

(1) suas subárvores não-terminais são isomórficas, dada a bijeção entre os nós-terminais nas regras da gramática sincrônica;

(2) como ambas são árvores de derivação nas respectivas gramáticas-fonte e -alvo, ambas têm a propriedade da equivalência entre precedência e precedência sintática.

Capítulo 4. Trabalhos relacionados

Neste capítulo, discutimos um conjunto de trabalhos que trazem contribuições ao tema da tradução de linguagens naturais para linguagens formais. Os métodos utilizados pelos vários trabalhos são por vezes muito diferentes, e juntos recobrem um amplo espaço de soluções. A maioria dos trabalhos comentados trate de tradução entre linguagens naturais e linguagens formais; GALLEY ET AL. (2004) é uma exceção, mas foi incluído porque apresenta uma estratégia que pode ser utilizada para a extração de regras sincrônicas livres de contexto como as que aparecem na solução proposta nesta dissertação.

4.1. Extração de regras sentença-para-árvore

GALLEY *et al*. (2004) notam que o modelo tradicional de tradução orientada à sintaxe foi posto à prova por FOX (2002), que mostrou que, em tradução entre linguagens naturais, a restrição de operações de reordenamento de constituintes a nós irmãos na árvore sintática é inadequada para o tratamento de uma série de fenômenos interlinguísticos bastante comuns, mesmo entre línguas parecidas. Os autores propõem então um modelo alternativo de ordenamento de estruturas sintáticas correlatas, que não abre mão da orientação à sintaxe. Em vez de usar transformações aplicáveis somente a nós irmãos, o algoritmo desenvolvido por GALLEY *et al*. (2004) opera com fragmentos de árvores.

O algoritmo proposto gera regras de transformação entre duas línguas a partir de alinhamentos em um corpus paralelo, um dos lados do qual está anotado com derivações sintáticas. O *input*, portanto, consiste em sentenças e as árvores sintáticas de suas traduções. O algoritmo define um processo gerativo pelo qual uma sentença na

linguagem-fonte é mapeada em uma árvore sintática da linguagem-alvo e um método para extrair regras de tradução desse processo.

Seja *f* uma sentença na linguagem-fonte e A_e uma árvore de derivação da linguagem-alvo, cuja fronteira é uma sentença *e*, tradução de *f*. Defina-se a derivação de uma tradução como uma sequência de transformações que mapeiam uma sentença em uma árvore de derivação. Seja α_1^n uma sequência de símbolos que representam *substrings* f_i^j ou subárvores A_{e_N} (enraizadas no nó *N*). Dado um alinhamento Ω, a derivação de uma tradução é uma sequência α_1^n tal que α_1 é a sentença *f* e α_n é a árvore A_e, e, dados dois membros α_{i-1} e α_i:

(1) uma subsequência α' de α_{i-1} foi substituída por uma subárvore A_{e_N} em α_i;

(2) qualquer subárvore da sequência α_{i-1} pertencente à subsequência substituída é uma subárvore de A_{e_N}, que a substituiu;

(3) qualquer subárvore não substituída é disjunta de A_{e_N};

(4) o conjunto dos pares formados pelo *tokens* de *f* com os nós de A_e pelos quais foram substituídos é um superconjunto dos pares dados pelo alinhamento Ω.

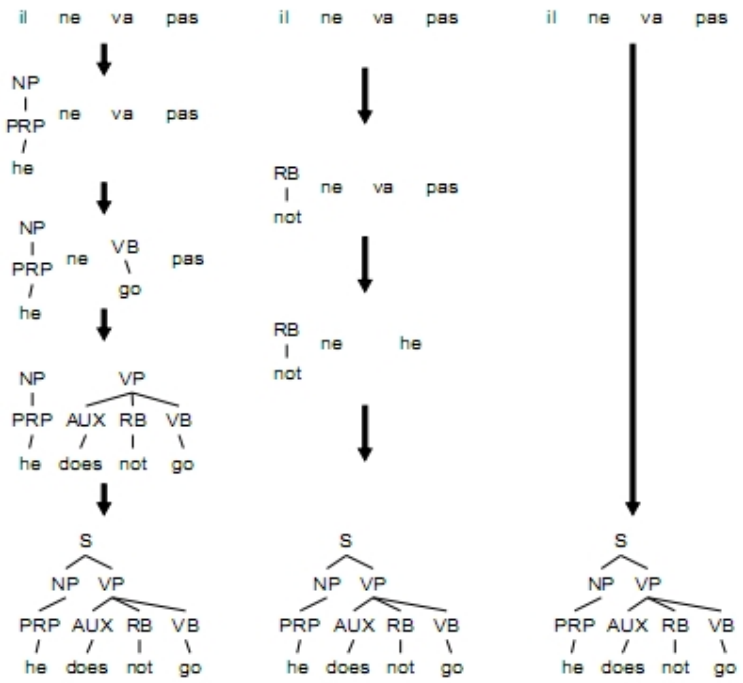

Figura 4. Passos de derivação (GALLEY *et al*., 2004) .

Um passo de derivação que substitui uma substring f_i^j por uma subárvore de A_e pode ser visto como a aplicação de uma regra (Figura 4). O *input* da regra é a sequência das raízes dos símbolos (onde a raiz de um símbolo é o próprio símbolo) ou subárvores substituídos, e o *output* é a diferença entre a subárvore que os substituiu e as subárvores já derivadas, com ponteiros indicando o local de inserção destas.

Assim, a compilação de todos os passos em todas as derivações de Δ compreende todas as regras relevantes que podem ser extraídas de uma tripla (f, A_e, Ω), o conjunto $\rho(f, A_e, \Omega)$. Resta definir um procedimento para extração dessas regras.

Seja A'_e uma árvore alinhada. Seja A'_{e_N} uma subárvore não trivial (ou seja, que contenha mais de um nó) desta que atenda à seguinte propriedade: se o nó N pertence a

A'_{e_N}, então ou todos os filhos de *N* pertencem a A'_{e_N}, ou nenhum filho de *N* pertence a A'_{e_N}.

Sejam os nós de fronteira de uma árvore o conjunto de nós cujo escopo é contíguo. Sejam fragmentos de fronteira os fragmentos nos quais a raiz e todos os nós-folha são nós de fronteira; e fragmentos de fronteira mínimos, aqueles que são um subgrafo de todos os outros fragmentos de fronteira com a mesma raiz. Cada nó de fronteira enraíza um único fragmento de fronteira mínimo, e cada fragmento de fronteira mínimo pode ser convertido em uma regra. GALLEY *et al*. (2004) demonstram que todas as regras assim extraídas pertencem a $\rho(f, A_e, \Omega)$, e conjecturam que são de fato todas as regras do conjunto. Se transformadas em regras livres de contexto, essas regras são equivalentes às obtidas pelo algoritmo básico utilizado em WONG e MOONEY (2006; 2007).

O método foi avaliado em termos da sua capacidade de explicar as traduções encontradas em dois corpora bilíngues: FBIS, em inglês e chinês, e Hansard, em inglês e francês, de acordo com o número de nós permitidos em uma regra — ou seja, o tamanho do fragmento de árvore que pode ser *input* de uma regra. Foram obtidos os seguintes resultados:

Tabela 1. Resultados obtidos por GALLEY *et al*. (2004)

	FBIS	**Hansard**
Traduções explicadas com regras de no máximo 1 nó	12,1%	16,5%
Mínimo de nós permitidos para explicar 100% das traduções	43	23

4.2. Análise semântica integrada a um analisador sintático

GE e MOONEY (2005; 2009) apresentam um método de análise semântica profunda baseado em um analisador sintático estatístico, o Collins Parser (COLLINS, 1996; COLLINS, 1997). A abordagem dos autores estende o analisador sintático para incluir anotações semânticas em cada constituinte, integrando os dois aspectos em um

mesmo modelo estatístico e buscando a análise globalmente mais provável. Um passo adicional mapeia as anotações semânticas para uma expressão na linguagem de representação de significado escolhida.

O Collins Parser atribui uma árvore de derivação livre de contexto a uma sentença com base em um modelo que permite calcular a probabilidade de cada possível derivação na sentença com a gramática de livre contexto dada. A gramática é lexicalizada; suas palavras trazem *tags* sintáticas, e a árvore de derivação resultante tem cada não-terminal anotado com uma palavra e uma *tag*, identificando o núcleo sintático do constituinte que o nó encabeça. Uma regra nessa gramática, portanto, pode ser escrita como

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)....R_m(r_m)$$

onde $H(h)$ é o núcleo sintático do constituinte encabeçado por P, $L_i(l_i)$ é um modificador à esquerda do núcleo, e $R_i(r_i)$ é um modificador à direita do núcleo. A geração das sequências $L(l_{syn})$ e $R(r_{syn})$ é balizada por conjuntos de condições impostas pelo núcleo quanto a que tipo de modificador é aceito; respectivamente, LC_{syn} e LC_{sem}..

A extensão semântica à gramática inclui em cada palavra um rótulo semântico, chegando-se a

$$P(h_{syn}, h_{sem}) \rightarrow L_n(l_{syn\,n}, l_{sem\,n})...L_1(l_{syn_1}, l_{sem1})H(h_{syn}, h_{sem})R_1(r_{syn_1}, r_{sem1})....R_m(r_{syn\,m}, r_{sem\,m})$$

As condições LC_{syn} e LC_{sem} recebem extensões semânticas, chegando-se a $<LC_{syn}, LC_{sem}>$ e $<RC_{syn}, RC_{sem}>$. Após vários passos de suavização, a probabilidade atribuída a cada regra depende de três propriedades:

(1) a probabilidade da anotação h para o núcleo do constituinte —

$$P_{h_{syn}}(H_{syn} \mid P, h) \times P_{h_{sem}}(H_{sem} \mid P, h, H_{syn})$$

(2) a probabilidade de LC e RC dado o núcleo —

$$P_{lc_{syn}}(LC_{syn} \mid P, H, h) \times P_{lc_{syn}}(LC_{sem} \mid P, H, h, LC_{syn})$$

e analogamente para *RC*;

(3) as probabilidades dos modificadores à esquerda e à direita —

$$\prod_{i=1}^{m+1} P_l(L_i(l_i) \mid H, P, h, \Delta_{i-1}, LC)$$

onde Δ é uma medida da distância do núcleo até o canto esquerdo do constituinte; e analogamente para o lado direito.

Essa última probabilidade é ainda mais suavizada pelos autores; GE e MOONEY (2006) informam os detalhes.

O modelo é treinado em um corpus de sentenças inicialmente analisadas com o Collins Parser (modelo 2 — COLLINS, 1997); as árvores de derivação resultantes foram corrigidas manualmente e estendidas com anotações semânticas para obtenção dos dados. Especificamente, as anotações associam às estruturas sintáticas predicados e conceitos relevantes no domínio. A base de predicados (com informação adequada sobre sua valência) e conceitos é construída manualmente. As anotações que puderam ser atribuídas a palavras isoladamente o foram; as anotações atribuídas a expressões foram então usadas no núcleo sintático desta, e as palavras a que isoladamente não se pudesse nenhuma anotação significativa receberam apenas "*null*".

Para construir a representação semântica a partir dessas anotações, (1) encontra-se o "núcleo semântico" da sentença, buscando-se o menor constituinte que tem uma anotação idêntica à da raiz da árvore. As anotações dos nós irmãos são então combinadas por uma função que atribui argumentos a predicadores com base em restrições sobre sua valência, e a expressão final é construída de baixo para cima com as partes combinadas passo a passo.

Os resultados reportados para o corpus GEO880, que contém 880 *queries* para um banco de dados de geografia dos EUA, são precisão (análises corretas/total de análises realizadas) de 91,25% e revocação (análises corretas/total de exemplos) de 72,3%.

Em GE e MOONEY (2006), os mesmos autores modificam a abordagem proposta introduzindo a ideia de reordenamento (*reranking*) das análises semânticas para determinar qual é a mais provável. Cada par de sentença *f* e derivação *D* é mapeado por uma função em um vetor $\overline{\phi}(S,D) \in \Re^d$; um vetor de propriedades $\overline{\theta}$, estimado com uma algoritmo de treinamento de percéptron (COLLINS, 2002), atribui pesos aos traços, produzindo o escore $\overline{\phi}(S,D) \cdot \overline{\theta}$. Para gerar um número maior de árvores candidatas para o ranqueamento, as restrições à seleção de argumentos (*LC* e *RC*) são relaxadas.

As propriedades utilizadas são divididas em dois grupos: as sintáticas e as semânticas. As propriedades sintáticas seguem a solução de COLLINS (2000) para o ranqueamento de derivações sintáticas; as semânticas são análogos, mas com referência às anotações semânticas. Alguns dos traços utilizados foram:

(1) contagem das ocorrências, na derivação, de cada uma das regras livres de contexto presente nos exemplos;

(2) contagem das ocorrências, na derivação, de cada bigrama (i.e., sequência de duas palavras) presente nos exemplos;

(3) mesma informação que (1), mas combinada com o rótulo do nó pai do nó à esquerda da regra;

(4) mesma informação que (2), mas combinada com o rótulo do nó pai do nó à esquerda da regra.

A estimativa de propriedades para esses e outros traços e o uso dos escores resultantes para reordenamento das árvores obtidas não melhorou a performance do algoritmo na base GEO880, mas trouxe um aumento de 2,8% sobre o *F-score* obtido anteriormente, representando uma redução de erros relativa de 15,8%, na base CLANG, que contém 300 exemplos de comandos na linguagem homônima, uma linguagem de instrução de um domínio de futebol entre robôs, e suas traduções em linguagem natural.

GE e MOONEY (2009) introduzem mais modificações, desta vez com os objetivos de automatizar a anotação semântica antes feita manualmente e assegurar que se possa fazer a construção de sentenças na linguagem formal a partir dessa anotação.

Para a tarefa de anotação, os autores adotam uma abordagem de alinhamento de palavras, usando o Modelo IBM 5 em um corpus paralelo de sentenças em linguagem natural e uma representação linear dos predicadores na sentença correspondente na linguagem formal em questão. Para obter essa representação, realiza-se a análise sintática da sentença e se extraem os predicadores na ordem da derivação mais à esquerda.

Um desambiguador estatístico é treinado para atribuir uma distribuição de probabilidades às possíveis derivações de representações de significado para cada sentença. As propriedades consideradas são dos seguintes tipos:

(1) número de vezes em que uma palavra é associada a determinado predicador;

(2) número de vezes em que uma palavra é associada a determinado predicador e certa palavra a precede ou a segue;

(3) número de vezes em que certa regra aparece na derivação.

Somente regras e predicadores usados nas melhores derivações encontradas são preservados. O vetor que atribui pesos aos parâmetros foi estimado com uma variante do algoritmo Inside-Outside (MANNING e SCHÜTZE, 1999).

Os melhores resultados reportados encontram-se abaixo.

Tabela 2. Resultados obtidos por GE e MOONEY (2009).

	CLang	GeoQuery
P	84,73%	74,00%
R	91,94%	88,18%

4.3. Tradução de linguagem natural para o cálculo-lâmbda

ZETTLEMOYER e COLLINS (2005) estão também entre os autores que abordam o problema da análise semântica profunda de enunciados em linguagem natural como tradução para uma linguagem formal, trabalhando porém com uma representação de uso geral, o cálculo-lâmbda, em vez de uma linguagem de domínio. Em seu trabalho, os autores buscam como output uma forma lógica expressa no cálculo lâmbda. Os dados de treinamento são um corpus paralelo contendo sentenças em linguagem natural e suas formas lógicas. O algoritmo desenvolvido induz uma gramática que mapeia uma linguagem na outra e encontra um modelo probabilístico que atribui uma distribuição às possíveis derivações sob a gramática induzida. O formalismo usado é a gramática categorial combinatória (CCG) (STEEDMAN, 2000), que procuramos definir brevemente abaixo para melhor entendimento do método.

Uma gramática categorial combinatória é um tipo especial de gramática categorial (BAR-HILLEL, 1953). Uma gramática categorial consiste basicamente de um léxico, contendo palavras anotadas com categorias (sintáticas ou semânticas, conforme a construção da gramática) e um conjunto de regras sobre como essas categorias se combinam. O formalismo básico é estritamente equivalente a gramáticas livres de contexto. A definição algébrica de uma gramática categorial é uma tupla (Σ, C, LX, R, CE), onde:

(1) Σ é um conjunto finito de palavras;

(2) C é um conjunto de categorias tais que:

a. existe um conjunto E de categorias elementares tal que $E \subset C$;

b. se $X, Y \in C$, então $(X/Y), (X\backslash Y) \in C$;

c. nada está em C exceto por (a) e (b).

(3) LX é um conjunto finito tal que $LX \subset (W \times C)$;

(4) R é um conjunto que compreende os dois esquemas de regra a seguir:

d. $\alpha_{(Y/X)} \bullet \beta_{(Y)} \rightarrow \alpha\beta_{(X)}$;

e. $\beta_{(Y\backslash X)} \bullet \alpha_{(Y)} \rightarrow \beta\alpha_{(X)}$;

(5) CE é um conjunto que compreende as categorias chamadas *expressões completas*, com $CE \subseteq C$.

$\alpha_{(Y/X)}$ e $\beta_{(Y)}$ denotam membros de LX; são pares de sequências de palavras, α e β, com categorias (possivelmente compostas), respectivamente Y/X e Y. α é um predicador, e β, um argumento; a categoria depois da barra do predicador é a categoria resultante quando este é concatenado a um argumento da categoria anterior à barra; se é uma barra comum, o argumento é esperado à direita; se é uma contrabarra, à esquerda. (Essencialmente, pode-se dizer que a regra $\beta_{(Y\backslash X)} \bullet \alpha_{(Y)} \rightarrow \beta\alpha_{(X)}$ corresponde às regras $X \rightarrow Y\backslash X\ Y$; em um formalismo de estrutura de constituintes.)

Uma expressão cuja categoria pertence a CE é uma sentença bem-formada.

