

Gramática de Unificação Funcional e Geração Sentencial de Português

ALEXSANDRO SANTOS SOARES¹, MARIA DAS GRAÇAS VOLPE NUNES²

¹Departamento de Ciência da Computação
Universidade Federal de Goiás
alex@catalao.ufg.br

²ICMC- Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo (USP)
Caixa Postal 668, 13560-970 São Carlos, SP, Brasil
mdgvnune@icmc.sc.usp.br

Abstract. This paper summarizes the results achieved with the specification of a portuguese monosentence generator. In our approach, we consider the use of constraint-based linguistic theories in an unification-based framework for the surface generation of simple phrases from the UNL (Universal Networking Language).

Palavras-chave: Realização morfossintática, Geração de Línguas Naturais, Gramática de Unificação Funcional, Universal Networking Language.

1 Introdução

A geração de língua natural (GLN) é uma subárea da inteligência artificial e da linguística computacional que se ocupa dos sistemas computacionais capazes de produzir textos compreensíveis em português ou em outras línguas humanas [10]. O processo de geração geralmente inicia-se com a utilização de algum formalismo de representação semântica, contendo a informação a ser passada, e termina com a construção do texto em alguma língua natural.

Uma das etapas do GLN é a chamada *geração sentencial* ou *lexicalização*, que consiste na geração do texto propriamente dito, isto é, na escolha das palavras e construções sintáticas que efetivamente veiculem a mensagem a ser transmitida. As abordagens existentes para a modelagem desta etapa variam de sistemas simples e pouco flexíveis, valendo-se essencialmente de textos pré-fabricados¹, a sistemas sofisticados que empregam *gramáticas computacionais de geração* (GCG).

As GCG permitem grande flexibilidade aos geradores sentenciais. Esta flexibilidade não é, no entanto, sem custo, pois a construção e a manutenção de tais gramáticas envolvem teorias linguísticas e formalismos computacionais variados.

As teorias linguísticas permitem-nos descrever os recursos das línguas naturais, assim como construir modelos que expliquem *como* uma língua natural funciona. Estes modelos são as gramáticas formais da língua e ajudarão a responder perguntas como: por que certas combinações de palavras formam orações e outras não, ou por que uma oração pode possuir alguns significados e não outros.

Os formalismos computacionais fornecem métodos e

notações diferentes para representar os recursos linguísticos em um computador sem depender exclusivamente da teoria linguística utilizada. Este último ponto é importante, pois embora as teorias linguísticas pareçam estar comprometidas com um formalismo particular, é necessário abstrair e separar as noções da teoria linguística daquelas presentes no formalismo [3]. Isto possibilita a codificação de gramáticas fundamentadas em teorias linguísticas diferentes (e até mesmo divergentes) utilizando o mesmo formalismo computacional e, *mutatis mutandis*, uma mesma teoria linguística poderia ser implementada em vários formalismos computacionais diferentes.

Neste trabalho apresentamos uma síntese dos resultados encontrados em [14], que objetivava investigar os requisitos necessários para geração de sentenças simples do português a partir de uma representação semântica. A representação escolhida foi a UNL (Universal Networking Language) e o formalismo computacional utilizado para a implementação da GCG foi uma *gramática baseada em restrições* (GBR), ou mais especificamente a *gramática de unificação funcional* (GUF).

Este artigo está organizado da seguinte forma. Na seção 2 apresentamos uma breve exposição da UNL e então seguimos com a enumeração das características desejáveis em um formalismo computacional para a GLN. Na seção 4 apresentamos a gramática de unificação funcional juntamente com um exemplo de sua aplicação para a geração. Nas seções posteriores apresentamos em detalhes a arquitetura utilizada no nosso gerador e os problemas encontrados em sua implementação.

¹Sentenças ou trechos que foram previamente embutidas no gerador e que serão utilizadas nas saídas.

```

tim(begin.@entry.@past, long_ago)
mod(city.@def, Babylon)
ppl(begin.@entry.@past, city.@def)
agt(begin.@entry.@past, people.@def)
obj(begin.@entry.@past, build)
obj(build, tower.@indef)
agt(build, people.@def)
aoj(huge, tower.@indef)
soj(seem.@past, tower.@indef)
obj(seem.@past, reach.@begin-soon)
obj(reach.@begin-soon, tower.@indef)
opl(reach.@begin-soon, heaven.@def.@pl)

```

Figura 1: Representação UNL para a sentença 1.

2 Universal Network Language

A Universal Network Language (UNL) é uma língua eletrônica (interlândia) projetada para computadores com o intuito de expressar e trocar qualquer tipo de informação [17].

