ALÉSSIO MIRANDA JÚNIOR

WiKLaTS –UM AMBIENTE DE INTERFACE E INTERAÇÃO PARA MANIPULAÇÃO E FORMALIZAÇÃO DE CONHECIMENTO PARA TRADUÇÃO ENTRE PARES DE LÍNGUAS BASEADA EM REGRAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Profª. Drª. Laura Sánchez García.

CURITIBA

2009

ALÉSSIO MIRANDA JÚNIOR

WiKLaTS –UM AMBIENTE DE INTERFACE E INTERAÇÃO PARA MANIPULAÇÃO E FORMALIZAÇÃO DE CONHECIMENTO PARA TRADUÇÃO ENTRE PARES DE LÍNGUAS BASEADA EM REGRAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Profª. Drª. Laura Sánchez García.

CURITIBA

2009

AGRADECIMENTOS

Começo agradecendo aos meus Avós e aos meus Pais, Aléssio e Marilene: por todo o carinho dedicado a construção de uma família realizada e que julgo feliz; suporte ao meu crescimento pessoal e profissional; serem os meus primeiros mestres; me ensinarem valores morais e cíveis; terem aberto os meus primeiros caminhos; mas principalmente por me darem ensinamentos críticos para fazer minhas escolhas.

Aos meus irmãos Allysson e Carolliny pelo carinho, companheirismo, risadas, respeito ao meu trabalho e por acreditarem que qualquer pessoa pode seguir o sonho que acreditar desde que seja justo e trabalhe com esperança.

À minha grande mestra Laura S. García, agradeço à oportunidade, ao convívio, amizade, preocupação, compreensão, paciência com a minha desorganização, calma com minhas idéias e confiança no meu trabalho.

Não posso deixar em nenhum momento de agradecer à várias amizades e grandes amigos que fiz em Curitiba. Dentre estes, especialmente, aqueles que posso chamar de minha família curitibana, primos, sobrinhos e irmãos de coração que me acolheram de maneira calorosa e fazem de minha estadia nesta cidade algo fantástico.

Ao fim e de maneira não menos importante, agradeço a toda a nação de brasileiros que investiram de maneira indireta em minha educação e formação, atravéz de suas instituições públicas de conhecimento. Espero ter a chance de retribuir a isto, contribuindo para que esta se torne mais justa e próspera.

Muito Obrigado a Todos.

SUMARIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE SIGLAS

APERTIUM – OPENTRAD APERTIUM MACHINE TRANSLATION SYSTEM

IHC – INTERAÇÃO HUMANO COMPUTADOR

MTA – MÁQUINA DE TRADUÇÃO AUTOMÁTICA

UFPR – UNIVERSIDADE FEDERAL DO PARANÁ

WIKLATS – WIKI-KNOWLEDGE FOR LANGUAGE TRANSLATION SYSTEMS

RESUMO

No campo do desenvolvimento de máquinas de tradução automática, é possível ver que os mais bem sucedidos esforços de desenvolvimento são baseados em software com dados proprietários, distribuídos como produtos estáticos e fechados. Neste contexto onde o conhecimento é proprietário e fechado, o Opentrad Apertium se apresenta como uma máquina de tradução automática de código aberto, flexível e robusta que se propões a preencher esta lacuna pela democratização do conhecimento sobre tecnologias de máquinas de tradução. No entanto, acreditamos que um dos principais obstáculos para a popularização do Apertium é a falta de uma interface específica para a manutenção da base de conhecimento lingüístico. A ausência de interfaces que trabalhem esta idéia dificulta o envolvimento de profissionais com a ferramenta, reduzindo o número de colaboradores potenciais a usuários com conhecimentos especialistas no assunto. No presente trabalho propomos uma arquitetura geral de um sistema que visa contribuir para a solução completa e um ambiente em forma de interface; um ambiente colaborativo que poderá capturar e organizar esse conhecimento de maneira razoavelmente simples para um leigo em Computação. O ambiente proposto prevê a capacidade de proporcionar ao sistema Apertium, ou a qualquer outra ferramenta deste tipo, uma crescente base de conhecimento com o objetivo de impulsionar o amadurecimento da tradução entre pares de línguas existentes e o desenvolvimento de novos pares.

ABSTRACT

In a time when the most successful development efforts in Machines Translation (MT) are based on closed software, Apertium has become an alternative mature, interesting and open source. However, one of the main obstacles for the improvement of its results and popularization is the absence of a specific interface to manage its linguistic knowledge, which, because of the imposed difficulty, reduces the number of potential collaborators to the development of the language pairs. We propose a generic architecture of a system that aims to contribute with a complete solution and an interaction-interface environment that can abstract the concepts of the system and the textual process available for the development of bi or monolingual dictionaries. In addition to that, it has the ability to capture and organize knowledge in a simple manner for non-experts in Computing, leading to the growth and development of new language pairs for the MT.

CAPÍTULO 1

INTRODUÇÃO

Neste capítulo teremos uma visão geral do contexto em que o projeto se enquadrou e os problemas que queremos tratar e resolver.

1.1 Visão geral do problema

A Tradução Automática consiste em traduzir automaticamente um texto escrito em uma língua natural (chamada de língua-origem) para outra língua natural (língua-destino), traduzindo todo seu conteúdo de maneira a gerar um resultado inteligível e que preserve suas características, como estilo, coesão e significado.

Definimos neste trabalho um “par de tradução entre línguas”, referido no texto como “par de tradução” ou como sendo o conhecimento necessário para que um tradutor automático transforme um texto de uma língua de origem em uma outra língua destino.

Nos últimos anos, vemos que as mais bem sucedidas Máquinas de Tradução Automática (MTA) são sempre um conjunto de software e bases de conhecimento proprietários, distribuídos como produtos estáticos, fechados e com propósito comercial. Este modelo tem como uma grande desvantagem a dificuldade imposta para estudos de melhoria da arquitetura, técnicas ou até desenvolvimento de novos pares de tradução entre línguas que não têm apelo financeiro significativo.