A gramática categorial combinatória usada por ZETTLEMOYER & COLLINS (2005) contém algumas operações adicionais, e é por isso mais poderosa do que as gramáticas livres de contexto (especificamente, levemente sensível ao contexto — VIJAY-SHANKER e WEIR, 1994). Uma das principais características que a distinguem das gramáticas categoriais mais simples é a presença de uma anotação adicional aos itens do léxico. Além das categorias já descritas acima, geralmente usadas

para descrever a sintaxe, CCGs costumam trazer suas expressões anotadas também com tipos semânticos, e uma extensão das regras para o cálculo do tipo semântico de combinações de expressões. As regras são, portanto, da forma:

$$\alpha_{(X/Y)} : f \bullet \beta_{(Y)} : g \rightarrow A : f(g)$$

$$\beta_{(Y)} : g \bullet \alpha_{(X\backslash Y)} : f \rightarrow A : f(g)$$

As categorias *f*, *G* e *f*(*G*) são tipos semânticos. Dessa forma, o formalismo modela uma interface entre sintaxe e semântica. Além disso, permitem-se operações além da simples aplicação de funções. A compreensão dos elementos básicos, porém, é suficiente para entendimento do algoritmo de ZETTLEMOYER e COLLINS (2005). Referimos o leitor a STEEDMAN (2000) para uma descrição mais completa das gramáticas combinatórias categoriais.

ZETTLEMOYER e COLLINS (2005) utilizam uma CCG probabilística, anotando cada regra com uma probabilidade. A propagação das probabilidades é análoga à propagação em gramáticas livres de contexto. No artigo em questão, dados uma forma lógica *F*, uma derivação *D* e uma sentença *e*, os autores definem um vetor de *d* propriedades mapeado por uma função

$$\bar{f}(F, D, e) = \left\langle f_1(F, D, e), ..., f_d(F, D, e) \right\rangle$$

onde f_j é em geral a contagem de alguma subestrutura de (*F*, *D*, *e*). O modelo é parametrizado por um vetor $\theta \in \Re^d$. O vetor θ é estimado com uma variante do algoritmo Inside-Outside (MANNING e SCHÜTZE, 1999), o que torna possível atribuir uma probabilidade a uma regra da gramática.

Como o corpus de treinamento não inclui nenhuma informação sobre a derivação das formas lógicas, as derivações são tratadas como uma variável oculta. Como o formalismo é CCG, as regras são aplicações de esquemas fixos a um léxico de palavras e suas categorias; portanto, a questão é encontrar esses itens lexicais. Partindo

de um léxico inicial Λ_0, dado por uma base de dados sobre o domínio (nesse caso, relacionado à Geografia; portanto o léxico inicial contém itens como nomes de estados americanos), o algoritmo adquire automaticamente novos itens lexicais. Para isso, os autores definem a função *GENLEX*.

A função toma como insumo uma sentença *S* e uma forma lógica *L*, e gera um conjunto de itens lexicais (i.e., membros de *LX* na definição dada) que permitam pelo menos uma derivação de *S*. Uma função *C*(*L*), definida por um conjunto de regras construído manualmente, mapeia as expressões da forma lógica para categorias da gramática, estabelecendo uma correspondência entre o cálculo-lâmbda e a CCG. (Note-se que a possibilidade de estabelecimento dessa correspondência está diretamente ligada ao uso de uma representação geral; dificilmente seria possível fazer o mesmo com uma linguagem de domínio.) Para cada subestrutura de uma forma lógica *L* mapeada por *C*(*L*), uma categoria é criada. A função *GENLEX* então gera as possíveis combinações das palavras da sentença com as categorias correspondentes às expressões de sua forma lógica. Então, para um par *S*, *L*:

$$GENLEX(S, L) = \{x := y \mid x \in W(S), y \in C(L)\}$$

O algoritmo de aprendizado mantém armazenados no vetor $\bar{\theta}$ os valores associados a cada item do léxico. O conjunto de itens é dado por:

$$\Lambda^* = \Lambda_0 \cup \bigcup_{i=1}^{n} GENLEX(S_i, L_i)$$

O algoritmo tem dois passos: (1) buscar um pequeno conjunto de itens lexicais que seja suficiente para analisar todos os exemplos do treinamento, e (2) reestimar os valores das propriedades dos itens lexicais encontrados em (1). As propriedades consideradas foram todos lexicais, ou seja, referiam-se ao número de vezes que cada palavra da gramática aparecia na derivação.

Na *t*-ésima iteração do passo (1), cada sentença é analisada com os atuais alores $\left(\bar{\theta}^{t-1}\right)$ e um léxico provisório, específico da sentença i $\left(\Lambda_0 \cup GENLEX(S_i, L_i)\right)$. Os itens lexicais finais são então extraídos somente das análises mais prováveis encontradas nesse espaço. O passo (2) então reestima os valores levando em conta os itens lexicais que acabam de aumentar o léxico.

Os experimentos foram realizados sobre dois domínios: a base GEO880; e a base JOBS640, que contém 640 consultas para um banco de dados de anúncios de emprego. As traduções para o cálculo-lâmbda foram feitas manualmente a partir de anotações semânticas aos dados originais, feitas em estilo de Prolog. Os autores reportam os seguintes resultados:

Tabela 3. Resultados obtidos por ZETTLEMOYER e COLLINS (2005).

	Geo880	**Jobs640**
P	96,25%	97,36%
R	79,29%	79,29%

Como observam os autores em ZETTLEMOYER e COLLINS (2007), uma gramática desse tipo pode ser excessivamente rígida para permitir uma análise adequada da sintaxe da linguagem natural espontânea, uma potencial desvantagem. Por isso, no artigo mencionado, propõem modificações ao algoritmo apresentado para relaxar as restrições da gramática e torná-la mais robusta.

São introduzidas no formalismo operações não-padrão que (1) permitem capturar expressões em que a ordem das palavras difere da prevista na gramática e (2) presumir na análise a presença de palavras que não estão realizadas no *input*. Além disso, a estimativa dos parâmetros é modificada para permitir aprendizado *online*.

Para conseguir (1), os esquemas de regra apresentados na descrição do formalismo CCG foram estendidos; além de regras da forma

$$\alpha_{(X/Y)} : f \bullet \beta_{(Y)} : g \rightarrow A : f(g)$$

passa-se a ter regras da forma

$$\alpha_{(X\backslash Y)} : f \bullet \beta_{(Y)} : g \rightarrow A : f(g)$$

Note-se a inversão do sentido da barra: o argumento que era esperado à esquerda foi aceito à direita do functor. Um exemplo simples da utilidade da regra seria uma inversão como "vôo amanhã para Nova York" em vez de "vôo para Nova York amanhã".

Além disso, introduziram-se regras (cujo formato não discutimos porque para isso se exigiria uma discussão mais aprofundada do formalismo-base, fora do escopo deste trabalho, mas que são apresentadas em ZETTLEMOYER e COLLINS (2007)) que permitem inserir na forma lógica expressões que não têm nenhuma realização no *input* de linguagem natural.

Essas regras adicionais são usadas a um certo custo para a derivação, de forma a evitar a geração indiscriminada de análises inadequadas através das regras menos rigorosas. Para refletir esse custo, foram adicionados à função $\bar{f}$ dimensões para o número de utilizações de cada uma dessas regras na derivação, cujos parâmetros são estimados de forma a penalizar as regras na medida certa.

Além dessa modificação à gramática, ZETTLEMOYER e COLLINS (2007) também altera a forma como o vetor $\bar{\theta}$ é atualizado na proposta anterior (ZETTLEMOYER e COLLINS, 2005). A principal mudança é que, enquanto anteriormente cada sentença era analisada com parâmetros aprendidos até cada iteração, e um léxico provisório, $\left(\bar{\theta}_i, \Lambda_0 \cup GENLEX(S_i, L_i)\right)$, o novo algoritmo atualiza a gramática e o vetor de parâmetros exemplo a exemplo, e apenas se for necessário. Cada sentença é primeiro analisada com o par $\left(\bar{\theta}_{i-1}, \Lambda_{i-1}\right)$, e somente se não se obtém uma análise correta são realizados os passos de extensão do léxico e atualização dos parâmetros.

Os resultados reportados, para 4978 sentenças da base ATIS, de informações sobre vôos nos Estados Unidos, são precisão de 90,61% e revocação de 81,92%.

Em um trabalho ainda mais recente (ZETTLEMOYER e COLLINS, 2009), os autores introduzem uma abordagem que leva em conta o contexto linguístico da sentença (em vez de serem analisados isoladamente, os exemplos do corpus foram divididos em sequências discursivas representando interações com o sistema), permitindo que a forma lógica incorpore, por exemplo, entidades do discurso que não foram realizadas naquela sentença, mas em outra anterior — como no caso de "Vôos para Boston amanhã. O [vôo para Boston] que for mais cedo."

4.4. Análise semântica com classificadores

KATE e MOONEY (2006) formulam um método de aprendizado de analisadores semânticos baseado em classificadores. Seu sistema, KRISP, assim como os outros aqui discutidos, toma como *input* um corpus paralelo de sentenças em linguagem natural e árvores de uma linguagem formal. As regras da gramática da linguagem formal são tomadas como conceitos semânticos, e um classificador SVM (CRISTIANINI e SHAWE-TAYLOR, 2000) é treinado para cada uma delas, de maneira a estimar, para subsequências das sentenças em linguagem natural, sua probabilidade de serem recobertas por aquela regra da gramática formal.

O objetivo da ferramenta treinada é produzir derivações semânticas de sentenças em linguagem natural; uma derivação é uma árvore na gramática formal cujos nós estão anotados com fatores da sentença. As *substrings* recobertas pelos filhos de um nó não podem ter interseção, a *substring* recoberta por um nó deve ser a concatenação daquelas recobertas por seus filhos, não necessariamente em ordem de precedência. A probabilidade de uma derivação *D* é dada por:

$$P(D) = \prod_{\pi, s_j^i \in D} P_\pi(s_j^i)$$

onde s_j^i é uma *substring* da sentença, e (π, s_j^i) é um par denotando um nó anotado com essa *substring*. Essa probabilidade pode ser calculada recursivamente a partir das probabilidades das subárvores, E^*_{n,s_j^i}, onde *n* é a raiz da subárvore e s_j^i é a *substring* que esta recobre.

$$E^*_{n,s_j^i} = \underset{\substack{\pi = n \to n_1 \ldots n_t \in G, \\ (p_1 \ldots p_t) \in partition(s_j^i, t)}}{\arg\max} (P_\pi(s_j^i) \prod_{k=1}^{t} P(E^*_{n_k, p_k}))$$

onde $partition(s_j^i, t)$ é uma função que retorna o conjunto de todas as partições de s_j^i em *t* elementos. A recursão é implementada com uma versão modificada do algoritmo de Earley, descrita em KATE (2007).

As probabilidades $P_\pi(u)$ são estimadas com classificadores SVM; esses classificadores encontram o hiperplano de maior margem que separa dois conjuntos de exemplos em um espaço vetorial; os positivos, e os negativos. Na aplicação em tela, os exemplos positivos são as sentenças do corpus paralela onde a derivação da tradução da sentença contém a regra em questão, e os negativos são todas as outras sentenças. Para cada regra, um classificador é treinado iterativamente, com o *kernel* definido como o número de subsequências comuns entre duas *strings*. Em cada iteração, as probabilidades estimadas são usadas para tentar obter as representações das sentenças, e com isso são gerados novos exemplos para a próxima iteração; os exemplos negativos são acumulados, e os positivos são substituídos.

Embora os classificadores sejam binários, a probabilidade de um exemplo ser coberto por uma regra é dada pela sua distância do hiperplano do SVM para aquela regra, normalizada para um intervalo de [0,1].

O sistema aceita a pré-definição das traduções de certos elementos.

KATE e MOONEY (2006) apresentam os resultados de seus experimentos em gráficos.

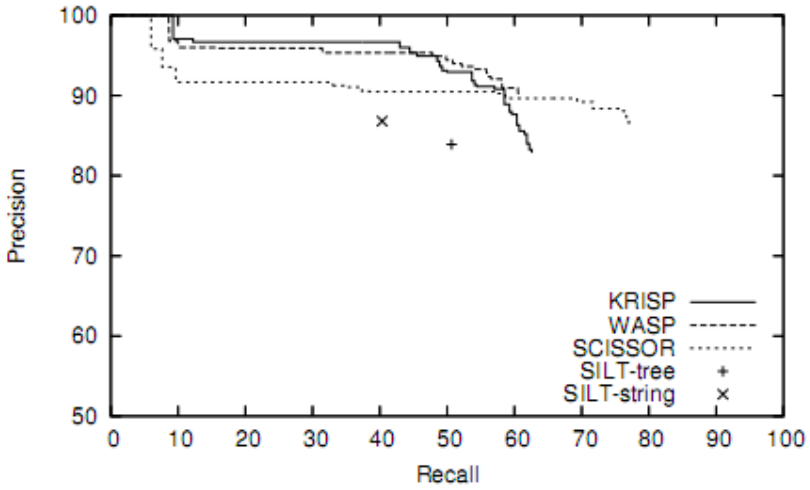

Figura 5. Resultados de KATE e MOONEY (2006) para o corpus CLang, comparados a outros *parsers* semânticos.

Em KATE e MOONEY (2007) os autores introduzem o conceito de supervisão ambígua, e com ele uma nova versão de sua solução. Em vez de usar um corpus contendo pares de sentenças e suas respectivas traduções em linguagem formal, o que consideram supervisão não ambígua, os autores agora utilizam uma relação entre os dois conjuntos de sentenças na qual uma sentença pode estar associada a mais de uma possível tradução, embora só haja uma correta.

O algoritmo anterior foi então modificado de maneira a (1) minimizar a ambiguidade, removendo do conjunto de possíveis traduções de cada sentenças aquelas que, por inconsistência lógica, não poderiam ser corretas — se T_k é a única tradução de S_i, não pode constar entre as traduções de S_j, por exemplo — e (2) atribuir pesos aos exemplos informados aos classificadores, inversamente proporcionais ao número de possibilidades para a sentença.

KATE (2008) investiga como usar transformações da gramática da linguagem formal para melhorar a performance de seu analisador semântico. Como as gramáticas de linguagens de domínio não são em geral projetadas com uma preocupação com a sua relação com a semântica da linguagem natural, sistemas de análise semântica que partem dessas gramáticas podem enfrentar problemas. O autor propõe transformações que sobre as gramáticas e avalia em experimentos os benefícios de sua introdução automática. As transformações, que exceto (4) preservam equivalência fraca, são:

(1) criação de um não-terminal a partir de um terminal: todas as ocorrências do terminal são substituídas pelo não-terminal, e uma regra lexical é introduzida;

(2) aglomeração de não-terminais: é introduzido um novo não-terminal e *n* regras nas quais pode ser reescrito como *n* não-terminais da gramática original;

(3) combinação de não-terminais: dois terminais t_1 e t_2 são combinados e uma regra $X \rightarrow t_1\ t_2$, e todas as ocorrências da sequência são substituídas por *X*;

(4) remoção de não-terminais: quando o mesmo não-terminal aparece duas vezes no *RHS* de uma regra, a segunda ocorrência é deletada, se for verificado no corpus que as subárvores introduzidas por ambas são com frequência as mesmas;

(5) eliminação de regra: uma regra é eliminada e são geradas novas regras em que todas as ocorrências do seu *LHS* em outras regras da gramática original são substituídas pelo seu *RHS*.

As transformações são aplicadas para melhorar um *parser* treinado com o alinhamento de palavras e regras; para cada regra, é computado um escore do número

de usos errôneos da regra em derivações formais geradas pelo *parser*. Se o escore e o total excedem limites estabelecidos, as operações são aplicadas. Se uma nova avaliação determina que o escore de erros é maior depois de uma transformação, esta é revertida.

Uma aplicação no corpus CLANG levou a um aumento de 50% do *F-score*.

4.5. Análise semântica com gramáticas sincrônicas

WONG e MOONEY (2006) apresentam a tradução com gramáticas sincrônicas extraídas de corpora paralelos como uma solução para o problema da tradução de linguagens naturais em linguagens formais. O algoritmo proposto aprende uma gramática sincrônica probabilística a partir de um corpus paralelo de sentenças em linguagem natural e suas traduções na linguagem formal, anotadas com as derivações sintáticas. A ferramenta WASP consiste de uma gramática sincrônica, *G*, e um modelo probabilístico, parametrizado por um vetor λ, que atribui a uma possível derivação em *GS* a probabilidade de estar correta para uma sentença *S* na linguagem-fonte. A tradução pode ser definida como:

$$f^* = m(\arg\max_{d \in D(S \underset{GS}{\Rightarrow}^* e)} P_\lambda(d \mid e)$$

onde *M(d)* é a tradução (em linguagem formal) gerada por uma derivação *d*, e $D(S_{GS} \underset{GS}{\Rightarrow}^* e)$ é o conjunto de derivações de *e* segundo *GS*. As regras de *GS* e o vetor λ são aprendidos a partir de um conjunto de exemplos, $\{\langle e_i, f_i \rangle\}$, onde cada exemplo

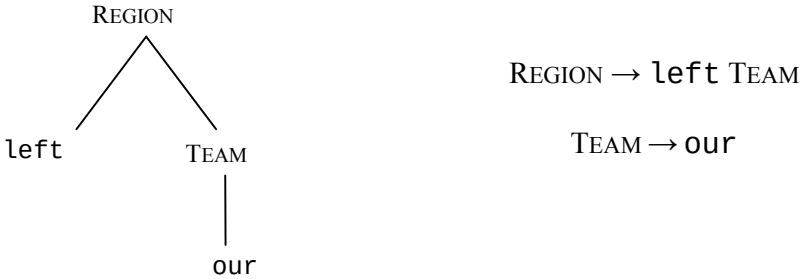

Figura 6. Extração de regras da árvore de derivação.

$\langle e_i, f_i \rangle$ é uma sentença em linguagem natural, e_i, e sua tradução f_i. É preciso dispor também da gramática da linguagem formal, G_F, e de um alinhamento Ω.

A ideia básica é formar um léxico bilíngue a partir do alinhamento Ω, utilizando os pares mais prováveis encontrados por um alinhador. WONG e MOONEY (2006) utilizam o GIZA++ (OCH e NEY, 2003) configurado para uso do Modelo IBM 5 (BROWN *et al.*, 1993). As sentenças em linguagem formal são então analisadas sintaticamente segundo a gramática dada, chegando-se a um corpus paralelo de sentenças e árvores, anotado com alinhamentos.

Os autores observam que as palavras da linguagem formal podem ser semanticamente vazias (como é o caso de símbolos de pontuação, como parênteses) ou, em outros casos, polissêmicas, levando a mais de um correspondente em linguagem natural; ambos os fenômenos podem levar a maus alinhamentos. Para contornar essa dificuldade, os alinhamentos em WONG e MOONEY (2006) são feitos não entre as palavras de ambas as linguagens, mas entre palavras em linguagem natural e nós sintáticos das árvores da linguagem formal, representados linearmente para uso do alinhador.