A UNL representa informação, isto é significado, sentença por sentença. A informação na sentença é representada como um hipergrafo tendo conceitos como nós e relações como arcos. Este hipergrafo é também representado como um conjunto de relações binárias dirigidas entre cada dois dos conceitos presentes na sentença.

Mais especificadamente, a UNL é uma interlândia que serve para descrever aspectos essenciais do significado das sentenças, tais como as relações semânticas que podem ser representadas por relações formais (morfológicas ou sintáticas) entre palavras de uma sentença. A UNL é capaz de representar de maneira uniforme o significado de orações que podem ser descritas e explicadas a partir de diferentes modelos gramaticais, pois se baseia na análise gramatical das mesmas e, logo, na representação superficial de suas estruturas textuais. Uma exposição mais detalhada da UNL e sua manifestação em português encontra-se em Sossolote et alli [15].

Um exemplo da UNL que representa a sentença 1 abaixo é dada na figura 1. Na figura 2 vemos a mesma representação na forma de um grafo.

1. Há muito tempo, na cidade de Babilônia, os homens começaram a construir uma imensa torre, que parecia quase atingir os céus.

3 Características desejáveis em um formalismo para a GLN

A tarefa de geração de linguagem requer a manipulação de estruturas complexas: representação de constituintes lingüísticos, entradas lexicais, suas notações semânticas, as ligações entre objetos lingüísticos e suas restrições semânticas. Além

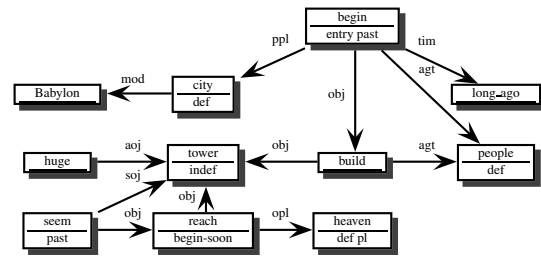


Figura 2: Grafo UNL

disso, geração consiste em se escolher entre diferentes dispositivos lingüísticos a fim de satisfazer um objetivo comunicativo e então construir uma estrutura lingüística que combine esses vários dispositivos numa sentença ou texto coerente. Dessa forma, um formalismo de geração está envolvido em constantes atividades de escolha, de tomada de decisão e de construção de uma estrutura lingüística a partir da composição de componentes menores. Segundo Elhadad [3], um conjunto de critérios para se determinar a adequação de um formalismo para a tarefa de geração inclui:

Descrição de entrada conceitual: o formalismo deve permitir a especificação de características semânticas e/ou pragmáticas, já que a entrada para o gerador de superfície deve ser desse tipo.

Especificação parcial da entrada: o gerador deve ser capaz de funcionar na ausência de algumas informações semânticas ou pragmáticas, muitas vezes não previstas na gramática subjacente.

Expressividade da entrada: a entrada deve ser genérica o suficiente para ser tratada por diferentes esquemas de representação de conhecimento, permitindo o desenvolvimento de sistemas de geração portáteis.

Descrição funcional de objetos lingüísticos: deve ser possível especificar como objetos lingüísticos se relacionam com objetos conceituais e como estes se relacionam funcionalmente com outros objetos lingüísticos.

Descrição estrutural de objetos lingüísticos: deve ser possível expressar a estrutura de objetos lingüísticos, especificando constituintes e como estes são intercalados.

Suporte para composicionalidade: possibilidade de relacionar a construção da estrutura lingüística com a estrutura da entrada conceitual.

Declaratividade: restrições sobre como objetos lingüísticos se combinam e se relacionam com a entrada devem ser expressas declarativamente.

Ordem de tomada de decisão: a ordem de processamento, *bottom-up*, *top-down*, *left-to-right*, ou qualquer outra variação, pode influenciar significativamente o modo como as restrições interagem entre si.

Eficiência: o formalismo deve ser tal que permita a definição de procedimentos eficientes.

Reversibilidade: é desejável, porém reconhecidamente difícil, que um sistema de PLN use uma única gramática tanto para análise quanto para geração.

Esse conjunto de características é desejável, mas difícil de ser alcançado por um único formalismo.

4 Gramática de Unificação Funcional

A gramática de unificação funcional (GUF) foi introduzida em 1979, por Martin Kay [5], sendo uma instância dos formalismos baseados em unificação.

O sucesso do uso da GUF para a geração é motivado principalmente por sua maior ênfase na perspectiva funcional do que na estrutural, uma vez que a geração começa com considerações funcionais (como mapear o significado para a forma) e termina com considerações estruturais (como organizar a forma) [9, 1].

Algumas características que distinguem a GUF são [3]:

Representação uniforme: a GUF conta com uma estrutura de dados simples chamada *descrição funcional* (DF) para codificar uniformemente fatores sintáticos, semânticos e pragmáticos. As DFs capturam tanto aspectos funcionais quanto estruturais de uma descrição lingüística.