Esta situação determina uma lacuna para o desenvolvimento de software livre capaz de transformar esta tecnologia em algo aberto e democrático. Acreditamos que criar oportunidades para que a comunidade possa contribuir, cada vez mais, para a evolução e o

desenvolvimento desta área interdisciplinar que envolve a Ciência da Computação e a Lingüística coloca-se como um desafio de relevância clara [1].

Dentre as várias MTAs que surgiram seguindo o modelo de software livre e focando os objetivos descritos, destacamos o "Opentrad Apertium" (Apertium) [2] como um projeto consolidado, que se encontra em constante evolução e com resultados expressivos na literatura**Erro! Fonte de referência não encontrada.**[3]. Ele, o Apertium, é uma MTA baseada em regras sintáticas superficiais, que utiliza uma base de conhecimento própria, com estrutura aberta e flexível no padrão XML.

As MTAs baseadas em regras sintáticas superficiais fazem a análise das sentenças buscando apenas o mínimo necessário para o processo, evitando entrar nos detalhes mais profundos da estrutura. Este tipo de MTA, em geral, é mais eficiente e pode substituir uma análise sintática completa com resultados satisfatórios.

1.2 Objetivos do trabalho

Apesar de o Apertium ter qualidades técnicas reconhecidamente relevantes pela comunidade científica internacional e da base de conhecimento em padrão estruturado e aberto, existem obstáculos para a sua popularização. Atualmente, as poucas alternativas de interface que possibilitam o desenvolvimento e a manutenção das bases de conhecimento, de pares e regras de tradução, ainda visam usuários especialistas em Computação e em Lingüística. Isso se reflete em um obstáculo que dificulta a evolução dos pares disponíveis, uma vez que a quantidade de usuários habilitados a utilizar a ferramenta é bem reduzido.

Um novo usuário, que tenha interesse em colaborar com o desenvolvimento de um novo par de tradução, precisa passar por um período de entendimento das estruturas, ferramentas e processos que, para um leigo em Computação, podem se tornar muito complexas. Devido à

precariedade de formalizações e ferramentas com o intuito de tornar este processo menos especializado, este período de aprendizagem pode se tornar longo, complicado e tende a desmotivar inúmeros colaboradores em potencial. Um dos objetivos deste trabalho é justamente minimizar este fator de desmotivação.

Entre os conceitos e aspectos que devem ser abstraídos ou amenizados pelo ambiente de interface está a marcação de arquivos XML, compilação das bases XML junto ao Apertium, organização, siglas e estrutura interna da base de conhecimento. Com a evolução do conhecimento associado aos pares, a manipulação direta destes arquivos também se torna algo complicado do ponto de vista humano.

A primeira contribuição do presente trabalho é a apresentação uma arquitetura completa, o projeto WiKLaTS – Wiki-Knowledge for Language Translation Systems, que visa gerenciar todo o ciclo de desenvolvimento e manutenção de pares de tradução. Idealizamos um sistema colaborativo, que tenha como perfil de usuário potencial, pessoas leigas em Computação e detentores de conhecimentos mínimos em Lingüística, que possa capturar e organizar de maneira simples o conhecimento sobre línguas e as regras de tradução entre um par de línguas.

A segunda contribuição é o ajuste e formalização de uma parte dos processos teóricos para especificação do conhecimento. Este estudo cria o suporte para a origem de um conjunto de interfaces que objetivam ser uma alternativa mais adequada e eficiente ao processo atual. Estas interfaces são inseridas no contexto de um dos módulos da arquitetura genérica idealizada, marcando o primeiro passo do desenvolvimento do WiKLaTS.

O ambiente de interface proposto tem como principais objetivos

- Tornar o entendimento e o funcionamento do processo mais padronizado e menos dependente de roteiros em forma textual.
- Reduzir ao mínimo os conhecimentos em Computação necessários para que um novo

colaborador entenda como criar ou evoluir o conhecimento associado a um novo par de tradução.

- Manipular as bases de conhecimento de forma confiável, genérica e com funcionalidades mínimas relativa à gestão da mesma.
- Impulsionar o amadurecimento da tradução entre pares de tradução existentes e o desenvolvimento de novos pares, proporcionando às MTAs, como a do projeto Apertium, maior confiabilidade e robustez das bases de conhecimento

1.3 Metodologia

O Desenvolvimento do ambiente proposto envolveu um estudo aprofundado das tecnologias e dos processos utilizados pelos usuários. Este universo se mostrou demasiadamente complexo e amplo para um primeiro estudo no nível de Mestrado. Optou-se então, por uma metodologia cíclica de pesquisa, onde cada ciclo passa por um conjunto de etapas e termina ao apresentar uma solução completa que contemple os objetivos propostos. Isso garante a construção parcial robusta da solução.

Devido ao tamanho do contexto aberto definimos o presente trabalho como um primeiro ciclo de pesquisa. Os objetivos a serem alcaçados foram definidios como: o contato inicial com os desenvolvedores do Apertium, o entendimento do sistema, a identificação dos problemas e o desenvolvimento da solução alternativa para parte deste processo.

Para alcançar os objeitivos almejados listamos as seguintes etapas que compõe este ciclo:

- Interação com a comunidade de desenvolvedores do Apertium visando o entendimento do problema e de sua arquitetura;
- Identificação de problemas e pontos de melhorias relacionados ao Projeto Apertium;
- Criação de uma arquitetura de solução completa para o sistema, o WiKLaTS;
- Especificação dos processos utilizados para especificação do conhecimento na

ferramenta Apertium;

- Projeto de um ambiente de interface-interação como alternativa mais adequada para a especificação do conhecimento necessário para efetuar a tradução ente um par de línguas utilizando conceitos semelhantes ao do Apertium;
- Desenvolvimento de um protótipo não funcional que materialize o projeto criado para avaliação.

Estas etapas foram concluídas e seus resultados são detalhados neste trabalho. Devido à complexidade envolvida, este trabalho não abrangeu conceitos teóricos de Interação Humano Computador (IHC) no desenvolvimento das interfaces do protótipo, uma vez que o nível de dificuldade se concentrou no entendimento do processo existente e da proposta de um novo conceito de interação com o usuário para o problema proposto.