A partir desse insumo são extraídas as regras de derivação sincrônica. De cada não-terminal da árvore é extraída uma regra; o próprio não-terminal é o *LHS*, e seus filhos, em ordem de precedência, formam o *RHS*.

De baixo para cima (ou seja, começando com as arestas terminais e prosseguindo em direção à raiz), as árvores da linguagem formal são analisadas e regras são extraídas de cada aresta. Para cada regra $X \rightarrow \beta$, é extraída a regra $X \rightarrow \langle \alpha, \beta \rangle$, onde α são as palavras alinhadas ao nó *X* e os nós não-terminais filhos de *X*, na ordem ascendente de precedência sintática, ou seja, na ordem em que seus rótulos ou os rótulos de seus cantos esquerdos ocorrem na *string*. Portanto, dado o alinhamento:

$$\langle our, \text{TEAM} \rightarrow \texttt{our} \rangle$$

se extrai a regra:

$$\text{TEAM} \rightarrow \langle our, \texttt{our} \rangle$$

E dados os alinhamentos (entre as strings *our half* e `half our`):

$$\langle our, \text{TEAM} \rightarrow \texttt{our} \rangle$$

$$\langle half, \text{REGION} \rightarrow \texttt{half TEAM} \rangle$$

se extraem as regras:

$$\text{TEAM} \rightarrow \langle our, \texttt{our} \rangle$$

$$\text{REGION} \rightarrow \langle \text{TEAM}_1\ half, \texttt{half TEAM}_1 \rangle$$

Note-se que, na segunda regra, os nós foram reordenados porque o exemplo do qual foram extraídas mostra que, na linguagem natural, a ordem correta é *our half*, de modo que TEAM precede sintaticamente *half*. Os índices que anotam os não-terminais representam o mapeamento entre estes.

Quando, na extração das regras, o escopo de um nó inclui palavras da linguagem natural que não foram alinhadas com nenhum nó, é inserido na regra um símbolo especial, *G(N)*, que denota uma lacuna de comprimento *N* e pode ser reescrito, em uma derivação, como até *N* palavras quaisquer da linguagem natural.

WONG e MOONEY (2006) identificam dois casos em que o algoritmo não extrai uma regra: o primeiro, em casos onde nenhum dos descendentes de um nó foi alinhado; isso ocorre, segundo os autores, quando um determinado conceito denotado na linguagem formal não é expresso explicitamente na linguagem natural, levando a regras do tipo

$$X \rightarrow \langle \varepsilon, \alpha \rangle$$

O segundo ocorre quando o escopo de um nó *N* inclui uma palavra alinhada a um nó *M* que não é dominado por *N*. Isso pode acontecer quando existem diferenças

grandes entre a sintaxe de uma linguagem e de outra. Quando se tem um caso como o das *strings our left penalty area* e `left penalty-area our`, que têm os alinhamentos:

$$\langle our, \text{TEAM} \leftrightarrow \texttt{our} \rangle$$

$$\langle left, \text{REGION} \leftrightarrow \texttt{left REGION} \rangle$$

$$\langle penalty, \text{REGION} \leftrightarrow \texttt{penalty-area TEAM} \rangle$$

$$\langle area, \text{REGION} \leftrightarrow \texttt{penalty-area TEAM} \rangle$$

o procedimento descrito leva à extração das regras

$$\text{REGION} \rightarrow \langle left\ \text{REGION}_1, \texttt{left REGION}_1 \rangle$$

$$\text{REGION} \rightarrow \langle \text{TEAM}_1\ penalty\ area, \texttt{penalty-area TEAM}_1 \rangle$$

$$\text{TEAM} \rightarrow \langle our, \texttt{our} \rangle$$

Essas regras não são adequadas, porque, por serem livres de contexto, não capturam o fato de que na linguagem natural haverá uma palavra entre a expansão de TEAM e os terminais *penalty area*. A solução apresentada é um achatamento da gramática, fazendo uma fusão dessa regra com outra que a domina. Chega-se assim a:

$$\text{REGION} \rightarrow \langle \text{TEAM}_1\ left\ penalty\ area, \texttt{left penalty-area TEAM} \rangle$$

$$\text{TEAM} \rightarrow \langle our, \texttt{our} \rangle$$

Esse tipo de operação leva a regras menos gerais na gramática. Para reduzir a necessidade da transformação, os autores descartam os alinhamentos mais problemáticos, contando quantas transformações são induzidas por cada par e subtraem alinhamentos, de modo a maximizar

$$v(\Omega) - v(\Omega \setminus \{a\})$$

a é um par do alinhamento e $v(\Omega)$ é o somatório do número de transformações induzido por cada par no alinhamento Ω. Quando a probabilidade do alinhamento é maior do que um limite, porém (0,9 no artigo), o alinhamento não pode ser removido.

WONG e MOONEY (2006) definem então um modelo para definir a distribuição de probabilidade sobre as derivações obtidas das *strings* de linguagem natural observadas a partir do método de extração de regras. A distribuição a ser obtida é

$$P_\lambda(d \mid e) = \frac{1}{Z_\lambda(e)} \exp \sum_i \lambda_i f_i(d)$$

onde f_i é uma função que retorna o valor das propriedades e $Z_\lambda(e)$ é um fator normalizador. As propriedades são dos seguintes tipos:

(1) para cada regra da gramática, há uma propriedade que indica quantas vezes a regra é usada em uma derivação;

(2) para cada palavra de *e*, há uma propriedade que indica quantas vezes *e* é gerada a partir de um símbolo especial *G(N)* em uma derivação;

(3) para cada derivação, há uma propriedade que indica o total de palavras geradas a partir de um símbolo especial *G(N)*.

Os parâmetros de cada propriedade são estimados maximizando-se a *log-likelihood* condicional do corpus de treinamento (WONG, 2007).

WONG e MOONEY (2006) não apresentam numericamente os resultados de seus experimentos, mas os resultados reportados para os mesmos experimentos em WONG (2007) são[4]:

Tabela 4. Resultados obtidos pelo sistema WASP.

	Geo880	**RoboCup**
P	87,2%	88,9%
R	74,8%	61,9%

WONG e MOONEY (2007) apresentam uma extensão ao formalismo de gramáticas sincrônicas que permite a derivação de formas lógicas expressas no cálculo-

[4] WONG (2007) nota que sua avaliação foi mais rigorosa do que a apresentada em ZETTLEMOYER e COLLINS (2007). O autor afirma que, ao utilizar a mesma metodologia de avaliação deste, obteve resultados semelhantes.

λ, introduzindo uma forma sistemática de tratar variáveis lógicas. As regras sincrônicas são representadas como:

$$X \rightarrow \alpha, \lambda x_1 ... \lambda x_k \beta$$

Essa extensão, que não discutimos em detalhes, permite trabalhar com um conjunto de linguagens maior do que aquele passível de tratamento no formalismo anterior.

Passamos agora a uma proposta de solução baseada na abordagem de WONG e MOONEY (2006).

Capítulo 5. Um método de extração de gramáticas sincrônicas

Neste capítulo, apresentamos uma formulação precisa do problema proposto e um passo-a-passo da solução, incluindo a discussão de alguns desafios surgidos e das estratégias adotadas para superá-los.

5.1. Descrição do problema

Este trabalho propõe uma abordagem ao problema de encontrar uma representação semântica em uma linguagem formal para um enunciado em linguagem natural. Uma tal representação pode ser obtida pelo uso de uma gramática sincrônica que recubra a correspondência entre as duas linguagens. Apresenta-se, portanto, um procedimento para aprendizado automático de uma gramática sincrônica para duas linguagens dadas, uma natural e uma formal e livre de contexto, e de uma ponderação da gramática que permita escolher a melhor tradução onde houver ambiguidade.

O método proposto recebe como insumo:

(1) um corpus de árvores de derivação em uma linguagem formal (doravante o conjunto MRS e a linguagem MRL);

(2) um corpus de sentenças em linguagem natural (NS, NL);

(3) uma gramática para a linguagem MRL (G_{MRL});

(4) um alinhamento (Ω), que pode ser

   a. entre palavras;

   b. entre palavras e nós terminais de G_{MRL};

   c. entre palavras e regras de G_{MRL}.

O resultado do procedimento é uma gramática sincrônica ponderada GS<*NL*, *MRL*> = (*N*, Σ, Φ, R, *S*). A gramática G_{NL} tal que G_{NL} = (*N*, Σ, R_1, *S*), onde

$$R_1 = \{n \rightarrow \alpha \mid A \rightarrow \langle \alpha, \beta \rangle \in R\}$$

será chamada a gramática-fonte, e gera linguagem natural (com as regras extraídas pelo algoritmo). A gramática G'_{MRL} que G'_{MRL} = (*N*, Σ, R_2, *S*), onde

$$R_2 = \{n \rightarrow \beta \mid A \rightarrow \langle \alpha, \beta \rangle \in R\}$$

será chamada a gramática-alvo, a gramática extraída para a linguagem formal. A gramática dada da linguagem formal, que é recebida como *input*, será designada simplesmente de gramática formal (G_{MRL}).

5.2. Etapas da solução

O algoritmo básico é

(1) fazer a **análise sintática das sentenças em MRS**, encontrando suas árvores de derivação;

(2) aplicar **filtros** a palavras em ambos os corpora que possam ser descartadas ou substituídas por rótulos genéricos;

(3) produzir um **alinhamento** Ω e obter árvores alinhadas, estendendo-as com os alinhamentos dados por Ω;

(4) projetar uma árvore virtual, com a subárvore não-terminal isomórfica à da árvore real e os terminais definidos com base nos alinhamentos obtidos, cuja fronteira é a tradução em NL da sentença inicial, e **identificar os nós virtuais problemáticos** que induzem regras não-livres-de-contexto na gramática-alvo;

(5) aplicar operações de **transformação da árvore** aos nós identificados, obtendo uma árvore livre de contexto;

(6) **reordenar os nós** nas árvores das sentenças-alvo;

(7) **extrair regras sincrônicas** livres de contexto da árvore alinhada;

(8) utilizar as regras para gerar **derivações sincrônicas** dos pares de strings pertences a $NS \times MRS$;

(9) **ponderar as regras sincrônicas** com sua probabilidade, calculada com base na frequência de seu uso em traduções corretas.

5.2.1. Análise sintática das sentenças em MRS

A análise sintática das sentenças em linguagem formal visa obter árvores de derivação que possam ser transformadas em árvores de derivação das traduções em linguagem natural. Qualquer algoritmo de análise sintática que se aplique à gramática-fonte do *input* pode ser utilizado para este passo. Nos experimentos apresentados, utilizou-se como base um reconhecedor *top-down* da classe LL(*) (PARR, 2007).

O reconhecedor foi modificado para realizar análise sintática e retornar a árvore de derivação da sentença (que é única, pois G_{MRL} é não-ambígua).

Para a obtenção dos alinhamentos, usaram-se duas representações diferentes das árvores: uma linearização simples, da esquerda para a direita, com os nós representados pelos seus rótulos, para o alinhamento de palavras com nós; e uma linearização similar com os nós representados pela concatenação de seus rótulos com os de seus filhos, para o alinhamento de palavras com regras.

5.2.2. Filtros

Trabalhando com a premissa de que alinhamentos expressam uma correlação semântica entre os termos alinhados (sejam eles palavras, regras ou nós), percebe-se que nem sempre é interessante que um termo faça parte do insumo do alinhador. WONG (2007) observa, por exemplo, que nem todas as palavras da MRL produzem bons alinhamentos; algumas por serem polissêmicas e encontrarem correspondentes em diferentes expressões da linguagem natural; outras por serem, ao contrário,

semanticamente vazias, e não terem nenhum correspondente explícito em suas traduções. Os mesmos fenômenos ocorrem com as palavras da linguagem natural: certas palavras podem corresponder a diferentes termos da linguagem formal em diferentes contextos, enquanto outras podem expressar conceitos que estão fora do domínio da linguagem formal e não são nela enunciados.

A presença desses termos no insumo fornecido ao algoritmo de alinhamento influencia negativamente o resultado deste. Para lidar com esse problema, propomos o uso de três tipos de filtros no corpus paralelo.

O primeiro é um simples filtro de remoção, que pode ser aplicado em ambas as linguagens. Um conjunto de termos pré-definidos (por enumeração ou com expressões regulares) é removido do corpus e a versão filtrada é passada para o alinhador. Exemplos claros de palavras cuja remoção pode ser benéfica são sinais de pontuação (em ambas as linguagens) ou artigos da linguagem natural, como <o>, <um>, <*the*>, etc.. Como essas palavras, que são parte da linguagem, deverão aparecer nas traduções geradas, todos os termos filtrados da linguagem formal são preservados nas árvores de derivação e aparecem nas regras extraídas; mesmo assim, o uso do filtro antes do processamento pelo alinhador garante, por exemplo, que nenhuma palavra da linguagem formal será alinhada a símbolos como <,>, e evita que as probabilidades dos outros alinhamentos do par sejam afetadas por esse mau alinhamento. As palavras filtradas da linguagem natural não precisam ser reintegradas às derivações, pois podem ser filtradas do *input* quando o analisador for utilizado.

Os dois outros tipos de filtro não removem a palavra filtrada, mas a substituem por um rótulo de classe. Esses filtros são aplicáveis somente à linguagem-fonte.

O segundo tipo de filtro se aplica a grupos pré-definidos de palavras e as substitui pelo rótulo do grupo. A ideia por trás do filtro se baseia na premissa de que há

casos em que um conjunto de palavras (ou expressões) pode receber exatamente o mesmo tratamento quando se traduz a sentença para a linguagem formal. Isso pode acontecer simplesmente porque as expressões são efetivamente sinônimas na linguagem natural, mas é especialmente útil para definir classes de palavras que, embora não equivalentes em linguagem natural, apresentam distinções de significado que perdem a relevância no domínio ou no formalismo da linguagem formal para a qual serão traduzidas.

Outra situação em que se pode dispensar o mesmo tratamento a vários termos é quando existe uma função trivial para traduzi-los para a linguagem formal, como é o caso de números e nomes próprios. Essa função pode ser inserida diretamente como regra na gramática sincrônica, sem necessidade de extração a partir de exemplos. Além de não ser necessário tratar cada instância dessas classes como um novo caso para o qual extrair uma regra de tradução, pode-se observar que, do ponto de vista sintático, elas são intercambiáveis, de maneira que a unificação dos exemplos sob um mesmo rótulo no corpus processado pelo alinhador ajuda a capturar com maior precisão as regras que se aplicam a todos os membros da classe.

Os rótulos resultantes podem ser filtrados com um filtro de remoção, se cabível. Adjetivos, por exemplo, podem constituir uma classe que não encontra expressão em determinada linguagem de domínio. Sendo esse o caso, pode ser interessante removê-los do corpus do alinhador e do *input* do analisador sincrônico.

O terceiro tipo de filtro é um caso especial do segundo em que as classes definidas têm significância linguística e seus membros podem ser identificados automaticamente. Tratam-se de filtros de lematização e radicalização (*stemming*) de palavras e de rotulação com classes morfossintáticas (*POS-tagging*). Um lematizador identifica a forma canônica de uma palavra em determinada linguagem (informalmente,

"a forma como a palavra aparece no dicionário" — <meninos> e <menina>, por exemplo, seriam ambos reduzidos a <menino>). Um radicalizador remove sufixos e prefixos e retorna o radical do termo linguístico ("menin-" para ambos os exemplos anteriores; note-se que a lematização de <meninice>, porém, é <meninice>, mas seu radical é também "menin-"). A identificação de classes morfológicas atribui um rótulo que representa uma classe morfossintática da língua, como "Preposição", "Verbo" ou (como é o caso dos exemplos) "Substantivo".

Todas essas classes podem ser relevantes na definição de alinhamentos e de regras de tradução para certos pares de linguagens. É preciso observar que (exceto talvez no caso de números e nomes próprios) nenhum filtro é útil *a priori*. Até mesmo um símbolo como <,> pode ser significativo na tradução para uma linguagem formal (na tradução de listas, por exemplo). O valor do uso de um filtro varia com as linguagens em questão. A escolha dos filtros usados nos experimentos foi feita a partir da análise da linguagem de domínio. Não é, porém, uma análise custosa.

5.2.3. Alinhamento

Para contornar os problemas causados pelo fato de que nem todas as palavras da linguagem formal podem ser alinhadas de forma significativa com as palavras da linguagem natural, WONG e MOONEY (2006) propõem que o alinhamento seja feito usando as derivações na linguagem formal, em vez de suas fronteiras. Os autores constroem a partir de MRS um corpus MRS', no qual cada sentença é transformada na sequência de regras da gramática de MRL que compõem sua derivação. Dessa forma, obtém-se um alinhamento entre palavras da linguagem natural e regras da linguagem formal.

Neste trabalho, os experimentos utilizam três formas diferentes de alinhamento: o alinhamento entre palavras, o alinhamento entre palavras e regras, e o alinhamento

entre palavras e nós. Estes últimos se obtêm com as representações linearizadas das árvores de derivação mencionadas na seção 5.2.1. . A diferença entre a segunda e a terceira formas de alinhamento está no fato de que, enquanto na segunda cada símbolo da sequência representa uma regra, na terceira cada símbolo representa um único nó terminal ou não-terminal. A segunda representação faz uma distinção mais fina; além disso, cada regra corresponde a um nó não-terminal, mas na linearização em nós representam-se tanto os não-terminais quanto os terminais.

É importante notar que, para extração das regras sincrônicas, o alinhamento de NS para MRS deve ser uma relação funcional entre os conjuntos de símbolos Σ e Φ:

$$\Omega : \Sigma \rightarrow \Phi$$

onde, dada G_{MRL} = (N, T, R, S), Φ pode ser igual a N, T ou R, de acordo com a forma de alinhamento escolhida.

Se uma mesma palavra da linguagem natural for alinhada a mais de uma palavra, regra ou nó da linguagem formal, serão extraídas regras que geram a palavra em dois passos diferentes da derivação, embora esta ocorra somente uma vez. Como a gramática é livre de contexto, não é possível expressar diretamente na regra extraída que a palavra deve ser gerada somente quando todos os nós a ela alinhados já tiverem sido derivados (exceto se for gerada ao mesmo tempo que estes). Portanto, nos restringimos a usar alinhamentos gerados sob essa condição.

5.2.4. Identificação dos nós problemáticos

Definimos como "nós problemáticos" os nós dos quais não se pode, por um dos critérios delineados nesta seção, extrair uma regra aceitável na gramática-fonte da gramática sincrônica, G_{NL}.