Unificação: GUF conta com uma única operação, a unificação, para manipular as DFs e tomar decisões. A unificação é utilizada em várias teorias lingüísticas (Head Phrase Structure Grammars, General Phrase Structure Grammars, Lexical Functional Grammars [13]). Devido à unificação, não há distinção entre tomada de decisão e realização, comum em outros formalismos.

Ordem de decisão flexível: sendo a unificação um processo monotônico bidirecional, é possível tomar decisões em qualquer ordem. Esta característica torna mais fácil o processo de escrita das gramáticas (não há preocupação sobre como uma restrição interage com outras decisões) e também mais modular (cada restrição só precisa ser especificada uma vez). A flexibilidade na tomada de decisão também é importante porque permite o processamento de entrada mista, contendo tanto restrições semânticas quanto lingüísticas.

Devido a algumas limitações da GUF, como quanto a eficiência, expressividade e interação entre planejamento

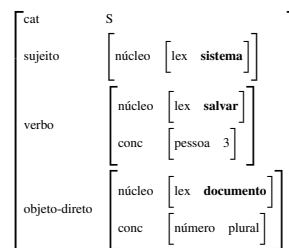


Figura 3: Descrição funcional da sentença *O sistema salva os documentos*.

de conteúdo e realização, Elhadad [3] propõe o Formalismo de Unificação Funcional - FUF, que é uma extensão e uma implementação de GUFs. O autor apresentou FUF em conjunto com SURGE (Surface Grammar for English) [4], que é a implementação, em FUF, de uma gramática de realização sintática do inglês, portátil e independente de domínio. Nesse sentido, sua substituição por uma (sub)gramática do português é potencialmente viável nesse ambiente computacional.

4.1 Exemplo de geração

Para ilustrar o mecanismo de geração, usaremos a DF apresentada na figura 3 como uma especificação de entrada, e a DF da figura 4 como a GUF. Esta gramática, representada através de uma matriz de atributos e valores, suporta frases simples com verbos transitivos diretos no presente do indicativo.

No nível superior, esta gramática fornece alternativas para sentenças (cat S), sintagmas nominais (cat SN) e sintagmas verbais (cat SV). No nível da sentença, ela suporta as funções oracionais de sujeito, verbo e objeto direto. Ela também provê a concordância entre sujeito e verbo, feita pela coindexação do traço *conc*. No nível do sintagma nominal, observam-se a concordância nominal, novamente através da coindexação, entre o determinante e o núcleo, e a escolha no nível do artigo entre as formas singular e plural do artigo definido masculino.

A especificação de entrada mostrada na figura 3 indica que esta sentença (*O sistema salva os documentos*) possui como sujeito *um determinado sistema* e como objeto direto *os documentos*. O verbo indica que o sistema está salvando os documentos no presente momento.

Para produzir uma sentença, a especificação de entrada é unificada com a gramática G mostrada na figura 4. Isto requer múltiplos passos através da gramática. O primeiro passo unifica a DF de entrada com o nível "S" da gramática. O resultado deste processo é mostrado na figura 5, onde as adições feitas pela gramática à DF de entrada estão em negrito.

Neste passo, várias informações de G que estavam no nível superior incorporam-se à DF de entrada. Por exem-

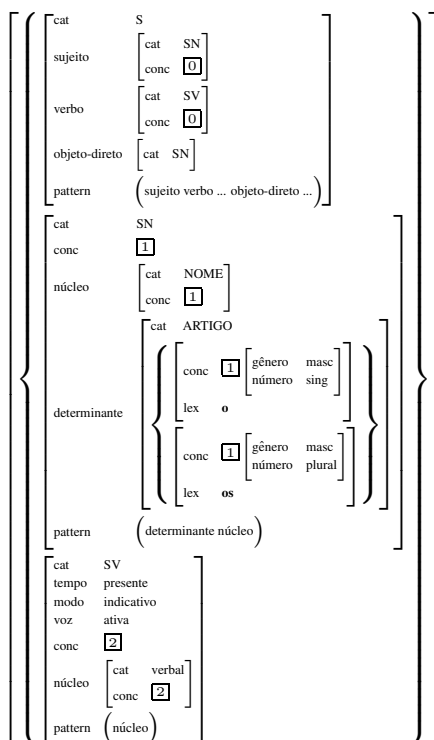


Figura 4: Uma gramática de unificação funcional G.

plo, o fato que sujeito e verbo devem concordar. Outra informação adicionada foi a maneira como a sentença deve ser linearizada, representada pelo traço *pattern*. Neste traço vê-se a presença de pontos. Estes pontos marcam a posição dentro da sentença onde novas construções sintáticas poderiam ser adicionadas. Na configuração em questão, é indicado que após o verbo poderiam ser adicionados vários adjuntos (por exemplo, o advérbio “repetidamente”), o mesmo se dando com o objeto direto.