1.4 Organização da Dissertação

O Capítulo 2, Resenha Literária, visa à contextualização do problema, a formalização de conceitos, apresentação do histórico e a revisão de trabalhos relacionados. Ele discorre sobre o que são linguagem natural e máquinas de tradução, tipos de MTA e evolução das mesmas até chegar às MTA atuais revisadas.

O Capítulo 3 detalha como é o processo atual de especificação de conhecimento no Apertium. Nele são citados os conceitos básicos, a arquitetura dos arquivos XML utilizados e um processo desenvolvido empiricamente para especificar o conhecimento propriamente dito. O processo atual foi descrito com base nos conhecimentos disponíveis na ferramenta *Wiki* do Projeto e a partir da utilização de meios de discussão com os desenvolvedores, lista de discussão por *e-mail* e Canal IRC do Projeto. Este processo se apropria de muito conhecimento pré-existente, mas tem uma parcela de contribuição na organização do mesmo. Não existe um processo oficial, mas sim guias para tarefas específicas.

O Capítulo 4 fala sobre a idealização do WiKLaTS, *Wiki-Knowledge for Language Translation Systems*, uma visão geral da sua arquitetura e dos módulos propostos. Trata-se de uma arquitetura complexa não passível de tratamento no presente trabalho, mas que insere o resultado desta dissertação no ângulo de um projeto futuro de maior porte.

O Capítulo 5 descreve interfaces criadas com o intuito de compor o primeiro passo para o desenvolvimento do WiKLaTS. Queremos aqui propor uma alternativa de automação para parte do processo descrito no capítulo 3, visando uma melhor experiência do usuário no desenvolvimento de pares de tradução para MTAs.

O Capítulo 6 resgata os resultados do trabalho e descreve as perspectivas para os trabalhos futuros.

CAPÍTULO 2

RESENHA LITERÁRIA

2.1 Linguagem Natural

Para o ser humano, a comunicação é um processo natural e instintivo que envolve a troca de informação, utilizando como suporte sistemas simbólicos. Este processo envolve inúmeros meios de se comunicar: duas pessoas conversando face-a-face ou por meio de gestos com as mãos, a fala e ou a escrita, permitindo interagir com as outras pessoas e fazer a troca de informação de alguma maneira.

Entendemos como “linguagem” a forma de expressão para se comunicar com outros de uma mesma comunidade ou entre comunidades que compartilhem o conhecimento sobre os códigos que representam ações e objetos, sendo este um meio de adequação do indivíduo à sociedade. Ela é o instrumento de transmissão de idéias e de especificação do conhecimento.

Uma língua é um instrumento vivo, dinâmico, constantemente em desenvolvimento e, juntamente com a cultura, são instrumentos inseparáveis. Diariamente, as línguas sofrem influência das culturas, seja na escrita ou na fala, “(...) dificilmente língua e cultura podem ser separadas. Consideramos que a língua é um dos sistemas de expressão de uma cultura e que diferentes línguas apresentam preferências que são influenciadas pela cultura”[5].

Esta afirmação é claramente observada na língua falada, que não pode ser controlada como a língua escrita. Não é a língua que determina o comportamento de seus falantes, mas exatamente o contrário, esse comportamento é que pode influenciar o uso dela.

Atualmente o processo de globalização tenta aproximar culturas, nações e povos que, por

conseqüência, necessitam adquirir e compartilhar a informação e o conhecimento gerado ao redor do mundo. A diversidade das línguas existentes, juntamente com suas particularidades, torna a intercomunicação entre diferentes comunidades uma tarefa bastante complicada. Neste contexto, existe a necessidade de intermediadores que chamamos de “intérpretes” ou “tradutores” os quais, por sua vez, possuem o conhecimento para fazer a ponte entre as línguas e culturas envolvidas.

Este processo é um desafio para a área da Ciência da Computação e ainda não foi automatizado de maneira satisfatória e sua execução se enquadra em uma categoria de problemas considerados computacionalmente muito complexos [6]. Entretanto, existe interesse em criar facilidades para que o conhecimento sobre quaisquer tipos de língua possa ser relacionado, criando regras que coordenem um processo de tradução automática objetivando aproximações cada vez mais adequadas.

2.2 Máquinas de tradução automática.

O fato de as linguagens serem vivas, extremamente dinâmicas e ambíguas é o determinante do grande desafio de fazer software capaz de efetuar tradução automática de maneira satisfatória. Ainda hoje é extremamente complexo formalizar conceitos, formas de expressão, representações e as relações entre duas línguas quaisquer.

Desde o início do desenvolvimento da Ciência da Computação estudos na área de Lingüística e Máquinas de tradução automática foram e estão sendo desenvolvidos. As razões políticas e econômicas para apoiar a tradução automática entre línguas são claras: o multilingüismo é política e culturalmente interessante, mas a sua realização demanda muito recurso relacionado a tempo e dinheiro para que seja assumida apenas por interesses comerciais. Devido a isso, em vários momentos da história do desenvolvimento da tradução automática foi preciso investir no escopo internacional, como foi feito com o EUROTRA[6],

um projeto não concluído que recebeu financiamento da *European Commission* entre 1970 e 1994 para o desenvolvimento de uma ambiciosa Máquina de Tradução Automática da Comunidade Econômica Européia (CEE).

A concepção popular de tradução automática sempre refletiu a idéia, alimentada a partir de meados dos anos 40, de que deveria ser possível fazer com que um computador gerasse a tradução correta, sem intervenção humana numa língua B, de uma frase na língua A. No caso de ambigüidade, ele deveria poder apresentar duas traduções alternativas [7].

Na década de 40, a tradução automática teria sido a primeira aplicação não numérica proposta dentro da nova área da Ciência da Computação, instigada pela explosão da transmissão de informação e pela relativa facilidade (segundo erroneamente se imaginava) de se calcar o processo computacional em uma técnica humana aparentemente simples[8].