Dispondo das árvores de derivação e os alinhamentos, podem-se produzir as árvores alinhadas. Estas são simplesmente as árvores de derivação, estendidas com

anotações em cada nó contendo o conjunto (possivelmente vazio) de posições da sentença de linguagem natural a que o nó está alinhado. Todas as possibilidades de alinhamento apresentadas podem ser facilmente representadas dessa forma; no caso do alinhamento entre palavras com regras $X \rightarrow \alpha$, as anotações são feitas no nó *X*.

A árvore alinhada permite a projeção da árvore de derivação da sentença-alvo. Conforme a definição apresentada, para extração de regras sincrônicas a partir de um par de árvores é preciso que as subárvores não-terminais da árvore-fonte e da árvore-alvo sejam isomórficas, e além disso que ambas sejam livres de contexto.

É sempre possível fazer uma projeção que garanta a propriedade de isomorfismo das subárvores não-terminais. Seja *N* um nó não-terminal da árvore-fonte. A projeção da árvore-alvo se faz percorrendo-se a subárvore não-terminal da árvore-fonte e efetuando-se os seguintes passos para cada nó visitado:

(1) cria-se um nó de mesmo rótulo e identificador na árvore-alvo;

(2) cria-se um nó terminal filho do nó-alvo para cada alinhamento anotado no nó-alvo;

(3) percorre-se a lista dos filhos do nó-fonte:

   a. para cada filho terminal do nó-fonte, é criado um nó terminal filho do nó-alvo para cada alinhamento anotado no terminal;

   b. para cada filho não-terminal do nó-fonte, volta-se ao passo (1).

A aplicação do passo (1) garante o isomorfismo entre as subárvores não-terminais, já que cada nó não-terminal na árvore-alvo é criado se e somente se existe um nó correspondente na árvore-fonte, e assim para todos os seus filhos. No entanto, essa transformação não garante que a árvore resultante seja livre de contexto. Sejam *X* e *Y* dois nós não-terminais da árvore-alvo, tais que *X* precede sintaticamente *Y*. Como os filhos de *X* e *Y* são criados com orientação ao alinhamento, pode surgir uma

configuração em que, na árvore projetada, existe um nó dominado por *X* que precede

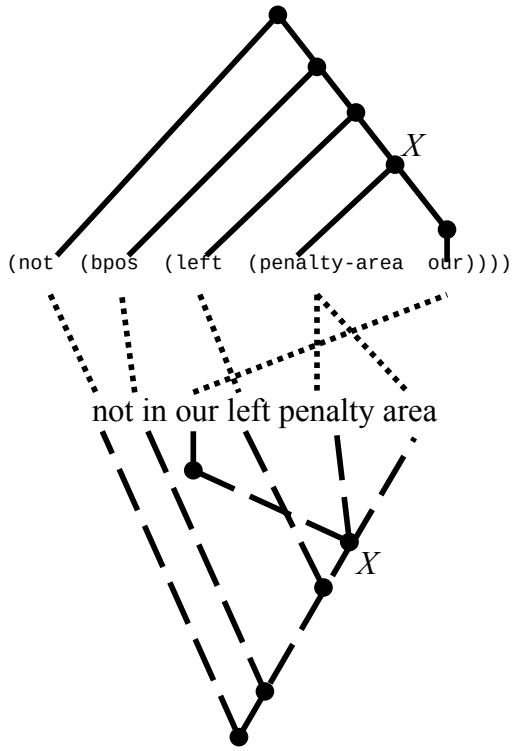

Figura 7. Uma árvore alinhada com árvore virtual projetada e um nó problemático, Y.

sintaticamente os nós dominados por *Y*, como na Figura 7, em que `left` precede sintaticamente `our`, dominado por *X*, embora *X* preceda sintaticamente `left`.

Nesse caso, não é possível extrair uma regra desse nó, pois nenhuma fronteira interna que se possa extrair será uma *substring* de uma fronteira interna da árvore.

As anotações sobre os alinhamentos são suficientes para identificar os nós problemáticos. Demonstramos que, dado um alinhamento conforme caracterizado, um nó da árvore projetada induz uma regra não-livre-de-contexto somente se seu escopo é descontínuo. Sejam *X*, *Y*, *u* e *v* nós da árvore-alvo projetada, tais que *X* e *Y* não estão no mesmo caminho. Se *X* precede sintaticamente *Y* mas nem todos os nós dominados por *X* precedem sintaticamente todos os nós dominados por *Y*, existe pelo menos um nó *u* dominado por *X* e um nó *v* dominado por *Y* tal que *v* precede sintaticamente *u*. Como *X* e *Y* não estão no mesmo caminho e os terminais da árvore são criados a partir de

palavras alinhadas a um único nó, *u* e *v* não podem ser dominados simultaneamente por *X* e *Y*; logo, *X* não domina *v*. Sabe-se ainda que *X* tem um canto esquerdo que precede sintaticamente *u*, ou não poderia preceder sintaticamente *Y*. Nesse caso, como *v* tem posição inferior a *u* e superior ao canto esquerdo de *X*, *X* tem um escopo descontínuo. Se *X* e *Y* estão no mesmo caminho, mas existe

Note-se, porém, que um nó pode ter escopo descontínuo e não induzir uma regra não-livre-de-contexto, pois é possível que haja uma palavra não alinhada cuja posição esteja contida no fechamento de seu escopo.

Portanto, *X* induz uma regra não-livre-de-contexto se:

$$\exists w \in NL \mid w \in e(N) \setminus \xi(N), \langle w, w' \rangle \in \Omega, w' \in \Sigma(MRL).$$

Assim, a determinação do escopo de um nó e uma consulta à lista de palavras alinhadas pode determinar se o nó induz uma regra não-livre-de-contexto. Os nós identificados precisarão sofrer transformações, discutidas na seção 5.2.5. , para que a árvore-alvo seja livre de contexto.

Existem ainda dois casos em que um nó é considerado problemático. A análise sintática das sentenças em linguagem natural será feita com a gramática-fonte da gramática sincrônica extraída. Essa gramática, portanto, não pode conter ciclos, dado que estes, por permitirem um número arbitrário de repetições, podem permitir que uma sentença tenha infinitas derivações possíveis (HOPCROFT e ULLMAN, 1979).

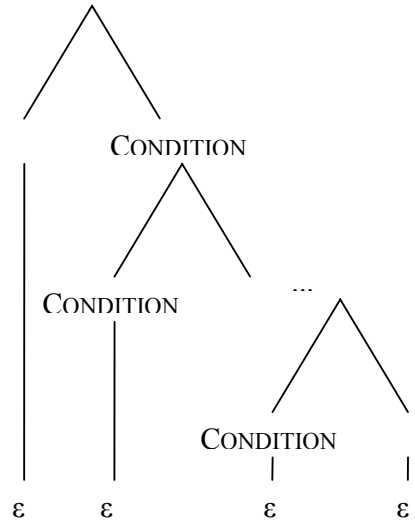

Figura 8. Um ciclo formado com uma regra-épsilon.

A existência de um ciclo em uma gramática pode ocorrer em dois casos. O primeiro caso é se existe uma regra $X \rightarrow \varepsilon$ e uma regra ou conjunto de regras que permita derivar sequências arbitrariamente longas de repetições do símbolo *X*, que podem então ser eliminadas com a regra-ε (Figura 8).

Mesmo que a gramática da linguagem formal, a partir da qual é construída a gramática sincrônica, não contenha ciclos, estes podem surgir na gramática-fonte com o alinhamento de nós. Se um nó que tiver somente filhos terminais na gramática original não for alinhado a nenhuma palavra da linguagem natural, será extraída deste uma regra-ε na gramática-fonte.

Se a gramática não tem regras-ε, pode ainda assim conter um ciclo se existir um conjunto de regras de um único não-terminal no lado direito tais que:

$$X \underset{G}{\Rightarrow}^{+} Y$$

$$Y \underset{G}{\Rightarrow} X$$

como na Figura 9.

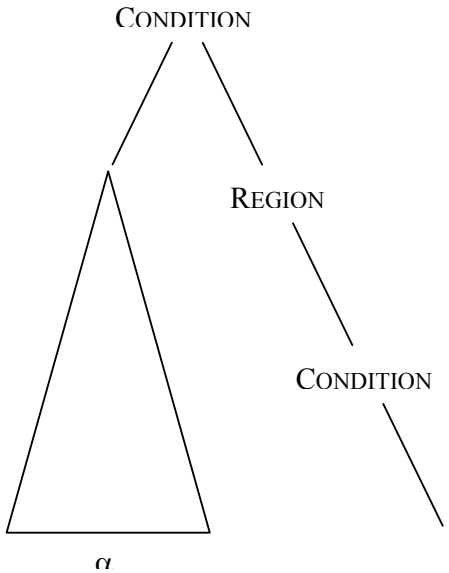

Figura 9. Um ciclo formado com uma cadeia de não-terminais.

Não basta, portanto, garantir a propriedade da G(MRL); é preciso eliminar os nós que podem levar à extração de regras-ε ou regras que formem ciclos. Seja um alinhamento direto um alinhamento com o próprio nó, com a regra encabeçada pelo nó, ou com um dos filhos terminais do nó. Portanto, definimos mais dois casos em que um nó é considerado problemático:

(1) se tiver somente filhos terminais e nenhum alinhamento direto;

(2) se tiver um único filho não-terminal, um conjunto não-vazio de filhos terminais, e nenhum alinhamento direto.

O primeiro caso leva forçosamente à extração de uma regra-ε na gramática-fonte, na medida em que o nó é mantido, mas não tem filhos não-terminais e nenhum alinhamento será inserido como filho terminal na árvore-fonte.

O segundo caso pode levar a um ciclo. Na ausência de regras-ε, um ciclo só pode ser formado por um encadeamento de regras que reescrevem um não-terminal como outro não-terminal, como exemplificamos na Figura 9. Evitamos assim as regras cujo *RHS* na gramática-fonte tem um único não-terminal. No entanto, sabe-se que as

regras já presentes na gramática formal não levam a ciclos; portanto, somente são considerados problemáticos os nós que levam a

$$X \rightarrow \alpha, \beta \text{ se } |\alpha| \neq |\beta|$$

Se α e β têm o mesmo comprimento, a regra já existia na gramática formal. Como a manutenção de regras da gramática formal não pode levar a ciclos, a regra pode ser mantida.

5.2.5. Transformação das árvores

Só são extraídas regras de nós não problemáticos, ou seja, nós que não se enquadrem em nenhum dos casos descritos na seção anterior. Quando um nó que não atende essa condição é encontrado, uma transformação é realizada para garantir que seja possível extrair alguma regra livre de contexto recobrindo a subárvore dominada por ele.

A operação realizada é uma fusão do nó com um antecedente na árvore. Seja *Y* o nó problemático e *X* o antecedente que o aceita. Todos os filhos de *Y* são movidos para *X*, preservando a ordem de precedência. Os nós filhos de *X* que precediam *Y* continuam precedendo todos os filhos de *Y*, e os que eram precedidos por *Y* são precedidos por todos os seus filhos. Tomem-se como exemplos as regras:

$$X \rightarrow \alpha Y \beta$$

$$Y \rightarrow \gamma \mid \chi .$$

A fusão dessas regras levaria a duas novas regras:

$$X \rightarrow \alpha\gamma\beta \mid \alpha\chi\beta$$

Note-se, porém, que, como as regras são extraídas de exemplos, não se pode garantir a extração das duas possibilidades, que ocorrerá somente se houver dois exemplos que independentemente permitam as duas operações. Em um corpus pequeno,

é provável, desse modo, que só uma das regras seja extraída, levando a uma perda de poder de generalização da gramática extraída em relação à gramática original.

5.2.6. Reordenamento dos nós das árvores-alvo

Com os alinhamentos restantes, a árvore-alvo correspondente a cada árvore-fonte pode ser construída através de uma transformação simples.

Os nós são percorridos em pós-ordem. Todos os nós terminais são removidos. Cada nó não-terminal recebe então como filhos novos nós rotulados com seus alinhamentos diretos. Se o fechamento do escopo do nó inclui palavras não-alinhadas, também são inseridos nós rotulados com essas palavras.

Esse tratamento das palavras não-alinhadas difere do proposto em WONG e MOONEY (2006); os autores preferem rotular os nós correspondentes com um símbolo-coringa que recobre qualquer palavra. No entanto, essa abordagem nos parece flexível demais, e claramente pode levar ao reconhecimento de sentenças não gramaticais. Preferimos extrair somente regras apoiadas nos exemplos do corpus. Generalizações, quando cabíveis, podem ser capturadas com o uso de filtros.

Inseridos todos os nós terminais, os nós da árvore-alvo transformada são ordenados de acordo com a precedência sintática em relação à sentença-alvo.

5.2.7. Extração de regras sincrônicas

A extração de regras acontece em dois momentos. Primeiramente, são extraídas as novas regras da gramática-alvo G'_{MRL}; com as operações de transformação realizadas anteriormente, essas regras são diferentes regras contidas na gramática dada, G_{MRL}. Por isso, é preciso percorrer as árvores transformadas e extrair as regras da gramática-alvo.

Da mesma forma, as árvores-fonte com nós terminais transformados e não-terminais reordenados a partir das árvores-alvo são percorridas para extração de regras. As regras são construídas inserindo-se cada nó não-terminal no LHS de uma regra, e

seus filhos, em ordem de precedência, no *RHS*. Note-se que, como existe uma bijeção entre os nós não-terminais das duas árvores, e uma bijeção entre os nós não-terminais de cada árvore e as regras extraídas em cada árvore, existe também uma bijeção entre as regras extraídas em cada árvore, permitindo que sejam unidas em regras sincrônicas. Para cada nó da árvore-fonte, portanto, é extraída uma regra sincrônica.

5.2.8. Obtenção das derivações sincrônicas

Para obter as derivações sincrônicas permitidas pela gramática extraída, e as contagens de regras necessárias para ponderar a gramática, utilizou-se um analisador de Earley adaptado para a gramática sincrônica. O analisador computa todas as possíveis *n* derivações

$$S \underset{GS}{\Rightarrow}^{*} < \alpha, \beta_i > \text{ para } 0 \geq i \geq n .$$

Têm-se assim todas as possíveis traduções β_i de acordo com a gramática.

5.2.9. Ponderação das regras da gramática sincrônica

Após o processo de extração das regras de todos os exemplos, a gramática-fonte da gramática sincrônica obtida será, possivelmente, uma gramática ambígua. Isso ocorre porque as mesmas palavras recebem diferentes alinhamentos em exemplos distintos, de modo que podem surgir regras alternativas para o mesmo insumo.

Para que seja possível escolher a melhor derivação entre as possibilidades compatíveis com a gramática, as regras são ponderadas para que haja um critério de decisão entre as alternativas. A ponderação é dada por:

$$P(X \rightarrow < \alpha_k, \beta_k >) = \frac{countCorrect(X \rightarrow < \alpha_k, \beta_k >)}{\sum_i countCorrect(X \rightarrow < \alpha_i, \beta_i >)}$$

onde $countCorrect(X \rightarrow < \alpha_k, \beta_k >)$ de uma regra é uma função que retorna a frequência com que a regra aparece em derivações corretas, ou seja, derivações

$S \underset{GS}{\Rightarrow}^{*} < \alpha, \beta_i >$ tais que $< \alpha, \beta_i > \in \tau$, no treinamento. As regras não utilizadas em nenhuma derivação correta receberam probabilidade 0,001, determinada empiricamente de maneira a garantir que essas regras tivessem probabilidade menor do que as regras observadas nas derivações corretas

5.3. Implementação

Os algoritmos foram implementados na linguagem Java. Os alinhamentos foram obtidos com o Berkeley Aligner (LIANG *et al.*, 2006; DENERO e KLEIN, 2007). O parser para G_{MRL} foi gerado com a ferramenta Antlr (PARR, 2007).

5.4. Resultados

Os resultados obtidos com o método proposto foram avaliados por meio de vários experimentos, realizados com o corpus CLANG, de 300 pares de sentenças manualmente traduzidos (ZELLE e MOONEY, 1996). Resumimos abaixo os dados obtidos.

Foram testados diferentes filtros e alinhamentos. As configurações testadas foram as seguintes:

(1) Experimento 1: usaram-se filtros para remover sinais de pontuação nos dois *corpora*, e o alinhamento foi feito entre palavras e palavras;

(2) Experimento 2: usaram-se filtros para sinais de pontuação nos dois *corpora*, filtros para remover determinantes <*the*> e <*a*> no corpus de linguagem natural, filtros para utilizar um rótulo em todos os números, e outro em todos os nomes de variáveis; e o alinhamento foi feito entre palavras e palavras;

(3) Experimento 3: usaram-se filtros para sinais de pontuação nos dois *corpora*, e o alinhamento foi feito entre palavras e regras;

(4) Experimento 4: usaram-se filtros para sinais de pontuação nos dois *corpora*, filtros para remover determinantes <*the*> e <*a*> no corpus de linguagem natural, filtros para utilizar um rótulo em todos os números, e outro em todos os nomes de variáveis; e o alinhamento foi feito entre palavras e nós;

(5) Experimento 5: usaram-se filtros para sinais de pontuação nos dois *corpora*, filtros para remover determinantes <*the*> e <*a*> no corpus de linguagem natural, filtros para utilizar um rótulo em todos os números, e outro em todos os nomes de variáveis; e o alinhamento foi feito entre palavras e regras.

Todos os experimentos foram feitos com alinhamentos treinados com cinco iterações cada do Modelo 1 e do Modelo 2 da IBM, na implementação do Berkeley Aligner. Considerou-se um acerto a produção da tradução correta de NL para MRL pela derivação mais provável encontrada com a gramática sincrônica extraída e ponderada no treinamento. A avaliação foi feita com divisão do *corpus* em k partes, sendo uma usada para teste e as outras para treinamento. Reportamos resultados para $k = 5$ e $k = 10$.

Tabela 5. Resultados.

	k = 5	***k = 10***
Experimento 1	4,3%	4,0%
Experimento 2	21,0%	22,3%
Experimento 3	4,6%	4,3%
Experimento 4	22,3%	24,0%
Experimento 5	19,7%	20,3%

Os resultados encontrados permitem observar, principalmente, a grande contribuição dos filtros para a gramática obtida a partir do alinhamento. Nota-se também que os melhores resultados foram obtidos alinhando-se palavras a nós, e não a regras, como proposto em WONG (2006). No entanto, os resultados não são, com esses

primeiros passos, comparáveis aos dos sistemas abordados. Propomos a seguir duas formas, não testadas, de refinar os resultados obtidos.