Da maneira como a gramática foi construída, ela necessariamente deve possuir um sujeito e sempre na primeira posição (ela inibe, por exemplo, a topicalização do objeto direto). Além disso, ela também impede que qualquer outra coisa seja entreposta entre o sujeito e o verbo.

No próximo passo, o mecanismo de geração reentra recursivamente na gramática G para cada um dos subconstituintes. Assim, ela entrará no nível do SN duas vezes (uma para o sujeito e outra para o objeto direto) e uma vez no nível do SV (para o verbo). A DF que resulta destes passos é mostrada na figura 6.

A descrição funcional resultante é enviada para o processo de linearização. Esse processo, por sua vez, acessa recursivamente todos os traços *pattern* da DF, iniciando no nível superior. Lá ele encontra a seguinte configuração (**sujeito verbo ... objeto-direto ...**), indicando que ele deve procurar dentro das DFs do sujeito, verbo e do objeto-direto,

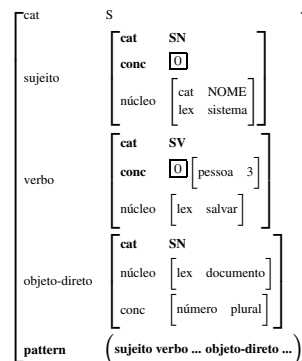


Figura 5: DF da entrada enriquecida pela gramática G.

nesta ordem, seus traços *pattern*, e assim sucessivamente. O processo de linearização termina quando não forem mais encontrados os traços *pattern* e, nesse caso, procura-se pelo traço *lex*, que indica o item lexical a ser usado, caso ele exista.

Terminado o processo de linearização, obtém-se a lista mostrada na figura 7. Esta lista será enviada para o processo responsável pelas sínteses ortográfica e morfológica, que fará as mudanças necessárias para realizá-la como uma sentença. Note que os itens lexicais já estão na ordem em que devem aparecer na sentença final.

5 A arquitetura do sistema

A figura 8 apresenta a arquitetura utilizada no sistema de geração de UNL para português. A escolha desta arquitetura baseou-se naquelas citadas por Elhadad [3], Temizsoy [16], Korkmaz [6] e Sérasset e Boitet [12].

5.1 Mapeamento do grafo UNL para árvore UNL

A UNL é um hipergrafo dirigido onde cada nó pode, ele mesmo, ser outro grafo [8]. Um destes nós é marcado com “entry” e indica a entrada para este hipergrafo. Entretanto, a maioria dos técnicas e algoritmos disponíveis requerem uma estrutura de árvore como mecanismo representacional básico. Assim, necessitamos transformar este hipergrafo em uma árvore, o que é feito pelo módulo de mapeamento do grafo UNL para uma árvore UNL (figura 9). A função principal deste módulo é converter um grafo UNL da sentença dada como entrada em árvore UNL que mantenha a direcionalidade e a rotulagem dos nós do grafo.

A primeira ação do módulo é varrer a sentença UNL de entrada para fazer a análise sintática (*parser*) e a validação da estrutura. A verificação da existência de um nó marcado com “entry” é feita nesta fase. Caso alguma inconformidade seja encontrada o processo todo é abortado e mensagens de erro são emitidas. Se a validação resultar correta, é gerado uma representação interna mais adequada para o

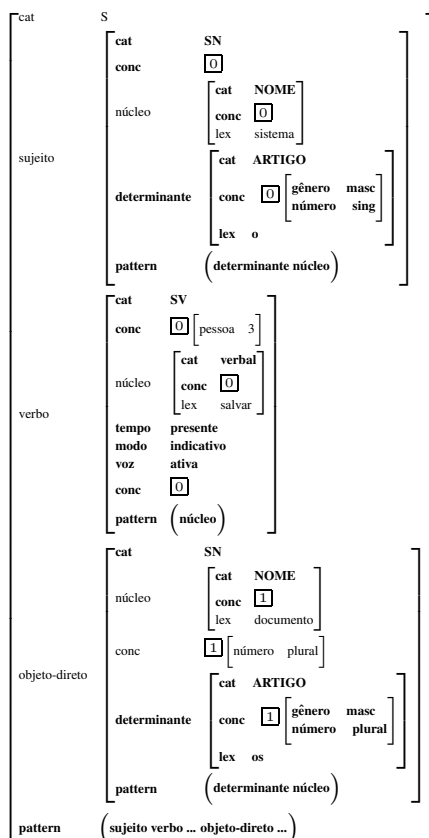


Figura 6: DF da entrada resultante da unificação com a gramática G.

processamento dos outros módulos.