Melby [9] nos anos 50 faz uma comparação entre a corrida espacial e as MTA. Ele dizia que criar MTAs seria mais fácil do que mandar um homem à Lua. Hoje sabemos que o homem já foi várias vezes à lua e a Tradução Automática ainda deixa muito a desejar.

Após anos de pesquisa, nos anos 60, temos uma descrença, uma vez que as aplicações na prática não correspondiam às previsões teóricas e a Lingüística formal não conseguia explicar os problemas ligados a estruturas, processos, funções, formas, entre outros, que se multiplicavam. A Tradução Automática como disciplina do conhecimento começou a ficar desacreditada, culminando em 1966 com a emissão de um importante Relatório da Academia de Ciências Americana (ALPAC) [10]. O relatório foi criticado como sendo tendencioso e limitado, mas seu conteúdo era fortemente negativo com relação às chances de sucesso da tradução automática e provocou um corte radical de verbas governamentais norte-americanas, levando a sobrevivência de apenas três projetos em 1973 e de nenhum em 1975, fato relatado por Slocum[11].

Curiosamente, a única recomendação não seguida foi a que indicava a importância de se

financiarem projetos de pesquisa de longo prazo. Nos Estados Unidos, apenas alguns cientistas isolados, como Peter Toma[6], responsável pelo desenvolvimento do sistema SYSTRAN (www.systranssoft.com), persistiram; na Europa o declínio foi um pouco menos radical.

Hoje existe pesquisa focada em MTA, porém existem dificuldades para um tratamento prático de casos, pois as tecnologias mais avançadas empregadas atualmente são proprietárias e utilizadas em sistemas vendidos comercialmente. São poucas e recentes as alternativas de sucesso que visam criar tecnologias abertas para pesquisa.

Não é fácil para o público leigo compreender os desafios que a tradução humana apresenta, nem entender as dificuldades do desenvolvimento e da avaliação da qualidade da tradução automática. É necessário ter conhecimentos nas áreas da Lingüística e da Informática, e nem isso basta para um planejamento mais complexo. A compreensão das possibilidades e limitações da tradução automática também envolve um entendimento da Filosofia e da Inteligência Artificial. Muito dinheiro já foi gasto sem sucesso na tradução automática devido à falta de uma visão mais abrangente.

A tradução não é um processo simples. Não é a apenas a troca ou a substituição seqüenciais de cada palavra, mas sim a habilidade de conhecer "todas as palavras" em uma sentença ou frase e a maneira como uma pode influenciar na outra, tratando ambigüidade e envolvendo o nível semântico. As línguas humanas podem ser analisadas pela Morfologia (o modo com que as palavras são montadas a partir de pequenas unidades de sentido), pela sintaxe (o conjunto de regras que determinam a estrutura das sentenças corretas), e da semântica (associada ao sentido do discurso). Mesmo textos simples podem estar repletos de ambigüidades.

2.3 Tipos de modelo teórico para a tradução automática

As atuais tecnologias ou modelos teóricos, mais utilizadas no desenvolvimento de MTA podem ser divididas em três categorias: modelos estatísticos, modelos baseados em exemplos e modelos baseados em regras. Na literatura encontramos soluções alternativas, mas estas são sempre modelos híbridos envolvendo estas três.

2.3.1 Máquinas de tradução baseadas em modelo estatístico

Como o próprio nome já cita, estas MTA tentam gerar traduções utilizando métodos estatísticos, atuando sobre corpus de textos de tradução bilíngües já consolidados. Quando este corpus está disponível, e uma vez que os textos sejam de um contexto e estilo semelhante, os resultados alcançados podem ser bastante expressivos. No entanto, encontrar ou gerar um corpus adequado ainda é uma tarefa difícil.

A primeira tradução automática baseado neste modelo foi “Candide”, um projeto de software experimental da IBM desenvolvida no IBM TJ Watson Research Center (http://www.watson.ibm.com/) no início dos anos 90.

No mercado, atualmente o “Google Tradutor” (http://translate.google.com/), antigo “Google SYSTRAN” utilizado por vários anos, foi mudado para um método estatístico de tradução em outubro de 2007. Eles melhoraram as suas capacidades de tradução inserindo cerca de 200 bilhões de palavras a partir de matérias das Nações Unidas para formar o seu sistema[12]. Apesar do grande esforço relacionando corpus de texto, a qualidade da tradução feita por ele é ainda questionada pelos usuários.

2.3.2 Máquinas de tradução baseadas em exemplos

Esta categoria de MTA tem semelhanças ao modelo estatístico, pois juntas compõem a categoria de MTA baseada em corpus de textos. A tradução automática baseada em exemplos ainda está em desenvolvimento e não há nenhum sistema de tradução automática no mercado

que a use, embora alguns sistemas tentem usar a tecnologia da memória de tradução, que é o arquivamento de traduções feitas a fim de tentar reaproveitá-las na tradução de novos textos uma vez que sejam considerados semelhantes.

Antes de processar uma frase por meio de uma análise lingüística, estes sistemas procuram a frase na memória de tradução ("arquivo de tradução"). Apenas no caso de não existir nenhuma frase similar na memória começa a análise lingüística.

Sistemas como o “Langenscheidt T1”, antes chamado de METAL[6], desenvolvido em parceria pela Universidade de Austin juntamente com a Siemens ou o “Personal Translator”, antes LMT desenvolvido pela IBM (http://www.linguatec.de/topics/mt2001.shtml), agora da Linguatec, combinam a abordagem tradicional da tradução automática com o conceito de memória de tradução.

2.3.3 Máquinas de tradução baseadas em regras

Esta é a tecnologia mais tradicional, é o modelo onde se concentram a grande maioria dos tradutores comerciais e livres disponíveis. O processo funciona coordenado por um conjunto de regras e uma base de conhecimentos usados para analisar um texto em várias etapas. Geralmente é gerada uma representação intermediária, simbólica, a partir da qual o texto na língua-alvo é gerado. Neste modelo podemos destacar três tipos de processo de tradução que são representados pela

Figura 2.1.