5.5. Refinamentos da solução

5.5.1. Transformações da gramática-alvo

Cada regra não-trivial (no sentido de ter mais de uma expansão possível) da gramática tem certo poder de generalização. A fusão de regras imposta pelo surgimento de regras não livres-de-contexto faz com que parte do poder total da gramática se perca; a gramática-alvo extraída ao final do processo é sempre correta, mas não necessariamente completa.

É importante por isso que se capture o máximo número de padrões passíveis de encodificação em regras livres de contexto. Digamos que se encontrem entre as regras da gramática dada o par:

REGION → `left` REGION

REGION → `right` REGION.

Se a primeira regra for extraída em um contexto de achatamento, como por exeplo:

REGION → • `left` REGION,

não será possível extrair

REGION → • `right` REGION

embora a regra seja legítima. No entanto, se `left` fosse substituído por um não-terminal que então pudesse ser reescrito como `left` ou `right`, a generalização seria capturada mesmo no contexto do achatamento.

Para investigar o impacto desse tipo de modificação da gramática dada, efetuaram-se algumas alterações nesse estilo, utilizando-se um algoritmo simples. Para todos os conjuntos de regras com mesmo *LHS*, foram analisados os subconjuntos grupos

de regras com *RHS* do mesmo comprimento. As regras que tinham entre si apenas uma posição diferente no *RHS* tiveram essa posição substituída por um novo não-terminal, e acrescentaram-se à gramática regras reescrevendo esse não-terminal em cada possibilidade que preenchia aquela posição nas regras originais. Portanto:

$$\text{REGION} \rightarrow \text{X REGION}$$

$$\text{X} \rightarrow \texttt{left}$$

$$\text{X} \rightarrow \texttt{right}$$

O Experimento 2, com alinhamento entre palavras e palavras, foi reproduzido com a gramática dada assim alterada. Os resultados obtidos para $k = 10$ foram de 25,3% de acerto.

5.5.2. Remoção de alinhamentos

Conforme mencionado, o custo da transformação dos nós da árvore é alto, pois as regras resultantes são sempre menos gerais do que as regras iniciais. A transformação das gramáticas formais é uma tentativa de reduzir o custo das transformações necessárias; apresentamos nesta seção uma outra estratégia, que visa diminuir o número de transformações que precisam ser feitas.

Um nó problemático *X* pode ser tornado não problemático e preservado se os alinhamentos recobertos pelo fechamento de seu escopo forem modificados de forma que deixe de haver alinhamentos com nós que não são dominados por *X*.

Um alinhamento é descartado se sua probabilidade é menor do que determinado mínimo e a sua preservação traz um custo para a árvore. Como a remoção de um alinhamento nunca torna um nó problemático

O problema da remoção de alinhamentos pode, dessa forma, ser visto como um problema de otimização combinatória. Cada nó problemático não estável pode ser tornado não problemático pela remoção de um entre vários conjuntos de alinhamentos.

Atribuindo-se à remoção de um alinhamento um custo que é uma função de sua probabilidade, podemos afirmar que a solução ótima é o conjunto de alinhamentos cuja remoção permite a preservação de todos os nós problemáticos não estáveis com o menor custo possível.

Fazendo uma investigação preliminar do papel da remoção de alinhamentos no resultado final, reproduzimos o Experimento 4 removendo todos os alinhamentos que induzissem regras problemáticas. Esses alinhamentos foram escolhidos da forma mais simples: a árvore foi percorrida em pré-ordem e cada nó, analisado em termos de seu escopo e do fechamento deste. Se dentro do fechamento do escopo de um nó foram identificados alinhamentos com nós não dominados por ele, esses alinhamentos foram imediatamente removidos. Obteve-se assim 21,0% de acertos para $k = 10$.

5.5.3. Uso de lacunas

Em todos os experimentos apresentados até aqui, os nós não alinhados foram tratados conforme a descrição que se encontra na Seção 5.2.6: quando o fechamento do escopo de um nó inclui palavras não-alinhadas, são inseridos na regra extraída daquele nó os rótulos dessas palavras, na ordem em que estas aparecem. No entanto, não é esse o tratamento dado por WONG e MOONEY (2006) a palavras não alinhadas; em seu lugar, os autores inserem, em vez das palavras, nós especiais, que chamamos de nós-lacuna, que podem ser expandidos como qualquer palavra da linguagem, ou ainda como ε. Alteramos inicialmente esse tratamento por dar margem a muitas extrapolações dos exemplos. Porém, reconhecemos que a flexibilidade trazida pelo uso de lacunas pode ser chave para a robustez da gramática extraída. Como a remoção de alinhamentos apresenta ampla oportunidade para lidar com palavras não alinhadas, reproduzimos o Experimento 2 usando remoção de alinhamentos e nós-lacuna para tratamento destas. Confirmando a previsão de que o uso de lacunas tem um papel importante nos bons

resultados obtidos pelos autores citados, esse último experimento trouxe 43,3% de acertos, um grande aumento em relação aos anteriores.

Capítulo 6. Conclusão

Este trabalho propôs uma abordagem ao problema da análise semântica, formulado como um problema de tradução do significado de fragmentos de linguagem natural para uma linguagem formal não ambígua. Trata-se de um problema relevante em várias áreas da Computação, da construção de interfaces baseadas em linguagens naturais à tradução automática de linguagens naturais. Investigou-se um método para aprendizado de regras de tradução entre uma linguagem natural e uma linguagem formal livre de contexto, com avaliação de algumas variações.

Verificamos que uma representação semântica de um fragmento de linguagem natural pode ser obtida pelo uso de uma gramática sincrônica que recubra a correspondência entre as duas linguagens. Apresentou-se um procedimento para aprendizado automático de uma gramática dessa natureza, bem como para uma ponderação de suas regras de modo a permitir escolher a melhor derivação onde houver ambiguidade.

O método proposto recebe como insumo:

(1) um corpus de árvores de derivação em uma linguagem formal (doravante o conjunto MRS e a linguagem MRL);

(2) um corpus de sentenças em linguagem natural (NS, NL);

(3) uma gramática para a linguagem MRL (*G*(MRL));

(4) um alinhamento (Ω), que pode ser

   a. entre palavras;

   b. entre palavras e nós terminais da gramática de MRL;

   c. entre palavras e regras da gramática de MRL.

O resultado do procedimento é uma gramática sincrônica ponderada (GS).

Partindo-se de uma formalização do problema da tradução de linguagens naturais em linguagens formais com gramáticas sincrônicas, analisou-se em detalhe o algoritmo de WONG e MOONEY (2006), e identificaram-se neste passos a desdobrar e melhorar. Foram introduzidos filtros, alinhamentos entre palavras e nós e transformações da gramática dada. Além disso, investigou-se com mais minúcias o papel de cada etapa do algoritmo nos resultados finais.

Mostrou-se que o uso de filtros para o alinhamento e extração das regras é uma maneira simples de obter grande incremento da performance. Duas propostas de para refinamento do algoritmo utilizado foram formuladas com base em um estudo minucioso de seu funcionamento. Revelou-se o papel importante do uso de lacuna e da remoção de alinhamento para a boa performance do algoritmo, implicando que a extração de regras rigorosamente fiéis aos exemplos pode ser restritiva demais para obter uma gramática robusta.

Como encaminhamentos deste trabalho, propomos a exploração de novos corpora, e possivelmente a ampliação do presente. Um número maior de exemplos pode vir a compensar a alta especificidade das regras extraídas. É de interesse também explorar melhor a extensão ao formalismo usado apresentada em WONG e MOONEY (2007), que pode levar a possibilidades de tradução para linguagens mais gerais de representação. Além disso, dois problemas importantes, levantados mas pouco explorados aqui, por se encontrarem além do escopo deste trabalho, se referem à otimização da remoção de alinhamentos, que aliada ao uso de lacunas parece ser um ponto-chave para a boa performance do algoritmo, e a definição de um bom modelo probabilístico para escolha entre derivações alternativas.

Como comentário final, observamos que, embora os resultados preliminares obtidos não sejam ainda compatíveis com os resultados de sistemas apresentados, os

experimentos e a análise dos resultados apontaram uma direção clara para aprimoramento do método.

Anexo A

Apresentamos abaixo o corpus utilizado nos experimentos realizados. Na coluna da esquerda se encontram as sentenças na linguagem formal CLANG, e na da direita, suas respectivas traduções.

1. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	1. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
2. ((bpos (half opp)) (do (player our {NUMBER}) (home (circle (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)) NUMBER))))	2. If ball is in opponent 's half then default position of player NUMBER should be within NUMBER meters of -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
3. ((true) (do (player our {NUMBER}) (home (pt NUMBER NUMBER))))	3. default position of player NUMBER should always be -LRB- NUMBER NUMBER -RRB-
4. ((and (playm play_on) (bpos (left-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))	4. If ball is close to far left corner when play mode is IDENT then player NUMBER should pass ball to opponent 's penalty area
5. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (left-quarter (midfield opp)))))	5. If ball is in -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- then all players except goalie should pass ball to far left side of opponent 's midfield
6. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass IDENT)))	6. When ball is in IDENT players NUMBER to NUMBER should pass ball to IDENT
7. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (clear (pt NUMBER NUMBER))))	7. If player NUMBER has ball then it should clear ball from -LRB- NUMBER NUMBER -RRB-
8. (definec IDENT (not (bowner (player our {NUMBER NUMBER}))))	8. Call condition where players NUMBER and NUMBER are not ball owner to be IDENT
9. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	9. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
10. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	10. If ball is in -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- then all players except goalie should pass ball to -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
11. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	11. Player NUMBER should not intercept ball if ball is within NUMBER meters of our goal line and not in our left penalty area
12. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (home (front-of-goal our))))	12. Before kick off default position of goalie should be directly in front of our goal
13. (definec IDENT (true))	13. Let IDENT be true
14. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER	14. If it is our goalie catch then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER

NUMBER)))))	NUMBER -RRB-
15. ((bpos IDENT) (do (player-except our {NUMBER}) (pass IDENT)))	15. If ball is in IDENT then all players except goalie should pass ball to IDENT
16. ((bpos (half our)) (do (player our {NUMBER}) (pos (front-of-goal our))))	16. If ball is in our half then position player NUMBER directly in front of our goal
17. ((and (bowner (player our {NUMBER})) (bpos (goal-area opp))) (do (player our {NUMBER}) (shoot)))	17. If ball is in opponent 's goal area then one who owns ball should shoot it
18. ((true) (do (player our {NUMBER}) (home (front-of-goal our))))	18. default position of goalie should always be in front of our goal
19. ((and (bowner (player our {NUMBER})) (bpos (right (penalty-area opp)))) (do (player our {NUMBER}) (pass (pt NUMBER NUMBER))))	19. If our team have ball and ball is in right side of opponent 's penalty area then our team should pass ball to -LRB- NUMBER NUMBER -RRB-
20. ((and (bowner (player our {NUMBER})) (bpos (right (near-goal-line our)))) (do (player our {NUMBER}) (pass (player our {NUMBER})))))	20. When goalie has ball on right side near our goal line it should pass ball to player NUMBER
21. ((not (playm bko)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	21. If it is not before kick off then player NUMBER should be positioned at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
22. ((bpos (midfield opp)) (do (player our {NUMBER}) (dribble (penalty-area opp))))	22. When ball is in opponent 's midfield player NUMBER should dribble ball to opponent 's penalty area
23. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER NUMBER NUMBER}) (dribble (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	23. If ball is in IDENT during normal play then players NUMBER NUMBER and NUMBER should dribble ball to -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
24. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	24. If ball is within NUMBER meters from our goal line but not in our left penalty area then player NUMBER should not intercept ball
25. (definec IDENT (true))	25. Let IDENT be true
26. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	26. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
27. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (from-goal our NUMBER))))	27. Whenever ball is NUMBER to NUMBER meters from our goal line player NUMBER should position itself NUMBER meters from our goal
28. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))	28. Let IDENT be condition where ball is within NUMBER meters of player NUMBER
29. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	29. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
30. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	30. If ball is within NUMBER meters from our goal line then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
31. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	31. When ball is in -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- all players except goalie should pass ball to -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
32. ((and (bpos (from-goal-line our	32. If ball is within NUMBER meters from

NUMBER NUMBER)) (not (bpos (right (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	our goal line but not in our right penalty area then player NUMBER should not intercept ball
33. ((and (playm fk_our) (bpos (penalty-area our))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	33. If ball is in our penalty area when it is our free kick then player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB-
34. ((bpos (right (penalty-area opp))) (do (player-except our {NUMBER}) (pass (left (penalty-area opp)))))	34. If ball is in opponent 's left penalty area then all players except goalie should pass ball to opponent 's right penalty area
35. ((and (playm fk_our) (bpos (penalty-area our))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	35. If ball is in our penalty area during our free kick then player NUMBER should be positioned at -LRB- NUMBER NUMBER -RRB-
36. ((and (bowner (player our {NUMBER})) (bpos (left (penalty-area opp)))) (do (player our {NUMBER}) (pass (pt NUMBER NUMBER))))	36. If our team have ball in opponent 's right penalty area then they should pass ball to -LRB- NUMBER NUMBER -RRB-
37. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	37. If ball is in our half then player NUMBER should be positioned at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
38. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	38. Position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB- if ball is in opponent 's half
39. ((bpos (left (penalty-area opp))) (do (player-except our {NUMBER}) (pass (right (penalty-area opp)))))	39. All players except goalie should pass ball to opponent 's left penalty area if ball is in right side of it
40. ((and (playm play_on) (bpos (left-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))	40. If ball is close to far left corner when play mode is IDENT then player NUMBER should pass ball to opponent 's penalty area
41. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	41. If ball is in opponent 's half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
42. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	42. Position player NUMBER at -LRB- NUMBER NUMBER -RRB- if ball is within NUMBER meters from our goal line
43. ((bpos (left-quarter (midfield our))) (do (player-except our {NUMBER}) (pass IDENT)))	43. All players except for goalie should pass to IDENT when ball is in far left side of our midfield
44. ((and (bowner (player our {NUMBER})) (bpos (left (near-goal-line our)))) (do (player our {NUMBER}) (pass (player our {NUMBER}))))	44. If goalie has ball on left side near our goal line then it should pass ball to player NUMBER
45. ((bpos (right (near-goal-line opp))) (do (player our {NUMBER}) (pos (from-goal opp NUMBER))))	45. Player NUMBER should position itself NUMBER meters from opponent 's goal if ball is on right side near opponent 's goal line
46. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	46. Position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB- when ball is in opponent 's half
47. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))	47. Define IDENT to be condition where ball is within NUMBER meters of player NUMBER
48. ((bpos (circle (pt (player our {NUMBER})) NUMBER)) (do (player our {NUMBER}) (pos (circle (pt ball) NUMBER))))	48. Player NUMBER should position itself within NUMBER meters of ball when ball is within NUMBER meters of player NUMBER
49. ((true) (do (player our {NUMBER})	49. default position of player NUMBER

(home (pt NUMBER NUMBER))))	should always be -LRB- NUMBER NUMBER -RRB-
50. ((and (bowner (player our {NUMBER})) (bpos (midfield))) (do (player our {NUMBER}) (shoot)))	50. If our team have ball in midfield then they should shoot it
51. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	51. Player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB- before kick off
52. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	52. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
53. ((bpos (left (penalty-area opp))) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))	53. If ball is in left side of opponent 's penalty area then players NUMBER to NUMBER should pass it to right of opponent 's penalty area
54. ((not (or IDENT IDENT IDENT IDENT IDENT IDENT)) (do (player our {NUMBER NUMBER NUMBER}) (shoot)))	54. If none of conditions IDENT IDENT IDENT IDENT IDENT or IDENT is true then our players NUMBER NUMBER and NUMBER should shoot
55. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass (left-quarter (midfield opp)))))	55. If ball is in IDENT then our players NUMBER to NUMBER should pass it to left quarter side of opponent 's midfield
56. ((and (playm play_on) (bpos (left-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))	56. During normal play if ball is in left quarter side near opponent 's goal line then player NUMBER should pass ball to opponent 's penalty area
57. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	57. If ball is within NUMBER meters from our goal line but not on left side of our penalty area then player NUMBER should not intercept
58. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (pass (left (penalty-area opp)))))	58. During normal play when ball is in region IDENT then player NUMBER should pass it to left side of opponent 's penalty area
59. ((bpos (right-quarter (near-goal-line opp))) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))	59. If ball is in right quarter of area near opponent 's goal line then players NUMBER NUMBER should pass it to right side of opponent 's penalty area
60. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	60. If ball is at distance NUMBER to NUMBER from our goal line then position player NUMBER to point -LRB- NUMBER NUMBER -RRB-
61. ((bpos (right (half our))) (do (player our {NUMBER NUMBER NUMBER}) (pass (pt NUMBER NUMBER))))	61. If ball is in our right half then players NUMBER NUMBER and NUMBER should pass ball to point -LRB- NUMBER NUMBER -RRB-
62. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	62. If ball is within NUMBER meters from our goal line then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
63. ((bpos IDENT) (do (player our {NUMBER}) (home (circle (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)) NUMBER))))	63. If ball is in IDENT then player NUMBER 's default position should be in circle of radius NUMBER from -LRB- NUMBER NUMBER -RRB- and its ball attraction should be -LRB- NUMBER NUMBER -RRB-
64. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (pass (player our {NUMBER NUMBER}))))	64. If player NUMBER has ball then it should pass to players NUMBER or NUMBER
65. ((bpos (half our)) (do (player our	65. If ball is in our half then position player

{NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
66. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (from-goal opp NUMBER))))

67. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (pass (left (penalty-area opp)))))
68. ((and (bowner (player our {NUMBER})) (bpos IDENT)) (do (player our {NUMBER}) (clear IDENT)))
69. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

70. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (from-goal our NUMBER))))

71. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass (left (penalty-area opp)))))
72. ((and (bowner (player our {NUMBER})) (bpos (penalty-area opp))) (do (player our {NUMBER}) (shoot)))
73. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
74. ((not (bpos (goal-area our))) (dont (player our {NUMBER NUMBER}) (intercept)))

75. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))

76. ((true) (do (player our {NUMBER}) (home (pt NUMBER NUMBER))))
77. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

78. ((or (playm gc_our) (playm gk_our) (playm fk_our)) (do (player our {NUMBER}) (clear (from-goal-line our NUMBER NUMBER))))
79. (definec IDENT (true))
80. (definec IDENT (and (bowner (player our {NUMBER})) (bpos IDENT)))
81. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (from-goal our NUMBER) (pt NUMBER NUMBER)))))
82. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
83. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos

NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
66. If ball is at distance NUMBER to NUMBER from opponent 's goal line then position player NUMBER at NUMBER meters from opponent 's goal
67. During normal play if ball is in IDENT then player NUMBER should pass it to left side of opponent 's penalty area
68. If ball is in IDENT then whoever owns ball should clear it from IDENT

69. If ball is at distance NUMBER to NUMBER from our goal line then position player NUMBER at point -LRB- NUMBER NUMBER -RRB-
70. If ball is between NUMBER to NUMBER meters from our goal line then position our player NUMBER at NUMBER meters from our goal
71. If ball is in region IDENT then players NUMBER NUMBER should pass it to left side of opponent 's penalty area
72. If ball is in opponent 's penalty area then whoever has ball should shoot

73. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
74. If ball is not in our goal area then players NUMBER and NUMBER should not intercept it
75. Define IDENT to be condition that ball is within distance NUMBER from our player NUMBER
76. default position of player NUMBER should be -LRB- NUMBER NUMBER -RRB-
77. If ball is at distance NUMBER to NUMBER from opponent 's goal line then our player NUMBER should be positioned at -LRB- NUMBER NUMBER -RRB-
78. During our goalie catch goal kick or free kick player NUMBER should clear ball up to NUMBER meters from our goal line
79. Call IDENT as true condition
80. Let IDENT be condition that player NUMBER has ball and ball is in IDENT
81. If ball is in our half then player NUMBER should be positioned at NUMBER meters from our goal with its ball attraction as -LRB- NUMBER NUMBER -RRB-
82. If ball is in opponent 's half then position player NUMBER at point -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
83. Whenever ball is at distance NUMBER to NUMBER from opponent 's goal line then player NUMBER should be positioned at -

(pt NUMBER NUMBER))))

84. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

LRB- NUMBER NUMBER -RRB-

84. If ball is at distance NUMBER to NUMBER meters from our goal line then position player NUMBER at point -LRB- NUMBER NUMBER -RRB-

85. (definec IDENT (and (bpos (penalty-area our)) (ppos-any (player our {NUMBER}) (midfield our)) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))

85. Let IDENT be condition that ball is in our penalty area player NUMBER is in our midfield while no opponent player is within distance of NUMBER from player NUMBER

86. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (left (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))

86. Define IDENT to be condition that ball is in region IDENT player NUMBER is in left side of our midfield and no opponent is within NUMBER distance from player NUMBER

87. ((bpos (half opp)) (do (player our {NUMBER}) (home (circle (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)) NUMBER))))

87. If ball is in opponent 's half then player NUMBER 's default position should be within NUMBER meters from point -LRB- NUMBER NUMBER -RRB- and set its ball attraction to be -LRB- NUMBER NUMBER -RRB-

88. ((bpos (near-goal-line our)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

88. If ball is near our goal line then position player NUMBER at -LRB- NUMBER NUMBER -RRB-

89. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

89. If ball is at distance NUMBER to NUMBER from opponent 's goal line then position player NUMBER at point -LRB- NUMBER NUMBER -RRB-

90. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass IDENT)))

90. When ball is in IDENT then players NUMBER NUMBER should pass ball to IDENT

91. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (right (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))

91. Let IDENT be condition that ball is in region IDENT our player NUMBER is on right side of our midfield and no opponent is around our player NUMBER within NUMBER distance

92. (definec IDENT (true))

92. Let IDENT be true condition

93. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass IDENT)))

93. If ball is present in region IDENT then players NUMBER NUMBER should pass it to region IDENT

94. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))

94. If ball is within NUMBER meters from our goal line and not on left side of our penalty area then player NUMBER should not intercept ball

95. ((bpos (goal-area our)) (do (player our {NUMBER}) (intercept)))

95. If ball is in our goal area then player NUMBER should intercept it

96. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

96. If ball is at distance NUMBER to NUMBER from opponent 's goal line then player NUMBER should be positioned at point -LRB- NUMBER NUMBER -RRB-

97. ((bpos (goal-area our)) (do (player our {NUMBER}) (intercept)))

97. If ball is in our goal area then player NUMBER should intercept it

98. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))

98. If ball is within NUMBER meters from our goal line but not on left side of our penalty area then player NUMBER should not intercept

99. ((bpos (circle (pt (player our {NUMBER})) NUMBER)) (do (player our {NUMBER}) (pos (circle (pt ball) NUMBER))))

99. If ball is within NUMBER meters from our player NUMBER then position player NUMBER somewhere within NUMBER meters of ball

100. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (pos (from-goal our NUMBER))))

100. Before kickoff or after our or opponent 's goal position player NUMBER at distance NUMBER from our goal

101. ((and (bowner (player our

101. If our player NUMBER has ball and

{NUMBER})) (bpos (right (field)))) (do (player our {NUMBER}) (clear (pt NUMBER NUMBER))))	ball is on right side of field then player NUMBER should clear it from point -LRB- NUMBER NUMBER -RRB-
102. ((and (bpos (half opp)) (playm play_on)) (do (player our {NUMBER}) (pos (circle (pt NUMBER NUMBER) NUMBER))))	102. During normal play if ball is in opponent 's half then player NUMBER should be positioned in circle with center -LRB- NUMBER NUMBER -RRB- and of radius NUMBER
103. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (right (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	103. Let IDENT be condition that ball is in region IDENT player NUMBER is on right side of our midfield and no opponent is within distance NUMBER from player NUMBER
104. (IDENT (do (player our {NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))	104. If condition IDENT holds then players NUMBER NUMBER and NUMBER should pass to player NUMBER
105. ((and (playm fk_our) (bpos (penalty-area our))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	105. During our free kick if ball is in our penalty area then position player NUMBER at -LRB- NUMBER NUMBER -RRB-
106. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	106. If ball is in opponent 's half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with its ball attraction set to -LRB- NUMBER NUMBER -RRB-
107. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (pass (penalty-area opp))))	107. When play mode is normal and ball is in region IDENT then our player NUMBER should pass ball to opponent 's penalty area
108. (IDENT (do (player our {NUMBER NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))	108. When condition IDENT is true then players NUMBER NUMBER NUMBER and NUMBER should pass to NUMBER
109. (IDENT (do (player our {NUMBER NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))	109. If IDENT condition is true then players NUMBER NUMBER NUMBER and NUMBER should pass to player NUMBER
110. (definec IDENT (bpos (near-goal-line our)))	110. Call IDENT condition that ball is near our goal line
111. ((and (playm fk_our) (bpos (penalty-area our))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	111. During our free kick if ball is in our penalty area then position our player NUMBER at -LRB- NUMBER NUMBER -RRB-
112. ((true) (do (player our {NUMBER}) (home (pt NUMBER NUMBER))))	112. default position of player NUMBER should be at -LRB- NUMBER NUMBER -RRB-
113. (definec IDENT (and (bowner (player our {NUMBER})) (bpos (reg-exclude (half our) (near-goal-line our)))))	113. Let IDENT be condition that our player NUMBER has ball and ball is in our half excluding area near our goal line
114. ((bpos (right-quarter (near-goal-line opp))) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))	114. If ball is on right quarter portion of area near opponent 's goal line then players NUMBER NUMBER should pass to right side of opponent 's penalty area
115. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (right (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	115. If ball is within NUMBER meters from our goal line and not on right side of our penalty area then player NUMBER should not intercept it
116. ((and (bowner (player our {NUMBER})) (bpos (right (penalty-area opp)))) (do (player our {NUMBER}) (pass (pt NUMBER NUMBER))))	116. If our player has ball and ball is on right side of opponent 's penalty area then he should pass it to point -LRB- NUMBER NUMBER -RRB-
117. (IDENT (do (player our {NUMBER NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))	117. If condition IDENT is true then players NUMBER NUMBER NUMBER and NUMBER should give pass to player NUMBER
118. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	118. During our goalie catch position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-

119. ((and (bowner (player our {NUMBER})) (bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))) (do (player our {NUMBER}) (pass (pt NUMBER NUMBER))))	119. If ball is in rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- and we have ball then it should be passed to -LRB- NUMBER NUMBER -RRB-
120. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (from-goal our NUMBER))))	120. When ball is at distance NUMBER to NUMBER from our goal line then position player NUMBER at NUMBER meters from our goal
121. ((bpos (field)) (do (player our {NUMBER}) (home (pt NUMBER NUMBER))))	121. When ball is anywhere in field then player NUMBER 's default position should be -LRB- NUMBER NUMBER -RRB-
122. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER NUMBER NUMBER}) (dribble (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	122. During normal play if ball is in region IDENT then player NUMBER NUMBER and NUMBER should dribble ball to rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
123. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (left (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	123. Call IDENT to be condition that ball is in region IDENT player NUMBER is in our left midfield and no oppnent is within distance NUMBER from our player NUMBER
124. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	124. If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-
125. ((true) (do (player our {NUMBER}) (home (pt NUMBER NUMBER))))	125. Player NUMBER 's default position should always be at -LRB- NUMBER NUMBER -RRB-
126. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	126. If ball is within NUMBER meters from our goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
127. ((true) (do (player our {NUMBER}) (home (circle (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)) NUMBER))))	127. Set player NUMBER 's default position within NUMBER meters from -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
128. ((bpos (right-quarter (from-goal-line opp NUMBER NUMBER))) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))	128. If ball is in far right side NUMBER to NUMBER meters from opponent 's goal line players NUMBER to NUMBER should pass ball to opponent 's right penalty area
129. ((bpos (midfield our)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	129. If ball is in our midfield position player NUMBER at -LRB- NUMBER NUMBER -RRB-
130. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (from-goal opp NUMBER) (pt NUMBER NUMBER)))))	130. If ball is in opponent 's half position player NUMBER at NUMBER meters from opponent 's goal with ball attraction -LRB- NUMBER NUMBER -RRB-
131. (definec IDENT (and (bpos (penalty-area our)) (ppos-any (player our {NUMBER}) (midfield our)) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	131. Let IDENT be condition where ball is in our penalty area player NUMBER is in our midfield and no opponents are within NUMBER meters from him
132. ((bpos (left-quarter (near-goal-line opp))) (do (player-except our {NUMBER}) (pass (left (penalty-area opp)))))	132. If ball is in far left corner our players except goalie should pass ball to opponent 's left penalty area
133. ((bpos (left-quarter (midfield opp))) (do (player-except our {NUMBER}) (pass (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	133. If ball is in far left side of opponent 's midfield all our players except goalie should pass ball to rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
134. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt	134. If ball is in opponent 's half position our player NUMBER at -LRB- NUMBER

NUMBER NUMBER) (pt NUMBER NUMBER)))))	NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
135. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	135. If ball is in our half position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
136. ((bpos (left-quarter (from-goal-line our NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (left-quarter (midfield our)))))	136. If ball is in far left side NUMBER to NUMBER meters from our goal line our players except goalie should pass ball to far left side of our midfield
137. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	137. If it is our goalie catch position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
138. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (right (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	138. If ball is within NUMBER meters from our goal line and it is not in our right penalty area player NUMBER should not intercept it
139. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	139. If ball is within NUMBER meters from our goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
140. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	140. If it 's our goalie catch position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
141. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (pass (player our {NUMBER})))))	141. If player NUMBER has ball it should pass ball to player NUMBER
142. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (right (midfield our)))))	142. If ball is in rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- our players except goalie should pass ball to our right midfield
143. (definec IDENT (true))	143. Call condition true to be IDENT
144. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	144. If ball is NUMBER to NUMBER meters from opponent 's goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
145. ((and (playm play_on) (bpos (right-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))	145. During play on if ball is in far right corner player NUMBER should pass ball to opponent 's penalty area
146. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	146. If ball is NUMBER to NUMBER meters from opponent 's goal line position our player NUMBER at -LRB- NUMBER NUMBER -RRB-
147. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (shoot)))	147. If our player NUMBER has ball it should shoot
148. ((and (bpos (from-goal-line our NUMBER NUMBER)) (not (bpos (left (penalty-area our))))) (dont (player our {NUMBER}) (intercept)))	148. If ball is within NUMBER meters from our goal line and ball is not in our left penalty area player NUMBER should not intercept ball
149. ((and (bpos (near-goal-line our)) (playm play_on)) (do (player our {NUMBER}) (pos (circle (pt NUMBER NUMBER) NUMBER))))	149. If ball is very close to our goal line during normal play position player NUMBER within NUMBER meter from -LRB- NUMBER NUMBER -RRB-
150. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	150. If ball is in our half position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
151. ((and (playm play_on) (bpos (midfield))) (do (player our {NUMBER NUMBER NUMBER}) (dribble ((pt ball) + (pt	151. During normal play if ball is in midfield player NUMBER NUMBER or NUMBER should dribble to point NUMBER meters in

NUMBER NUMBER)))))
152. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
153. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (pass (player our {NUMBER NUMBER}))))
154. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
155. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (left (penalty-area opp)))))

156. ((true) (do (player our {NUMBER}) (home (front-of-goal our))))
157. ((bpos (right-quarter (from-goal-line our NUMBER NUMBER))) (do (player-except our {NUMBER}) (pass (right-quarter (midfield our)))))
158. (definec IDENT (and (bpos (penalty-area our)) (ppos-any (player our {NUMBER}) (midfield our)) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))
159. ((bpos (right-quarter (near-goal-line opp))) (do (player-except our {NUMBER}) (pass (right (penalty-area opp)))))
160. ((bpos (near-goal-line our)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
161. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
162. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
163. (definec IDENT (or (playm ki_our) (playm ck_our) (playm fk_our) (playm gk_our) (playm ko_our)))
164. ((bpos (circle (pt (player our {NUMBER})) NUMBER)) (do (player our {NUMBER}) (pos (circle (pt ball) NUMBER))))
165. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
166. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))
167. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
168. ((true) (do (player our {NUMBER NUMBER NUMBER}) (shoot)))
169. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

front of ball
152. If it is before kick off after our goal or after opponent 's goal position player NUMBER at -LRB- NUMBER NUMBER -RRB-
153. If player NUMBER has ball it should pass ball to player NUMBER or NUMBER

154. If ball is NUMBER to NUMBER meters from opponent 's goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
155. If ball is in rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- our players except goalie should pass ball to left side of opponent 's penalty area
156. Set goalie 's default position to be in front of our goal
157. If ball is in far right side NUMBER to NUMBER meters from our goal line our players except goalie should pass ball to far right side of our midfield
158. Let IDENT be condition where ball is in our penalty area player NUMBER is in our midfield and no opponents are within NUMBER meters from him
159. If ball is in far right corner our players except goalie should pass ball to right side of opponent 's penalty area
160. If ball is near our goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
161. If ball is NUMBER to NUMBER meters from opponent 's goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
162. If ball is NUMBER to NUMBER meters from opponent 's goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
163. Let IDENT be condition where play mode is our kick in corner kick free kick goalie 's kick or kick off
164. If ball is within NUMBER meters of our player NUMBER position player NUMBER within NUMBER meters of ball
165. If ball is in opponent 's half position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
166. If ball is NUMBER to NUMBER meters from opponent 's goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB-
167. If ball is within NUMBER meters from our goal line position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
168. Players NUMBER NUMBER or NUMBER should shoot
169. If ball is in rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- position player NUMBER at -LRB- NUMBER NUMBER -RRB-

170. ((and (bowner (player our {NUMBER})) (bpos (right (field)))) (do (player our {NUMBER}) (clear (pt NUMBER NUMBER))))	170. If goalie has ball in right field he should clear ball from -LRB- NUMBER NUMBER -RRB-
171. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (pos (from-goal our NUMBER))))	171. If it is before kick off after our goal or after opponent 's goal position player NUMBER at NUMBER meters from our goal
172. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	172. If ball is in opponent 's half position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
173. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	173. If ball is on our half of field player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB- with ball attraction of -LRB- NUMBER NUMBER -RRB-
174. ((bpos (circle (pt (player our {NUMBER})) NUMBER)) (do (player our {NUMBER}) (pos (circle (pt ball) NUMBER))))	174. Player NUMBER should intercept ball if it is close by
175. ((and (playm play_on) (bpos (right-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))	175. In play on mode if ball is in far right corner then player NUMBER should pass to opponent 's penalty area
176. ((and (playm fk_our) (bpos (penalty-area our))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	176. If it is our free kick and ball is in our penalty box player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB-
177. ((bpos (goal-area our)) (do (player our {NUMBER}) (intercept)))	177. If ball is in our goal box then goalie should intercept ball
178. ((bpos (near-goal-line our)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	178. If ball is in near quarter of field player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB-
179. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))	179. Let IDENT be condition where ball is within NUMBER meters of our player NUMBER
180. ((and (bowner (player our {NUMBER NUMBER NUMBER})) (bpos (right (half our)))) (do (player our {NUMBER NUMBER NUMBER}) (pass (pt NUMBER NUMBER))))	180. If our player NUMBER NUMBER or NUMBER has ball and ball is in near right quarter of field then players NUMBER NUMBER and NUMBER should clear ball to -LRB- NUMBER NUMBER -RRB-
181. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (pass (right (penalty-area opp)))))	181. If it is playon mode and ball is in IDENT player NUMBER should pass to right side of opponent 's penalty area
182. ((and (bpos (right (penalty-area our))) (bowner (player our {NUMBER}))) (do (player our {NUMBER}) (pass (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	182. If ball is in right side of our penalty box then ball owner should pass to rectangle -LRB- NUMBER NUMBER NUMBER NUMBER -RRB-
183. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (pass (penalty-area opp))))	183. If it is playon mode and ball is in IDENT player NUMBER should pass to opponent 's penalty area
184. (definec IDENT (and (bowner (player our {NUMBER NUMBER NUMBER NUMBER})) (bpos (near-goal-line our))))	184. Let IDENT be condition where our player NUMBER NUMBER NUMBER or NUMBER have ball and ball is close to our goal line
185. ((bpos (near-goal-line opp)) (do (player our {NUMBER NUMBER}) (pass (player our {NUMBER}))))	185. If ball is in far quarter of field players NUMBER and NUMBER should pass to player NUMBER
186. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (from-goal our NUMBER) (pt NUMBER NUMBER)))))	186. If playmode is our gc then player NUMBER should position itself NUMBER meters in front of our goal with ball attraction of -LRB- NUMBER NUMBER -RRB-
187. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos	187. If ball is within NUMBER meters to our goal line player NUMBER should position itself

(pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
188. (IDENT (do (player our {NUMBER NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))
189. (definec IDENT (true))
190. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))