Em seguida, a representação interna gerada pelo *parser* é enviada ao módulo de seleção lexical que, consultando o dicionário UW²-Português, converte as UWs nos vários lexemas candidatos. Pode haver mais que um lexema candidato e assim, o resultado desta fase é a lexicalização da representação interna, com os vários lexemas candidatos acrescidos aos nós do grafo. As entradas destes lexemas no dicionário UW-Português também contribuem com descrições morfosintáticas e semânticas dos lexemas selecionados.

A última fase do processamento deste módulo consiste em fazer a conversão do grafo lexicalizado em uma árvore. Esta conversão deverá levar em conta a direção e os rótulos presentes em cada nó do grafo original. Para este fim, usaremos o algoritmo apresentado por Sérasset e Boitet [12] e visto na figura 10. Este algoritmo divide os nós que são alvos de mais de um nó, invertendo a direção do menor número possível de arestas.

²Uma UW (Universal Word) é uma entrada lexical que especifica um conceito individual em termos de sua relação com outros conceitos, assim como a correspondência de cada conceito com a forma que ele deve tomar

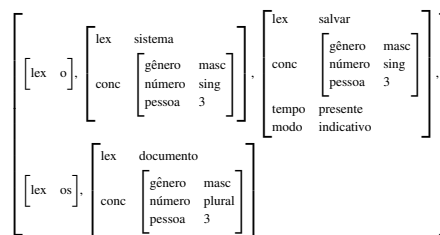


Figura 7: A saída do processo de linearização.

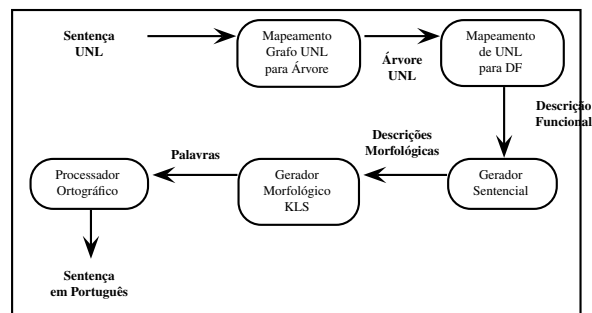


Figura 8: Arquitetura do sistema de geração UNL-Português

Para mostrar este algoritmo em ação, considere a sentença 1, cujo grafo UNL é dado na figura 2.

Após a aplicação do algoritmo de conversão obtemos a árvore mostrada na figura 11. Note a duplicação dos nós contendo as UWs *tower.@indef* e *people.@def*, as únicas que são alvos de mais de uma RL no grafo. Note também que somente duas arestas na árvore tiveram suas direções invertidas, mantendo as mesmas orientações do grafo UNL.

5.2 Mapeamento da árvore UNL para uma descrição funcional

O passo seguinte é mapear a árvore UNL em uma descrição funcional que servirá de entrada para o módulo de geração propriamente dito.

A árvore apresentada a este módulo é essencialmente semântica em natureza, excetuando-se as UWs que já possuem parte das informações sintáticas necessárias para o seu processamento. Assim, cabe a este módulo mapear as estruturas semânticas em estruturas sintáticas. Vejamos alguns dos problemas que deverão ser tratados aqui:

- Os rótulos de relações não possuem um mapeamento bijetivo com as funções sintáticas da oração em português. A mesma RL pode ser mapeada para funções sintáticas diferentes dependendo do contexto em que ela aparece. É o caso de *soj* que pode se manifes-

em determinada língua.

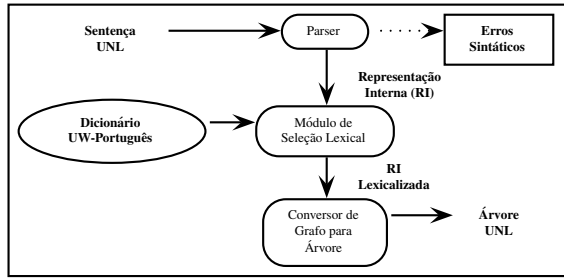


Figura 9: Módulo responsável pelo mapeamento de um grafo UNL para uma árvore

Seja Σ o conjunto de nós de G , R o conjunto de rótulos, A a árvore criada e N o conjunto de nós de A . Defina o grafo $G = \{(a, b, r) | a \in \Sigma, b \in \Sigma, r \in R\}$ como um conjunto de arestas dirigidas rotuladas e defina $L_a = \{(n_G, n_A) | n_G \in \Sigma \text{ e } n_A \in N\}$ como a lista associativa que estabelece a correspondência entre os nós da árvore e os nós do grafo.