No primeiro deles o texto original recebe uma abordagem de tradução “palavra-a-palavra” para se obter o texto de destino. Chamamos este processo de “Tradução Direta” e hoje é considerado um processo simples e de resultados pouco relevantes.

Com a evolução do processo anterior, temos o processo “Por Transferência”, que, além da tradução “palavra-a-palavra”, existe uma etapa de análise, o qual é gerado uma representação simbólica do texto. Por meio de regras específicas é possível fazer correspondências entre

representações simbólicas de línguas e com ela gerar novamente um novo texto para concluir a tradução. De acordo com a natureza da representação intermediária gerada, este modelo é descrito como uma abordagem “interlíngua” de tradução automática[6].

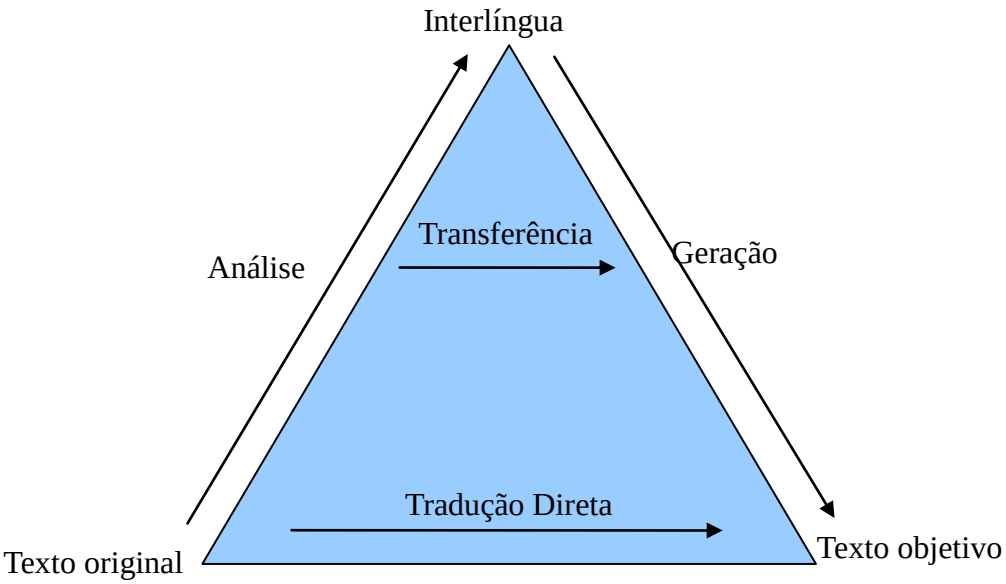

Figura 2.1 Estrutura do processo de tradução

Métodos de tradução por Transferência exigem complexas bases de conhecimento para análise léxica morfológica, sintática e, eventualmente, semântica, além de um conjunto de regras de transferência de grande porte e muito bem organizada.

Na tradução interlíngua, durante a fase de análise, uma língua qualquer poderia ser transcrita em uma língua simbólica, independente de qualquer outra língua, preparada para conter toda a informação do discurso. A partir desta língua simbólica, durante a fase de geração, seria possível executar a tradução para qualquer outra língua. Computacionalmente mais interessante, uma abordagem completamente interlíngua atualmente se encontra desacreditada. Vários estudos já foram conduzidos neste sentido, mas nenhum deles chegou a uma posição suficientemente satisfatória e representativa para possibilitar seu desenvolvimento. O grande desafio é criar uma interlíngua única que consiga representar qualquer outra língua, a diversidade de regras, palavras e sentidos tornam esta tarefa algo, até o momento, utópico.

Embora seguindo o mesmo princípio, no processo de tradução de regras por Transferência, a fase de análise não é executada buscando uma interlíngua, mas sim uma representação computacionalmente mais formal do discurso em questão. Neste momento regras de transferência atuam sobre esta representação, transformando-a em uma mais próxima da língua de destino, de maneira a facilitar a fase de geração. O grande desafio desta abordagem é que é necessário criar regras de tradução diferentes para cada par, porém, é computacionalmente mais adequada.

2.4 Trabalhos relacionados

As mais consagradas máquinas de tradução utilizadas até o momento são todas proprietárias, tanto do ponto de vista do sistema, como da sua base de conhecimento. Estes produtos são distribuídos comercialmente ou acessíveis gratuitamente pela internet com restrições de uso.

Software proprietário, de código e conhecimento fechado, não é facilmente adaptável ou personalizável às necessidades específicas de pesquisas e aplicações. Isto impõe maiores restrições de acesso aos profissionais de tradução e pesquisadores da área, tornando mais difícil o trabalho de contribuir para o desenvolvimento das tecnologias já existentes. A maneira como são distribuídos tem um impactos negativos como: dificuldades práticas para o desenvolvimento de novas técnicas; ampliação do conhecimento formalizado sobre tradução ou línguas; e a criação de novos pares de línguas menos atrativos comercialmente.

O desenvolvimento de software livre pela comunidade na ultima década foi impulsionado pela Internet e o aprimoramento de estratégias de desenvolvimento, aliada à reutilização de código, repositórios unificados e compartilhamento de informação na Rede[13]. Com estes princípios os pesquisadores puderam focar seus esforços em suas respectivas áreas visando o contínuo melhoramento do software já desenvolvido. Este cenário criou uma estrutura e um ambiente maduro para o desenvolvimento de software livre completo de desenvolvimento

continuado e seguro. Isso favoreceu, também, o desenvolvimento de máquinas de tradução alternativas às comerciais.

Citaremos algumas iniciativas software livre de MTA até chegarmos ao projeto Apertium, no qual temos maior interesse.

2.4.1 OpenLogos

O sistema de tradução Logos da "Globalware" é um dos tradutores automáticos comerciais mais antigos, já com 30 anos de desenvolvimento e o "OpenLogos"[14] se tornou um projeto de software livre derivado da liberação do código fonte do tradutor comercial (http://logos-os.dfki.de). Este projeto teve sua ultima atualização em Novembro de 2007 de acordo com o site oficial.