191. (definec IDENT (and (bowner (player our {NUMBER})) (bpos (reg-exclude (half our) (near-goal-line our)))))
192. ((bpos (reg-exclude (field) (near-goal-line opp))) (dont (player-range our NUMBER NUMBER) (shoot)))
193. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (left (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))
194. ((bpos (midfield our)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

195. ((and (bowner (player our {NUMBER})) (bpos IDENT)) (do (player our {NUMBER}) (clear IDENT)))
196. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

197. (definec IDENT (and (bpos (penalty-area our)) (ppos-any (player our {NUMBER}) (midfield our)) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))
198. (definec IDENT (and (bowner (player our {NUMBER})) (bpos IDENT)))

199. ((not (or IDENT IDENT IDENT IDENT IDENT IDENT)) (do (player our {NUMBER NUMBER NUMBER}) (shoot)))
200. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass IDENT)))

201. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))

202. (definec IDENT (and (bowner (player our {NUMBER})) (bpos (reg-exclude (half our) (near-goal-line our)))))
203. ((bpos (half opp)) (do (player our {NUMBER}) (pos (front-of-goal our))))

204. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (dribble (penalty-area opp))))
205. ((and (playm play_on) (bpos (right-quarter (near-goal-line opp)))) (do (player our {NUMBER}) (pass (penalty-area opp))))
206. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos

at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
188. If IDENT is true then players NUMBER NUMBER NUMBER and NUMBER should pass to player NUMBER
189. Define IDENT to be true condition
190. Define IDENT NUMBER as condition that ball is within NUMBER meters from our player NUMBER
191. Let IDENT be condition where our player NUMBER has ball and it is in our half excluding area near our goal line
192. Unless ball is in far quarter of field players NUMBER NUMBER should not shoot

193. Call IDENT as condition that ball is in IDENT and our player NUMBER is on left side of our midfield and no opponents are within NUMBER m of our player NUMBER
194. If ball is in our midfield then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
195. If one of our players has ball and ball is in IDENT then he should clear ball to IDENT

196. If ball is in rectangle -LRB- NUMBER NUMBER NUMBER NUMBER -RRB- then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
197. Call IDENT to be condition that ball is in our penalty area and our player NUMBER is in our midfield and no opponents are within NUMBER m of our player NUMBER
198. Let IDENT NUMBER be condition that our player NUMBER has ball and it is in IDENT NUMBER
199. If IDENT IDENT IDENT IDENT IDENT and IDENT are all false then players NUMBER NUMBER and NUMBER should shoot
200. If ball is in IDENT then players NUMBER NUMBER should pass to someone in IDENT

201. If ball is at distance of NUMBER to NUMBER from opponent 's goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
202. Let IDENT be condition where our player NUMBER has ball and it is our half excluding area near our goal line
203. If ball is in opponent 's half then goalie should position itself directly in front of our goal

204. If ball is NUMBER m to NUMBER m from opponent 's goal line then player NUMBER should dribble to their penalty box
205. If it is in play on mode and ball is in far right corner then player NUMBER should pass ball to opponent 's penalty area
206. If ball is NUMBER to NUMBER meters from opponent 's goal line then player

(pt NUMBER NUMBER))))	NUMBER should go to -LRB- NUMBER NUMBER -RRB-
207. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (right (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	207. Let IDENT be condition that ball is in IDENT and our player NUMBER is in our right midfield and no opponents are within NUMBER m of him
208. (definec IDENT (bpos (circle (pt (player our {NUMBER})) NUMBER)))	208. Let IDENT NUMBER be condition where ball is within NUMBER meters of our player NUMBER
209. ((playm gc_our) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	209. If playmode is IDENT then player NUMBER should make its position -LRB- NUMBER NUMBER -RRB- with ball attraction of -LRB- NUMBER NUMBER -RRB-
210. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (right (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	210. Call IDENT condition that ball is in IDENT and our player NUMBER is on right side of our midfield and no opponents are within NUMBER m of our player NUMBER
211. (definec IDENT (and (bowner (player our {NUMBER})) (bpos (reg-exclude (half our) (near-goal-line our)))))	211. Let IDENT be condition where our player NUMBER has ball and it is our half but not near our goal line
212. ((bowner (player our {NUMBER NUMBER NUMBER})) (do (player our {NUMBER NUMBER NUMBER}) (shoot)))	212. If our player NUMBER NUMBER or NUMBER has ball then he should shoot it
213. ((bpos (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))) (do (player-range our NUMBER NUMBER) (pass (right-quarter (from-goal-line opp NUMBER NUMBER)))))	213. If ball is in rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- then players NUMBER to NUMBER should pass ball to far right side NUMBER to NUMBER meters from opponent 's goal line
214. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass IDENT)))	214. If ball is in IDENT then players NUMBER NUMBER should pass to someone in IDENT
215. ((bpos (half opp)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	215. If ball is in opponent 's half then player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB- with ball attraction of -LRB- NUMBER NUMBER -RRB-
216. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	216. If ball is NUMBER to NUMBER meters from opponent 's goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
217. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	217. If ball is in our half then player NUMBER should set its position to -LRB- NUMBER NUMBER -RRB- with ball attraction of -LRB- NUMBER NUMBER -RRB-
218. (definec IDENT (and (playm play_on) (bowner (player opp {NUMBER})) (bpos (half our))))	218. Let DefenseSituation be condition where play mode is play on opponents have ball and ball is in our half
219. ((and (bpos (right (penalty-area our))) (bowner (player our {NUMBER}))) (do (player our {NUMBER}) (pass (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	219. If ball is in our right penalty box and one of our players has ball then he should pass to teammate in rectangle -LRB- NUMBER NUMBER NUMBER NUMBER -RRB-
220. ((bpos (left-quarter (near-goal-line opp))) (do (player-except our {NUMBER}) (pass (left (penalty-area opp)))))	220. If ball is in left quarter of area near opponent 's goal line then all players except NUMBER should pass to left side of opponent 's penalty area
221. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	221. If ball is NUMBER to NUMBER meters from opponent 's goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-

222. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (from-goal our NUMBER) (pt NUMBER NUMBER)))))	222. If ball is in our half then player NUMBER should position itself at NUMBER meters from our goal with ball attraction of -LRB- NUMBER NUMBER -RRB-
223. ((bowner (player our {NUMBER})) (do (player our {NUMBER}) (shoot)))	223. If our player NUMBER has ball then he should take shot on goal
224. ((or (playm bko) (playm ag_our) (playm ag_opp)) (do (player our {NUMBER}) (home (front-of-goal our))))	224. If it is before kick off then goalie should make its home position directly in front of our goal
225. ((bpos (left-quarter (midfield))) (do (player our {NUMBER}) (intercept)))	225. If ball is in left quarter of midfield then player NUMBER should intercept it
226. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (left (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	226. Let IDENT be condition where ball is in IDENT and our player NUMBER is in our left midfield and no opponents are within NUMBER m of him
227. ((bpos IDENT) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))	227. If ball is in IDENT then players NUMBER NUMBER should pass to someone in opponent 's left penalty area
228. (IDENT (do (player our {NUMBER NUMBER NUMBER NUMBER}) (pass (player our {NUMBER}))))	228. If IDENT is true then players NUMBER NUMBER NUMBER and NUMBER should pass to player NUMBER
229. ((bpos (right (from-goal-line our NUMBER NUMBER))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	229. If ball is on right side within NUMBER m from our goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
230. (definec IDENT (and (bpos IDENT) (ppos-any (player our {NUMBER}) (right (midfield our))) (ppos-none opp (circle (pt (player our {NUMBER})) NUMBER))))	230. Call IDENT as condition that ball is in IDENT and our player NUMBER is in our right midfield and no opponents are within NUMBER m of our player NUMBER
231. (definec IDENT (and (bowner (player our {NUMBER})) (bpos (reg-exclude (half our) (near-goal-line our)))))	231. Let IDENT be condition where our player NUMBER owns ball when ball is in our half but not near our goal line
232. ((bpos (left-quarter (midfield our))) (do (player-except our {NUMBER}) (pass IDENT)))	232. If ball is in left quarter of our midfield then everyone but goalie should pass to IDENT
233. ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	233. If ball is NUMBER to NUMBER meters from opponent 's goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
234. ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	234. If ball is NUMBER to NUMBER meters from our goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
235. (definec IDENT (and (bowner (player our {NUMBER})) (bpos IDENT)))	235. Let IDENT NUMBER be true if our player NUMBER has ball and it is in IDENT NUMBER
236. ((and (playm play_on) (bpos IDENT)) (do (player our {NUMBER}) (dribble ((pt ball) + (pt NUMBER NUMBER)))))	236. If it is play on mode and ball is in IDENT then player NUMBER should dribble to point NUMBER m in front of ball
237. ((bpos (right (from-goal-line our NUMBER NUMBER))) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	237. If ball is on right side NUMBER to NUMBER meters from our goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
238. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	238. If ball is in our half then player NUMBER should position itself at -LRB- NUMBER NUMBER -RRB- with ball attraction of -LRB- NUMBER NUMBER -RRB-
239. ((bpos (half our)) (do (player our {NUMBER}) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))	239. If ball is our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- with ball attraction -LRB- NUMBER NUMBER -RRB-
240. ((bpos (from-goal-line our NUMBER	240. If ball is within NUMBER meters from

NUMBER)) (do (player our {NUMBER}) (pos (pt NUMBER NUMBER))))	our goal line then player NUMBER should go to -LRB- NUMBER NUMBER -RRB-
241. (definer IDENT (midfield))	241. Call midfield to be IDENT
242. (definer IDENT (left-quarter (midfield opp)))	242. Call left quarter of opponent 's midfield as IDENT
243. (definer IDENT (right (from-goal-line our NUMBER NUMBER)))	243. Let IDENT be right side of field NUMBER to NUMBER meters from our goal line
244. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	244. Let RegOppMid be rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
245. (definer IDENT (circle (pt NUMBER NUMBER) NUMBER))	245. Call circle of radius NUMBER meters from -LRB- NUMBER NUMBER -RRB- as IDENT
246. (definer IDENT (right (penalty-area opp)))	246. Call right side of opponent 's penalty area to be IDENT
247. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	247. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT
248. (definer IDENT (penalty-area opp))	248. Let opponent 's penalty area be called IDENT
249. (definer IDENT (right-quarter (near-goal-line opp)))	249. Call far right corner to be IDENT
250. (definer IDENT (left (penalty-area opp)))	250. Define IDENT to be left side of opponent 's penalty box
251. (definer IDENT (midfield our))	251. Define IDENT as our midfield
252. (definer IDENT (reg (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)) (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))))	252. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- or -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT NUMBER
253. (definer IDENT (penalty-area our))	253. Call our penalty area IDENT
254. (definer IDENT (front-of-goal our))	254. Let area in front of our goal be called IDENT
255. (definer IDENT (left (from-goal-line our NUMBER NUMBER)))	255. Let IDENT be left side of field NUMBER to NUMBER meters from our goal line
256. (definer IDENT (half opp))	256. Let OppHalf be opponent 's half
257. (definer IDENT (reg-exclude (reg-exclude (field) (near-goal-line our)) (near-goal-line opp)))	257. Call IDENT field excluding our and opponent 's goal lines
258. (definer IDENT (circle (pt (player our {NUMBER})) NUMBER))	258. Let IDENT be circle of NUMBER meters radius from our player NUMBER
259. (definer IDENT (circle (pt ball) NUMBER))	259. Let ArcofBall be area NUMBER m within ball
260. (definer IDENT (right-quarter (near-goal-line our)))	260. Define IDENT as right quarter of area near our goal line
261. (definer IDENT (reg (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)) (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))))	261. Call rectangles -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- or -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT NUMBER
262. (definer IDENT (from-goal-line opp NUMBER NUMBER))	262. Let IDENT be region NUMBER to NUMBER meters from opponent 's goal line
263. (definer IDENT (left-quarter (near-goal-line opp)))	263. Call IDENT as left quarter of area near opponent 's goal line
264. (definer IDENT (left (from-goal-line our NUMBER NUMBER)))	264. Let IDENT be left side of field NUMBER to NUMBER meters from our goal line
265. (definer IDENT (circle (pt (player our	265. Call area within NUMBER m of our

{NUMBER})) NUMBER))	player NUMBER to be IDENT
266. (definer IDENT (midfield our))	266. Let our midfield be called IDENT
267. (definer IDENT (reg (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)) (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER))))	267. Call rectangles -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- or -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT NUMBER
268. (definer IDENT (left (from-goal-line our NUMBER NUMBER)))	268. Let IDENT be left side of field NUMBER to NUMBER meters from our goal line
269. (definer IDENT (midfield our))	269. Call our midfield IDENT
270. (definer IDENT (right (penalty-area opp)))	270. Let IDENT be right side of opponent 's penalty area
271. (definer IDENT (circle (pt (player our {NUMBER})) NUMBER))	271. Let IDENT be circle of NUMBER m radius around player NUMBER
272. (definer IDENT (right-quarter (near-goal-line our)))	272. Let IDENT be right quarter portion of area near our goal line
273. (definer IDENT (near-goal-line our))	273. Let IDENT be area near our goal line
274. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	274. Let IDENT be rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
275. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	275. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- as IDENT
276. (definer IDENT (circle (pt (player our {NUMBER})) NUMBER))	276. Let IDENT be circle of NUMBER meters radius from our player NUMBER
277. (definer IDENT (left-quarter (midfield opp)))	277. Let IDENT be far left side of opponent 's midfield
278. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	278. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- as IDENT
279. (definer IDENT (right-quarter (near-goal-line our)))	279. Let IDENT be near right corner
280. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	280. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT
281. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	281. Define IDENT to be rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
282. (definer IDENT (circle (pt NUMBER NUMBER) NUMBER))	282. Let IDENT be circle of NUMBER meters radius from point -LRB- NUMBER NUMBER -RRB-
283. (definer IDENT (circle (pt (player our {NUMBER})) NUMBER))	283. Let IDENT be area within NUMBER m of player NUMBER
284. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	284. Let IDENT stand for rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
285. (definer IDENT (left-quarter (midfield opp)))	285. Let IDENT be far left side of opponent 's midfield
286. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	286. Call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- to be IDENT
287. (definer IDENT (circle (pt NUMBER NUMBER) NUMBER))	287. Define IDENT to be circle of radius NUMBER around point -LRB- NUMBER NUMBER -RRB-
288. (definer IDENT (from-goal-line opp NUMBER NUMBER))	288. Let IDENT be field NUMBER to NUMBER meters from opponent 's goal line
289. (definer IDENT (left-quarter (near-goal-line our)))	289. Let IDENT be region close to near left corner
290. (definer IDENT (circle (pt (player our	290. Call circle of radius NUMBER meters

{NUMBER})) NUMBER))	around our player NUMBER as IDENT
291. (definer IDENT (right (from-goal-line our NUMBER NUMBER)))	291. Let IDENT be right side NUMBER to NUMBER meters from our goal line
292. (definer IDENT (left-quarter (from-goal-line opp NUMBER NUMBER)))	292. Define IDENT to be far left side of field NUMBER to NUMBER meters from opponent 's goal line
293. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	293. Let us call rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB- as IDENT
294. (definer IDENT (rec (pt NUMBER NUMBER) (pt NUMBER NUMBER)))	294. Let IDENT be rectangle -LSB- -LRB- NUMBER NUMBER -RRB- -LRB- NUMBER NUMBER -RRB- -RSB-
295. (definer IDENT (circle (pt NUMBER NUMBER) NUMBER))	295. Let IDENT be circle of NUMBER m radius centered at -LRB- NUMBER NUMBER -RRB-
296. (definer IDENT (from-goal-line opp NUMBER NUMBER))	296. Call IDENT to be area between distance NUMBER to NUMBER from opponent 's goal line
297. (definer IDENT (left (midfield)))	297. Let IDENT be left midfield
298. (definer IDENT (right (penalty-area opp)))	298. Call opponent 's left penalty area to be IDENT
299. (definer IDENT (midfield opp))	299. Call opponent 's midfield as IDENT
300. (definer IDENT (circle (pt NUMBER NUMBER) NUMBER))	300. Let IDENT be circle within NUMBER meter from point -LRB- NUMBER NUMBER -RRB-

Anexo B

Apresentamos abaixo a gramática dada utilizada nos experimentos realizados.