$e_G \in \Sigma$ é o nó de entrada em G
 $e_A \leftarrow \text{CriaNovoNoArvore}(e_G, \text{entry})$
 $A \leftarrow e_A; N \leftarrow \{e_A\}; L_a \leftarrow \{e_G, e_A\}$
enquanto $G \neq \emptyset$ **faça**
 se existe (a, b, r) **em** G **tal que** $(a, a_A) \in L_a$ **então**
 $G \leftarrow G \setminus (a, b, r);$
 $b_A \leftarrow \text{CriaNovoNoArvore}(b, r);$
 $L_a \leftarrow L_a \cup \{(b, b_A)\};$
 seja $a_A \in N$ **tal que** $(a, a_A) \in L_a;$
 adicione b_A aos filhos de $a_A;$
 senão se existe (a, b, r) **em** G **tal que** $(b, b_A) \in L_a$ **então**
 $G \leftarrow G \setminus (a, b, r);$
 $a_A \leftarrow \text{CriaNovoNoArvore}(a, r^{-1});$
 $L_a \leftarrow L_a \cup \{(a, a_A)\};$
 seja $b_A \in N$ **tal que** $(b, b_A) \in L_a;$
 adicione a_A aos filhos de $b_A;$
 senão encerre com erro ("grafo desconexo");

Figura 10: Algoritmo de conversão de grafo UNL para árvore UNL [12]

tar sintaticamente como sujeito, predicativo do sujeito, adjunto adnominal ou objeto direto em virtude do contexto que o circunda. Decidir entre uma função ou outra é difícil, pois entram nesta discussão tanto informações sintáticas (nem sempre disponíveis no dicionário) quanto informações semânticas (por exemplo, se um substantivo é concreto e animado). Por outro lado, a mesma função morfossintática pode ser exercida por RLs diferentes. O sujeito, por exemplo, se manifesta na UNL pelos RLs *soj*, *obj*, *agt*, *cau*, *ins* e *met* [15].

- Conforme notado por Martins [8] e Sérasset [11], a própria UNL, na sua forma atual, não possui todos os atributos multilinguísticos para ser considerada verdadeiramente independente dos pares de línguas usadas. Por problemas de localização cultural, alguma

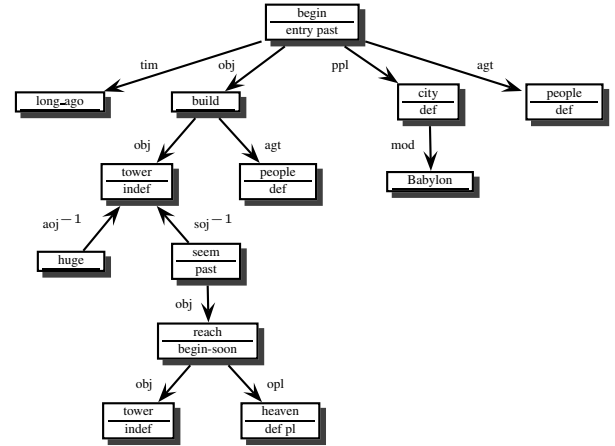


Figura 11: Árvore UNL

informação crucial pode estar ausente da sentença UNL, dependendo da língua fonte da mensagem (por exemplo sexo, modalidade, número, determinação, formas de tratamento, etc.).

Uma maior compreensão do funcionamento interno deste módulo ainda depende de uma investigação mais profunda. Por hora, um caminho que nos parece viável é aquele seguido por Temizsoy [16] que faz o mapeamento de expressões TMR³ (Text Meaning Representations) para DFs. Esta arquitetura é dividida em dois módulos: um de seleção lexical responsável pelo processamento dos lexemas candidatos e outro chamado de módulo de aplicação de regras de mapeamento. Este último, reúne todas as regras de mapeamentos que serão aplicadas à sentença TMR sendo processada e também à estrutura sintática sendo construída. Estas regras são compostas de condições e ações. Se as condições especificadas forem satisfeitas tanto pela sentença TMR quanto pela DF representando a estrutura sintática, então as ações correspondentes são realizadas sobre a DF, atualizando-a.

5.3 O gerador sentencial

A figura 12 mostra a arquitetura interna do gerador sentencial. O seu funcionamento já foi explicado na subseção 4.1. Aqui somente comentaremos sobre os problemas encontrados no desenvolvimento de uma gramática de unificação funcional.

Existem várias questões a serem resolvidas quando se deseja construir uma gramática computacional de geração, a saber:

1. Quais fenômenos linguísticos serão tratados? Dito de

³TMR foi desenvolvida pelo projeto Microcosmos da New Mexico State University e assemelha-se a UNL, embora possua formas mais ricas para a representação dos conhecimentos linguísticos e de mundo.

outra forma, qual a cobertura gramatical que se pretende?