Atualmente o foco do projeto é migrar a implementação do Logos para componentes livres e estão disponíveis, como línguas de origem e de destino, línguas como Português, Francês, Espanhol, Inglês, Alemão e Italiano, por meio da tradução para o Inglês e Alemão que são as línguas chamadas de "referenciais".

Apesar de ser um projeto que herdou um grande legado do Logos ele tem dificuldades pela falta de documentação e seu processo de tradução visa exclusivamente línguas de origem européia. Este foco cria dificuldades para a generalização do projeto para línguas de origens diferentes.

2.4.2 PSMT (Prolog Statistical Machine Translation)

O PSMT é uma máquina de tradução baseada em modelos estatísticos e apesar de ainda ser um projeto experimental é uma alternativa interessante para pesquisadores que desejam trabalhar com máquinas de tradução e linguagem Prolog [4].

No site oficial (http://psmt.sourceforge.net) existe um protótipo *online* onde é possível fazer tradução para o par Ingles-Francês, mas os resultados não são expressivos, uma vez que

o corpus disponível não é suficiente para um bom desempenho.

2.4.3 Moses

O Moses (http://www.statmt.org/moses/) é atualmente a mais ativa iniciativa livre de máquinas de tradução envolvendo modelos estatísticos. Ele prevê meios de treinamento para a tradução entre qualquer par de línguas, sendo apenas necessário proporcionar ao sistema um corpus de tradução adequado[4].

O projeto conta com o suporte de universidades européias e órgãos de financiamento que dão credibilidade à iniciativa. Existem versões estáveis para *download*, demonstrações *online* do resultado de alguns pares. Está em fase de amadurecimento, podendo se tornar uma importante iniciativa no futuro.

2.4.4 Apertium

Dedicaremos propositalmente uma maior atenção ao "Opentrad Apertium" (http://apertium.sf.net), que se define como uma plataforma de máquina de tradução de código aberto, baseada em regras sintáticas superficiais, com capacidade de análise morfológica, sintática e realização de transferências gramaticais por meio de regras de tradução [4].

O projeto é mantido no repositório de software livre SourceForge (www.sf.net), e coordenado em cooperação principalmente por um grupo de pesquisa da Universidade *d`Alacant*, Espanha, e a empresa *Prompsit Language Engineering*, mas com parcerias em várias universidades espalhadas pelo mundo. Inicialmente desenvolvido para a tradução entre pares de línguas de origem comum, atualmente em sua versão 3.0, dá suporte à tradução entre quaisquer pares de línguas que possam ser formalizadas por de regras sintáticas e utilize caracteres UNICODE. Em 2009 este projeto teve mais uma demonstração de representatividade, estando incluído dentro do grupo de 150 projetos de software livre

patrocinados pela entidade *Google no Google Summer Of Code*.

Dentro da plataforma Apertium, máquina de tradução é constituída basicamente por oito módulos independentes e coordenados, que se comunicam por meio de artefatos em formato texto (Figura 2.2). Não entraremos em detalhes sobre as características de cada um dos módulos, mas pode-se dizer que esta estrutura favorece muito o desenvolvimento de aplicações que visam auxiliar o processo. Além de ter seu código fonte disponibilizado, também são abertos tanto seus dados lingüísticos quanto a estrutura de representação, que utiliza o formato XML pensando na interoperabilidade com outros sistemas e na independência do sistema em relação ao conjunto de caracteres utilizados.

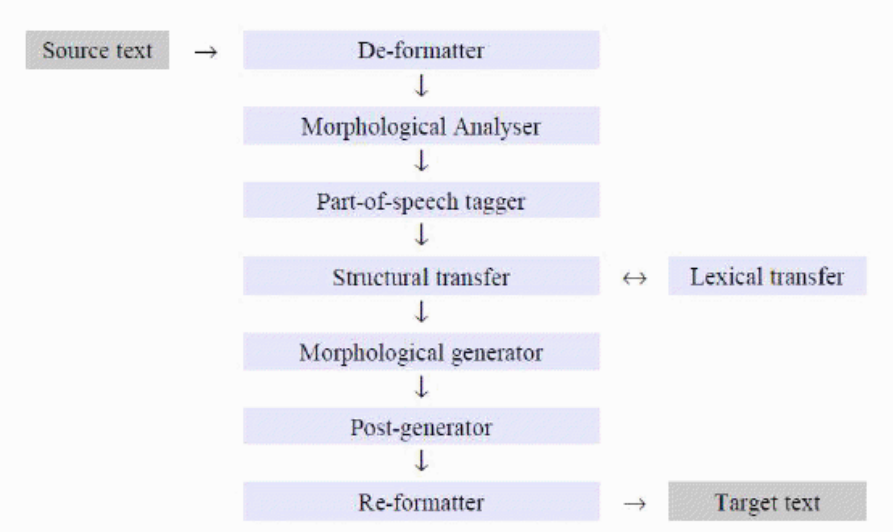

Figura 2.2 Os oito módulos que constituem a MTA Apertium.

Até a última revisão desta dissertação, no site oficial do projeto existiam 18 pares de tradução considerados estáveis, e aproximadamente outros 20 em estágio de desenvolvimento. Cada língua "estável" contempla atualmente uma coleção média de aproximadamente 14.000 lexemas, um número considerado pequeno comparado com o número de itens lexicais da Língua Portuguesa, descritas no dicionário Houaiss com 228 mil lexemas ou Wiktionary (http://www.wiktionary.org/) iniciativa com mais de 55 mil lexemas, mas ainda assim suficiente para efetuar traduções e mostrar seu potencial.

Apesar de ser uma máquina de tradução robusta e em constante evolução, o Apertium

ainda enfrenta problemas para a popularização do seu uso. Dois dos fatores responsáveis por isso são justamente o tamanho limitado por barreiras gerenciais e a qualidade da base de conhecimento associados aos pares de línguas que já podem ser utilizadas em processos de traduçã.