RULE → CONDITION DIRECTIVE

CONDITION → '(' 'true' ')'
| '(' 'false' ')'
| '(' 'ppos' PLAYER UNUM UNUM REGION ')'
| '(' 'ppos-any' PLAYER REGION ')'
| '(' 'ppos-none' 'our' REGION ')'
| '(' 'ppos-none' 'opp' REGION ')'
| '(' 'bpos' REGION ')'
| '(' 'bowner' PLAYER ')'
| '(' 'playm' 'bko' ')'
| '(' 'playm' 'timeOver' ')'
| '(' 'playm' 'playOn' ')'
| '(' 'playm' 'koOur' ')'
| '(' 'playm' 'koOpp' ')'
| '(' 'playm' 'fkOur' ')'
| '(' 'playm' 'fkOpp' ')'
| '(' 'playm' 'ckOur' ')'
| '(' 'playm' 'ckOpp' ')'
| '(' 'playm' 'gkOur' ')'
| '(' 'playm' 'gkOpp' ')'
| '(' 'playm' 'gcOur' ')'
| '(' 'playm' 'gcOpp' ')'
| '(' 'playm' 'agOur' ')'
| '(' 'playm' 'agOpp' ')'
| IDENT
| '(' '<' (NUM | UNUM) (NUM | UNUM) ')'
| '(' '>' (NUM | UNUM) (NUM | UNUM) ')'
| '(' '==' (NUM | UNUM) (NUM | UNUM) ')'
| '(' '<=' (NUM | UNUM) (NUM | UNUM) ')'
| '(' '!=' (NUM | UNUM) (NUM | UNUM) ')'
| '(' 'and' CONDITION CONDITION ')'
| '(' 'or' CONDITION CONDITION ')'
| '(' 'not' CONDITION ')'

DIRECTIVE → '(' 'do' PLAYER ACTION ')'
| '(' 'dont' PLAYER ACTION ')'

ACTION → '(' 'pos' REGION ')'
| '(' 'home' REGION ')'
| '(' 'mark' REGION ')'
| '(' 'mark' PLAYER ')'
| '(' 'oline' REGION ')'
| '(' 'pass' REGION ')'
| '(' 'pass' PLAYER ')'
| '(' 'dribble' REGION ')'
| '(' 'clear' REGION ')'
| '(' 'shoot' ')'
| '(' 'hold' ')'
| '(' 'intercept' ')'
| '(' 'tackle' PLAYER ')'

```
PLAYER  → '(' 'PLAYER' 'our' '{' UNUM '}' ')'
| '(' 'PLAYER' 'our' '{'UNUM UNUM'}' ')'
| '(' 'PLAYER' 'our' '{'UNUM UNUM UNUM'}' ')'
| '(' 'PLAYER' 'our' '{'UNUM UNUM UNUM UNUM'}' ')'
| '(' 'PLAYER' 'opp' '{' UNUM '}' ')'
| '(' 'PLAYER' 'opp' '{' '0' '}' ')'
| '(' 'PLAYER' 'our' '{' '0' '}' ')'
| '(' 'PLAYER-range' 'our' UNUM UNUM ')'
| '(' 'PLAYER-range' 'opp' UNUM UNUM ')'
| '(' 'PLAYER-exception' 'our' '{' UNUM '}' ')'
| '(' 'PLAYER-exception' 'opp' '{' UNUM '}' ')'

REGION  → '(' 'arc' POINT (NUM | UNUM) (NUM | UNUM) (NUM | UNUM) (NUM | UNUM) ')'
| '(' 'circle' POINT (NUM | UNUM) ')'
| '(' 'null' ')'
| '(' 'reg' REGION REGION ')'
| '(' 'reg-exclude' REGION REGION ')'
| '(' 'field' ')'
| '(' 'half' ('our' | 'opp') ')'
| '(' 'penalty-area' ('our' | 'opp') ')'
| '(' 'goal-area' ('our' | 'opp') ')'
| '(' 'midfield' ')'
| '(' '3midfield' ('our' | 'opp') ')'
| '(' 'near-goal-line' ('our' | 'opp') ')'
| '(' 'from-goal-line' ('our' | 'opp') (NUM | UNUM) (NUM | UNUM) ')'
| '(' 'left' REGION ')'
| '(' 'right' REGION ')'
| '(' 'left-quarter' REGION ')'
| '(' 'right-quarter' REGION ')'
| IDENT

POINT  → '(' 'pt' (NUM | UNUM) (NUM | UNUM) ')'
| '(' 'pt' 'ball' ')'
| '+' POINT POINT
| '-' POINT POINT
| '*' POINT POINT
| '/' POINT POINT
| '(' 'pt-with-ball-attrACTION' POINT POINT ')'
| '(' 'front-of-goal' ('our' | 'opp') ')'
| '(' 'from-goal' ('our' | 'opp') (NUM | UNUM) ')'

STATEMENT  → '(' 'definec' IDENT CONDITION ')'
| '(' 'definer' IDENT REGION ')'
| RULE
```

Anexo C

Apresentamos abaixo uma amostra de gramática extraída com filtros para sinais de pontuação nos dois *corpora*, filtros para remover determinantes <*the*> e <*a*> no corpus de linguagem natural, filtros para utilizar um rótulo em todos os números, e outro em todos os nomes de variáveis, e alinhamento entre palavras e nós, sem remoções. Para facilidade de leitura, as regras sincrônicas estão representadas da seguinte forma. Dada uma regra

$$X \rightarrow <\alpha, \beta>$$

onde $\alpha \in$ MRL e $\beta \in$ NL, escrevemos:

$$\text{MRL} >> X \rightarrow \alpha$$

$$\text{NL} >> X \rightarrow \beta$$

Cada bloco de regras, separado por linhas em branco, foi extraído de uma árvore. Os índices numéricos indicam a bijeção entre os não-terminais da árvore que corresponde àquele bloco.

MRL>> STATEMENT(1)→ ((and (playm play_on) (bpos IDENT)) (do (player our { NUMBER }) (pass (left (penalty-area opp)))))
NL>> STATEMENT(1)→ During normal play when ball is in region IDENT then player NUMBER should pass it to left side of opponent 's penalty area

MRL>> STATEMENT(1)→ ((bpos (right-quarter (near-goal-line opp))) (do (player-range our NUMBER NUMBER) (pass (right (penalty-area opp)))))
NL>> STATEMENT(1)→ If ball is in right quarter of area near opponent 's goal line then players NUMBER NUMBER should pass it to right side of opponent 's penalty area

MRL>> STATEMENT(1)→ ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our { NUMBER }) (pos (pt NUMBER NUMBER))))
NL>> STATEMENT(1)→ If ball is at distance NUMBER to NUMBER from our goal line then position player NUMBER to point -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our { NUMBER }) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
NL>> STATEMENT(1)→ If ball is within NUMBER meters from our goal line then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bpos IDENT) (do (player our { NUMBER }) (home (circle (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)) NUMBER))))

NL>> STATEMENT(1)→ If ball is in IDENT then player NUMBER 's default position should be in circle of radius NUMBER from -LRB- NUMBER NUMBER -RRB- and its ball attraction should be -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bowner (player our { NUMBER })) (do (player our { NUMBER }) (pass (player our { NUMBER NUMBER }))))
NL>> STATEMENT(1)→ If player NUMBER has ball then it should pass to players NUMBER or NUMBER

MRL>> STATEMENT(1)→ ((bpos (half our)) (do (player our { NUMBER }) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
NL>> STATEMENT(1)→ If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our { NUMBER }) (pos (from-goal opp NUMBER))))
NL>> STATEMENT(1)→ If ball is at distance NUMBER to NUMBER from opponent 's goal line then position player NUMBER at NUMBER meters from opponent 's goal

MRL>> STATEMENT(1)→ ((and (playm play_on) (bpos IDENT)) (do (player our { NUMBER }) (pass (left (penalty-area opp)))))
NL>> STATEMENT(1)→ During normal play if ball is in IDENT then player NUMBER should pass it to left side of opponent 's penalty area

MRL>> STATEMENT(1)→ ((and (bowner (player our { NUMBER })) (bpos IDENT)) (do (player our { NUMBER }) (clear IDENT)))
NL>> STATEMENT(1)→ If ball is in IDENT then whoever owns ball should clear it from IDENT

MRL>> STATEMENT(1)→ ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our { NUMBER }) (pos (pt NUMBER NUMBER))))
NL>> STATEMENT(1)→ If ball is at distance NUMBER to NUMBER from our goal line then position player NUMBER at point -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bpos (from-goal-line our NUMBER NUMBER)) (do (player our { NUMBER }) (pos (from-goal our NUMBER))))
NL>> STATEMENT(1)→ If ball is between NUMBER to NUMBER meters from our goal line then position our player NUMBER at NUMBER meters from our goal

MRL>> STATEMENT(1)→ ((and (bowner (player our { NUMBER })) (bpos (penalty-area opp))) (do (player our { NUMBER }) (shoot)))
NL>> STATEMENT(1)→ If ball is in opponent 's penalty area then whoever has ball should shoot

MRL>> STATEMENT(1)→ ((bpos (half our)) (do (player our { NUMBER }) (pos (pt-with-ball-attraction (pt NUMBER NUMBER) (pt NUMBER NUMBER)))))
NL>> STATEMENT(1)→ If ball is in our half then position player NUMBER at -LRB- NUMBER NUMBER -RRB- and set its ball attraction to -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((not (bpos (goal-area our))) (dont (player our { NUMBER NUMBER }) (intercept)))
NL>> STATEMENT(1)→ If ball is not in our goal area then players NUMBER and NUMBER should not intercept it

MRL>> STATEMENT(1)→ (definec IDENT (bpos (circle POINT(4) NUMBER)))
NL>> STATEMENT(1)→ Define IDENT to be condition that ball is within distance NUMBER from POINT(4) player NUMBER
MRL>> POINT(4)→ (pt PLAYER(5))
NL>> POINT(4)→ PLAYER(5)
MRL>> PLAYER(5)→ (player our { NUMBER })
NL>> PLAYER(5)→ our

MRL>> STATEMENT(1)→ ((true) (do (player our { NUMBER }) (home (pt NUMBER NUMBER))))
NL>> STATEMENT(1)→ default position of player NUMBER should be -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((bpos (from-goal-line opp NUMBER NUMBER)) (do (player our { NUMBER }) (pos (pt NUMBER NUMBER))))
NL>> STATEMENT(1)→ If ball is at distance NUMBER to NUMBER from opponent 's goal line then our player NUMBER should be positioned at -LRB- NUMBER NUMBER -RRB-

MRL>> STATEMENT(1)→ ((or (playm gc_our) (playm gk_our) (playm fk_our)) (do (player our { NUMBER }) (clear (from-goal-line our NUMBER NUMBER))))
NL>> STATEMENT(1)→ During our goalie catch goal kick or free kick player NUMBER should clear ball up to NUMBER meters from our goal line

MRL>> STATEMENT(1)→ (definec IDENT CONDITION(2))
NL>> STATEMENT(1)→ Call IDENT as CONDITION(2) condition
MRL>> CONDITION(2)→ (true)
NL>> CONDITION(2)→ true

MRL>> STATEMENT(1)→ (definec IDENT (and (bowner (player our { NUMBER })) (bpos IDENT)))
NL>> STATEMENT(1)→ Let IDENT be condition that player NUMBER has ball and ball is in IDENT

Referências bibliográficas

ACOMB, K.; BLOOM, J.; DAYANIDHI, K.; *et al*. "Technical support dialog systems: issues, problems, and solutions". In: **North American Chapter Of The Association For Computational Linguistics**, pp. 25-31, Rochester, NY, 2007.

AHO, A. V.; ULLMAN, J. D. **The Theory of Parsing, Translation, and Compiling**. 1 ed. Englewood Cliffs, NJ, Prentice Hall Professional Technical Reference, 1972.

BETHARD, S.; MARTIN, J. H. "CU-TMP: temporal relation classification using syntactic and semantic features". In: **Proceedings of the 4th International Workshop on Semantic Evaluations,** pp. 129-132, Praga, 2007.

BRILL, E. "Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging**"**. In: **Proceedings of the 3rd Workshop on Very Large Corpora**, pp. 1-13, Cambridge, MA, 1995.

BROWN, P. F.; PIETRA, V. J.; PIETRA, S. A.; *et al*. "The mathematics of statistical machine translation: parameter estimation". **Computational Linguistics**, v. 19, n. 2, pp. 263-311, 1993.

CHIANG, D. "A hierarchical phrase-based model for statistical machine translation". In: **Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics**, pp. 263-270, Ann Arbor, MI, 2005.

CHIANG, D.; KNIGHT, K.; WANG, W. "11,001 new features for statistical machine translation". In: **Human Language Technology Conference**, pp. 218-226, Boulder, CO, 2009.

COLLINS, M. "A new statistical parser based on bigram lexical dependencies". In: **Proceedings of the ThirtyFourth Annual Meeting of the Association for Computational Linguistics**, pp. 184-191, Santa Cruz, CA, 1996.

COLLINS, M. "Three Generative, Lexicalised Models for Statistical Parsing". In: **Proceedings of the 35th annual meeting on Association for Computational Linguistics,** pp. 16-23, Madri, 1997.

COLLINS, M. "Discriminative Reranking for Natural Language Parsing". In: **Proceedings of the Seventeenth International Conference on Machine Learning**, pp. 175-182, Stanford, CA, 2000.

COLLINS, M. "Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms". In: **Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10**, pp. 1-8, Filadélfia, PA, 2002.

CRISTIANINI, N.; SHAWE-TAYLOR, J. **An introduction to support vector machines: and other kernel-based learning methods**. 1 ed. Cambridge, Cambridge University Press, 2000.

DENERO, J.; KLEIN, D. "Tailoring Word Alignments to Syntactic Machine Translation". In: **Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics**, pages 17–24, Praga, 2007.

FATTAH, M. A.; REN, F. "GA, MR, FFNN, PNN and GMM based models for automatic text summarization". **Computer Speech and Language**, v. 23, n. 1, pp. 126-144, 2009.

FOX, H. J. "Phrasal cohesion and statistical machine translation". In: **Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10**, pp. 304-311, Filadélfia, PA, 2002.

GALLEY, M.; HOPKINS, M.; KNIGHT, K.; *et al*. "What's in a translation rule?". In: **Proceedings of HLTNAACL**, pp.273-280, Boston, MA, 2004.

GE, R. **Learning semantic parsers using statistical syntactic parsing techniques.** Ph.D. Dissertation Proposal, University of Texas at Austin, Austin, TX, 2006.

GE, R.; MOONEY, R. J. "A statistical semantic parser that integrates syntax and semantics". In: **Proceedings of the Ninth Conference on Computational Natural Language Learning**, pp. 9-16, Ann Arbor, MI, 2005.

GE, R.; MOONEY, R. J. "Learning a compositional semantic parser using an existing syntactic parser". In: **Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2**, pp. 611-619, Suntec, Singapura, 2009.

GILDEA, D.; JURAFSKY, D. "Automatic labeling of semantic roles". **Computational Linguistics**, v. 28, n. 3, pp. 245, 2002.

HOPCROFT, J. E.; ULLMAN, J. D. **Introduction to Automata Theory. Languages and Computation**. 1 ed. Reading, MA, Addison-Wesley, 1979.

JOHNSTON, M.; BANGALORE, S.; VASIREDDY, G.; *et al*. "MATCH: an architecture for multimodal dialogue systems". In: **Proceedings of the 40th Annual Meeting on Association for Computational Linguistics**, pp.376-383, Filadélfia, PA, 2002.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**. 2 ed., Upper Saddle River, NJ, Prentice Hall, 2008.

KATE, R. J. **Learning for semantic parsing with kernels under various forms of supervision**. Ph.D. dissertation, University of Texas at Austin, Austin, TX, 2007.

KATE, R. J. "Transforming meaning representation grammars to improve semantic parsing. In: **Proceedings of the Twelfth Conference on Computational Natural Language Learning**, pp. 33-40, Manchester, 2008.

KATE, R. J.; MOONEY, R. J. "Using string-kernels for learning semantic parsers". In: **Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics**, pp. 913-920, Sydney, 2006.

KATE, R. J.; MOONEY, R. J. "Learning language semantics from ambiguous supervision". In: **AAAI Conference On Artificial Intelligence**, pp. 895-900, Vancouver, 2007.

KOEHN, P.; HOANG, H.; BIRCH, A.; *et al*. "Moses: open source toolkit for statistical machine translation". In: **Proceedings of the Twelfth Conference on Computational Natural Language Learning**, pp. 177-180, Praga, 2007.

KOEHN, P.; OCH, F. J.; MARCU, D. "Statistical phrase-based translation". In: **Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1**, pp. 48-54, Edmonton, 2003.

LI, Z.; CALLISON-BURCH, C.; DYER, C.; *et al*. "Joshua: an open source toolkit for parsing-based machine translation". In: **Proceedings of the Fourth Workshop on Statistical Machine Translation**, pp. 135-139, Atenas, 2009.

LIANG, P.; TASKAR, B.; KLEIN, D. "Alignment by agreement". In: **Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics**, pp. 104-111, New York, NY, 2006.

LOPEZ, A. "Statistical machine translation". **ACM Computing Surveys**, v. 40, n. 3, pp. 1-49, 2008.

LYCAN, W. G. **Philosophy of language: a contemporary introduction**. 1 ed. New York, NY, Routledge, 2000.

MANNING, C. D.; SCHUETZE, H. **Foundations of Statistical Natural Language Processing**. 1 ed., Cambridge, MA, The MIT Press, 1999.

MARTINICH, A. (ed.) **The Philosophy of Language**. 2 ed. New York, NY, Oxford University Press, 1990.

MOONEY, R. J. "Learning for semantic parsing". **Lecture Notes in Computer Science**, v. 4394, pp. 311-324, 2007.

NARADOWSKY, J.; GOLDWATER, S. "Improving morphology induction by learning spelling rules". In: **Proceedings of the 21st international jont conference on Artifical intelligence,** pp. 1531-1536, Pasadena, CA, 2009.

OCH, F. J.; NEY, H. "A systematic comparison of various statistical alignment models". **Computational Linguistics**, v. 29, n. 1, pp. 19-51, 2003.

PANG, B.; LEE, L. "Opinion Mining and Sentiment Analysis". **Foundations and Trends in Information Retrieval**, v. 2, n. 1-2, pp. 1-135, 2008.

PARR, T. J. **The Definitive ANTLR Reference: Building Domain-Specific Languages**. 1 ed. Raleigh, NC, The Pragmatic Bookshelf, 2007.

PRASAD, J.; PAEPCKE, A. "Coreex: content extraction from online news articles". In: **Proceeding of the 17th ACM conference on Information and knowledge management**, pp.1391-1392, Napa Valley, CA, 2008.

SIDDHARTHAN, A.; COPESTAKE, A. "Generating research websites using summarisation techniques". In: **Human Language Technology Conference**, pp. 5-8, Columbus, OH, 2008.

SMITH, D. A.; EISNER, J. "Bootstrapping feature-rich dependency parsers with entropic priors". In: **Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning**, pp. 667-677, Praga, 2007.

SONG, L.; CHENG, X.; GUO, Y.; *et al*. "ContentEx: a framework for automatic content extraction programs". In: **Proceedings of the 2009 IEEE international conference on Intelligence and security informatics**, pp. 188-190. Richardson, TX, 2009.

STEEDMAN, M. **The syntactic process**. 1 ed., Cambridge, MA, MIT Press, 2000.

VANDERWENDE, L.; SUZUKI, H.; BROCKETT, C.; *et al*. "Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion". **Information Processing and Management: an International Journal**, v. 43, n. 6, pp. 1606-1618, 2007.

VIJAY-SHANKER, K.; WEIR, D. J. "The equivalence of four extensions of context-free grammars". **Mathematical Systems Theory**, v. 27, n. 6, pp. 511-546, 1994.

WONG, Y. W. **Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques**. Ph.D. dissertation, University of Texas at Austin, Austin, TX, 2007.

WONG, Y. W.; MOONEY, R. J. "Learning for semantic parsing with statistical machine translation". In: **Human Language Technology Conference**, pp.439-446, New York, NY, 2006.

ZELLE, J.; MOONEY, R. J. " Learning to parse database queries using inductive logic programming". In: **Proceedings of the National Conference on Artificial Intelligence**, pp. 1050-1055, Palo Alto, CA, 1996.

ZETTLEMOYER, L. S.; COLLINS, M. "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars". In: **Proceedings of 21st Conference on Uncertainty in Artificial Intelligence**, pp. 658-666, Edinburgo, 2005.

ZETTLEMOYER, L. S.; COLLINS, M. "Online learning of relaxed CCG grammars for parsing to logical form**"**. In: **Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning**, pp. 678-687, Praga, 2007.

ZETTLEMOYER, L. S.; COLLINS, M. "Learning context-dependent mappings from sentences to logical form". In: **Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2**, pp. 976-984, Suntec, Singapura, 2009.