2. Qual tipo ou quais tipos de teorias linguísticas melhor espelham esta cobertura?
3. Qual nível de abstração será exigido da entrada?
4. Existe algum formalismo computacional implementado, com o qual a teoria linguística possa ser expressa? Qual sua eficiência? Como ele pode ser combinado com outros componentes?

A primeira questão em geral é respondida através do uso de um corpus de textos [10]. Assim, é feito um estudo e a posterior seleção de textos representativos do domínio de atuação do gerador. Mesmo nestes textos há que se examinar, cuidadosamente, quais construções da língua são gerais e recorrentes o suficiente para justificar uma investigação mais detalhada. Um fenômeno linguístico interessante mas com baixa frequência de aparecimento pode não justificar todo o esforço dispendido na tentativa de cobri-lo. Em geral, a construção de GCG é um processo longo e iterativo. Começa-se com um núcleo básico cobrindo alguns fenômenos e então, paulatinamente, estende-se este núcleo básico até que se obtenha uma cobertura satisfatória. Este processo não é linear, assim a adição de novos conhecimentos à uma gramática já operacional pode ocasionar erros devido, principalmente, à interação entre as regras.

A segunda questão possui uma resposta mais subjetiva. Pois a questão da escolha das teorias linguísticas subjacentes dependerá, em grande medida, da experiência anterior da equipe de desenvolvimento com alguma delas. Teorias diferentes poderão tratar melhor alguns fenômenos do que outras. Não é raro o caso em que uma GCG precise usar conceitos e métodos de diferentes escolas linguísticas. SURGE [4], por exemplo, possui como eixo principal a teoria linguística sistêmica funcional, entretanto em muitos casos foi preciso usar conceitos da gramática tradicional, assim como conceitos de HPSG (*Head-driven Phrase Structure Grammar*), para realizar muitas das construções hoje possíveis. Outro ponto a se observar é a existência de trabalhos linguísticos que detalhem em nível suficiente como as gramáticas poderiam ser construídas.

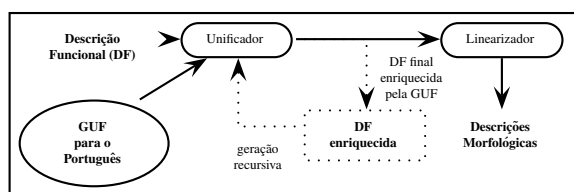


Figura 12: O gerador sentencial

A terceira questão trata sobre o nível de abstração exi-

gido da representação. Vamos examiná-la dentro do processo construtivo das gramáticas de unificação funcional.

De modo geral, uma GUF é a diferença entre a DF final na saída do unificador e aquela dada inicialmente como entrada. Desta forma, tudo o que for desejado na saída e não estiver na entrada deve ser explicitamente fornecido pela gramática. Logo, quanto mais alto for o nível de abstração da DF de entrada, mais complexa será a gramática. Por outro lado, quanto mais sintática e lexicalizada for a DF de entrada mais simples será a GCG. É tarefa do projetista decidir sobre a complexidade da gramática.

A última questão aborda assuntos relacionados com eficiência do formalismo, sua capacidade de integração com outros componentes e as características fornecidas pelo ambiente computacional de desenvolvimento e teste das gramáticas. As GUFs, ou mais genericamente as formalismos gramaticais baseados em restrições, contam hoje com várias ferramentas que diferem de acordo com estes pontos:

- FUF de Elhadad [3], que por ser implementado em LISP conta com todo o aparato disponível há anos para os programadores, além daqueles criados especificamente para acompanhar a ferramenta. Vale notar que SURGE foi criada usando FUF.
- KPML [2] (Komet-Penman Multilingual) é um ambiente de desenvolvimento para a escrita e desenvolvimento de gramáticas. NIGEL, a gramática de geração que o acompanha, foi desenvolvida inteiramente nele.
- Genesys [7], é um ambiente para desenvolvimento de gramáticas, possuindo editores especiais, ferramentas de depuração e interface gráfica.

5.4 Os processadores morfológico e ortográfico

A saída do gerador é uma lista ordenada de descrições morfológicas que, possivelmente, necessitarão de processamento morfológico e ortográfico para se tornarem efetivamente sentenças do português. A ordem indicada nesta lista é a ordem final em que serão posicionados os lexemas após este processamento.

Entre os exemplos de lexemas que precisarão de tratamento morfológico estão:

- Os verbos que precisarão ser flexionados em modo, tempo e pessoa;
- Os substantivos e adjetivos que necessitarão se adequar em número, gênero ou grau;
- Crases, contrações e as formas clíticas;

Terminada a fase do processamento morfológico chega a vez do processamento ortográfico. Algumas das funções do processador ortográfico são:

- Maiusculizar as letras iniciais dos nomes próprios e do início de um período;
- Acrescentar os sinais de pontuação que podem apenas estar indicados na descrição morfológica;
- Colocar marcas de realce tais como aspas, sublinhados, negritos, etc.