Atualmente a manutenção da base de conhecimento no Apertium é feita diretamente pela manipulação de documentos XML ou por meio de ferramentas não específicas para a tarefa. Isto torna o processo extremamente complexo e difícil à medida que aumentam a sua complexidade e o seu tamanho.

Não devemos o desprezar empenho daqueles que atualmente desenvolvem pares de tradução, mas as ferramentas disponíveis para expandir e manipular as bases de conhecimento são primitivas e insatisfatórias para que leigos em Informática em todo o mundo possam tentar adaptá-las às suas necessidades. Mesmo estes usuários especialistas se deparam com problemas para gerenciar os gigantes arquivos que definem os dicionários bilíngües e monolíngües, entre outros. Atualmente estes alguns destes arquivos ultrapassam 70mil linhas identadas.

O Apertium encontra-se em um estado de desenvolvimento suficientemente maduro, mas falta um esforço no desenvolvimento de ferramentas que possam ampliar e gerenciar a base de conhecimentos específicos de uma maneira mais apropriada.

Propomos o desenvolvimento de um sistema web colaborativo que incorpore os ideais de colaboração de um sistema Wiki [12], e que tenha o enfoque em um ambiente de interface com o usuário, permitindo que qualquer pessoa com conhecimentos mínimos em computação possa moldar e desenvolver a formalização de línguas e de mapeamentos. Este conhecimento será justamente usado para melhorar os resultados de máquinas de tradução baseados em regras, como o Apertium.

CAPÍTULO 3

ESPECIFICAÇÃO DE CONHECIMENTO NO APERTIUM

Atualmente o processo de especificar um novo par de tradução no Apertium é uma tarefa que exige conhecimentos avançados nas áreas de Lingüística, Lingüística Computacional, além de conhecimentos específicos de Computação. Um pesquisador ou usuário que tenha interesse em desenvolver pares de tradução para a MTA Apertium tem disponíveis ferramentas não específicas com recursos de interface-interação primitivos e roteiros em forma puramente textual que detalham partes do processo.

Nosso objetivo neste capítulo é exibir e formalizar uma sequência de passos necessários para a especificação do conhecimento associado à construção de um par de línguas. Um processo que se torna trabalhoso e complexo, principalmente quando se projeta pares de tradução com volume de conhecimento semelhante aos pares atualmente estáveis.

Vamos ilustrar o processo simulando a criação dos artefatos mínimos para o funcionamento de um par de línguas, Português do Brasil (pt) para inglês (en). Queremos traduzir uma frase pequena, mas suficientemente complexa para ser usada como exemplo: “Limpo livros brancos”, cuja saída esperada será “I clean white books”. Não será mostrado o processo completo, mas sim uma compilação dos passos necessários, com vista ao entendimento mais abrangente do processo.

Primeiro temos que introduzir alguns conceitos necessários para a compreensão da modelagem do conhecimento e da estrutura básica dos arquivos a serem criados. Em seguida iremos detalhar o processo de criação dos arquivos de representação do conhecimento.

Não vamos detalhar neste trabalho a estrutura completa dos arquivos a serem criados no padrão XML, seus Dtds de validação que estão descritos nos Anexos A e B. Criar dicionários

e arquivos de regras implica necessariamente entender e conhecer a sintaxe, as etiquetas e a hierarquia de marcação. A fim de diferenciar a nomeclatura vamos nos referir às etiquetas de marcações fixas do XML apertium como “elementos”, o conceito de marcadores ficará relacionado às definições flexíveis da linguagem. No decorrer do texto serão utilizados trechos de exemplos dos arquivos de marcação do Apertium, sendo que à medida que forem sendo necessários, os significados dos elementos serão incluídos a fim de simplificar a compreensão do leitor.

3.1 Conceitos para compreensão da arquitetura do Apertium

O primeiro conceito a ser trabalhado é o de “Lexema”, que é um conceito de Lingüística e consiste em uma unidade léxica (palavra) desconectada de qualquer informação gramatical. Em Português, por exemplo, os lexemas de substantivos são tipicamente representados na sua forma singular no masculino e os de verbos em sua forma infinitiva.

O segundo conceito é o de “Símbolo”, que, dentro do contexto do Apertium, se refere a uma etiqueta de marcação ou classificação, que pode ser gramatical ou de auxílio às etapas da execução da máquina. No caso de classificarmos a palavra “livros” como um substantivo no plural e masculino, seguindo a gramática, ela teria pelo menos as seguintes etiquetas associadas: “substantivo”, “plural” e “masculino". Nas entradas e saídas dos módulos do Apertium estes nomes são tipicamente delimitados por “< >” e representados por abreviações como em:

```
<subs>; para "substantivos".
<pl>; para "plural".
```

Código 3.1 Exemplo de marcadores de Simbolos

O terceiro conceito relevante é “Paradigma”, que se refere a um roteiro de flexões morfológicas para um grupo de palavras. Dentro dos dicionários monolingues, os lexemas são em sua maioria ligados diretamente a paradigmas, para que todas as variações do lexema

sejam descritos sem necessariamente reescrever todas as formas de flexão.

Como forma de exemplificar a idéia, em Português existe um grande grupo de verbos classificados como Verbos Regulares. A principal característica deste grupo é que todos os componentes flexionam da mesma maneira. No contexto do Apertium bastaria criar um único paradigma para representar a flexão destes verbos sem o radical e ligar todos os verbos a este paradigma.

3.2 Arquitetura do conhecimento do Apertium

A arquitetura do Apertium é baseada em uma máquina de tradução genérica que pode ser associada a diferentes bases de conhecimento sobre dicionários e regras de transferência superficial entre línguas. A junção destes componentes efetua a tradução de textos por um processo de transferência superficial, cujo diferencial consiste em não fazer uma completa análise sintática, sendo que as regras são operações sobre unidades léxicas. Um novo par necessita da especificação do conteúdo para pelo menos cinco arquivos, que dividiremos em três categorias, como ilustrado no Quadro 3.1. Estes cinco arquivos são justamente a base de conhecimento básica utilizada pela máquina genérica. Existem outros arquivos que desempenham ajustes mais profissionais à tradução da máquina, mas não serão tratados aqui.