6 Problemas e limitações

Construímos um protótipo usando Visual Prolog⁴ para testar a viabilidade da arquitetura proposta e, apesar de algum sucesso inicial, nos deparamos com os seguintes problemas: o módulo de mapeamento de árvore UNL para uma descrição funcional não consegue tratar todos os casos e a cobertura gramatical da GUF é pequena.

A maior dificuldade encontrada no primeiro problema é formular o conjunto de regras que possibilitariam transformar a árvore UNL em uma DF. Este problema, surge em parte, da natureza da própria UNL que não possui um mapeamento bijetivo entre os RLs e as funções sintáticas.

O segundo problema possui suas raízes na inexistência de estudos abrangentes envolvendo a lingüística sistêmica funcional que tem sido aplicada com sucesso no desenvolvimento de outras GCG, como é o caso de SURGE, NIGEL e outras, todas construídas tendo a língua inglesa como base.

A ausência destes estudos fez com que utilizássemos uma adaptação de uma gramática desenvolvida originalmente para o inglês. Esta gramática trata somente de orações simples e sem adjuntos adverbiais e também não lida com construções passivas.

O protótipo possui limitações quando o contrastamos com FUF. Muitas das operações desenvolvidas em FUF para facilitar o desenvolvimento de gramáticas ou para aumentar a eficiência do gerador estão ausentes na nossa implementação. Isto não representa um grande problema para a pequena gramática que usamos, mas certamente será para gramáticas mais realísticas.

Referências

- [1] Douglas E. Appelt. *Planning English Sentences*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, 1985.
- [2] John Bateman. Enabling technology for multilingual natural language generation: The kpml development environment. *Natural Language Engineering*, 3:15–55, 1997.
- [3] Michael Elhadad. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. PhD thesis, Columbia University, 1992.
- [4] Michael Elhadad and Jacques Robin. *Surge: A comprehensive plug-in syntactic realisation component for text generation*. Technical report, Computer Science Department, Ben-Gurion University, Beer Sheva, Israel, 1997.
- [5] Martin Kay. Functional grammar. In *Proceedings of the 5th meeting of the Berkeley Linguistics Society*. Berkeley Linguistics Society, 1979.
- [6] Turgay Korkmaz. *Turkish text generation with systemic-functional grammar*. Master's thesis, Bilkent University, June 1996.
- [7] Tadashi Kumano, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. Genesys: An integrated environment for developing systemic functional grammars. In *Proceedings of International Workshop on Sharable Natural Language Resources*, pages 78–85, Nara, Japan, August 1994. Nara Institute of Science and Technology.
- [8] Ronaldo T. Martins et al. An interlingua aiming at communication on the web: How language-independent can it be? In *Workshop on Applied Interlinguas: Practical Applications of Interlingual Approaches to NLP*, Seattle, Washington, USA, April 2000.
- [9] Kathleen R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, 1985.
- [10] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, New York, 2000.
- [11] Gilles Sérasset and Christian Boitet. Unl-french de-conversion as transfer & generation from an interlingua with possible quality enhancement through offline human interaction. In *MT Summit*, 1999.
- [12] Gilles Sérasset and Christian Boitet. On unl as the future "html of linguistic content" & the reuse of existing nlp components in unl-related applications with the example of a unl-french deconverter. In *COLING*, 2000.
- [13] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. University of Chicago Press, Chicago, 1986.

⁴Visual Prolog é um produto do PDC Group e pode ser encontrado no site www.pdc.dk

- [14] Alessandro Santos Soares. *Gramática de unificação funcional: Levantamento de requisitos para a geração sentencial de português*. Dissertação de mestrado, Instituto de Ciências Matemáticas e de Computação, São Carlos, Setembro 2001.
- [15] C. R. C. Sossolote, C. Zavaglia, Lúcia H. M. Rino, and Maria G. V. Nunes. *As manifestações morfosintáticas da língua UNL no português do Brasil*. Relatório Técnico 2, Instituto de Ciências Matemáticas e Computação de São Carlos (ICMSC) - USP, Novembro 1997.
- [16] Selman M. Temizsoy. *Design and implementation of a system for mapping text meaning representations to f-structures of turkish sentences*. Master's thesis, Bilkent University, August 1997.
- [17] Hiroshi Uchida, Meiying Zhu, and Tarcisio Della Senta. *The UNL, a Gift for a Millennium*. UNU/IAS/UNL Center, Tokyo, Japan., 1999.