Aquilo que chamamos de Dicionários Monolíngües representam dentro de nosso contexto os dicionários morfológicos de cada língua envolvida. Para fins de melhor descrição do problema usaremos sempre o termo monolíngüe em detrimento ao termo morfológico. Eles contem os lexemas, as flexões e tudo o necessário para identificar e classificar morfologicamente uma frase, expressão e palavras na língua em questão. Podem existir ambigüidades na classificação, mas estas são resolvidas em etapas subseqüentes do processo da tradução que não iremos tratar agora. Como a ferramenta trabalha com pares de tradução, são necessários dois dicionários morfológicos para cada situação.

Os dicionários bilíngües fazem as correspondências entre lexemas e símbolos das duas línguas envolvidas. O seu conteúdo mantém o conhecimento para as duas vias de tradução. Seu principal conteúdo é a relação de correspondência "palavra-a-palavra" entre as línguas em questão.

Tipo	Conteúdo	Exemplo	Nome
Dicionário Monolíngües	Lexemas e regras de flexão destes dentro uma língua.	Português (pt)	"apertium-pt-en.pt.dix"
		Inglês (en)	"apertium-pt-en.en.dix"
Dicionário Bilíngüe	Correspondências entre lexemas e símbolos das duas línguas	Bidirecional Português-Inglês (pt-en)	"apertium-pt_br-en.pt_br-en.dix"
Regras de Transferência	Regras que governam: Como as palavras serão reordenadas nas sentenças. Controle de concordâncias, gênero, número, grau, entre outros. Aonde e como inserir e eliminar itens lexicais.	Português (pt) para Inglês (en)	"apertium-pt_br-en. trules-pt_br-en.xml"
		Inglês (en) para Português (pt)	"apertium-pt_br-en. trules-en-pt_br.xml"

Quadro 3.1 Arquitetura básica dos arquivos de especificação de pares

Os arquivos que especificam as regras de transferência detalham os ajustes, normalmente gramaticais, necessários para que o novo conteúdo na língua destino seja adequado, mesmo que as línguas tenham estruturas bem diferentes entre elas. Como já descrevemos, o Apertium é uma MTA por transferência não baseada em interlíngua completa. São estas regras que fazem as correspondências entre as duas interlínguas parciais, da língua de origem e da língua de destino.

3.3 Dicionário Monolíngüe

A partir de agora trabalharemos com a especificação dos arquivos que se utilizam de

conceitos ligados a linguagem de marcação XML. O dicionário monolíngüe se inicia com as marcações do trecho descrito a seguir. O elemento <dictionary> deve delimitar todo o conteúdo, que vamos separar em quatro partes.

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
      //Conteúdo do dicionário
</dictionary>
```

Código 3.2 Definição base de dicionários

A primeira parte é a que define o alfabeto, conjunto de caracteres válidos para uma língua, sendo delimitado pelo elemento <alphabet>. O dicionário de Português deve ter seu conteúdo semelhante ao Código 3.3. Este conjunto de caracteres faz a validação da leitura dos caracteres do trecho de texto a ser traduzido.

```
<alphabet>
ÀÁÂÃÇÈÉÊÌÍÎÒÓÔÕÙÚÛÜàáâãäçèéêëìíîòóôõùúûüABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrstuvwxyz
</alphabet>
```

Código 3.3 Definição de exemplo para <alphabet>

Na seqüência, como segunda parte, temos a classificação dos símbolos da língua. Se a língua fosse restrita ao nosso exemplo, “Limpo livros brancos”, o elemento <sdefs> delimitaria o conteúdo a seguir. O Quadro 3.2 mostra a relação das etiquetas com as palavras do exemplo.

```
<sdefs>
   <sdef n="subs" c=”substantivo”/>
   <sdef n="adj" c=”adjetivo”/>
   <sdef n="verbreg" c=”verbo regular”/>
   <sdef n="sg" c=”singular”/>
   <sdef n="pl" c=”plural”/>
   <sdef n="m" c=”masculino”/>
   <sdef n="f" c=”feminino”/>
   <sdef n="mf" c=”semgenero”/>
   <sdef n="p1" c=”primeira pessoa”/>
   <sdef n="tpres" c=”presente”/>
   <sdef n="prn" c="Pronome"/>
   <sdef n="pess" c="Pessoal"/>
</sdefs>
```

Código 3.4 Definição de exemplo para <sdefs>

Palavra	Classificação	Descrição
Limpo	verbreg	Verbo regular
	tpres	Tempo Presente
	p1	Primeira Pessoa
	sg	Singular
Livros	subs	substantivo
	pl	plural
	m	masculino
Brancos	adj	adjetivo
	pl	plural
	m	masculino

Quadro 3.2 Palavras e classificações

O próximo passo (terceiro) consiste em definir uma seção de paradigmas delimitada pelo elemento `<pardefs>` e, momentaneamente, se encontra vazio. Iremos preenchê-lo mais adiante no texto.

```
<pardefs>
     <!-- Paradigmas definidos-->
</pardefs>
```

Código 3.5 Definição vazia de <pardefs>

O elemento `<section>` que sucede `<pardefs>` identifica as seções de conteúdos lexicais. Um atributo importante de `<section>` é "*type*", que define o tipo de conteúdo da seção e pode receber dois valores: "*standart*", simbolizando que o conteúdo é tipicamente de lexemas e "incondicional" simbolizando a definição de pontuações e outros símbolos que não trataremos no presente trabalho.

```
<section id="main" type="standard">
    <!-- Secção de lexemas -->
</section>
```

Código 3.6 Definição vazia de <section>

Neste momento obtemos o "esqueleto" de um dicionário monolíngüe que deveria ser semelhante ao Código 3.7:

Agora que temos a estrutura básica e o conteúdo estabelecido para as <alphabet> e <sdefs> vamos adicionar à estrutura um substantivo. O substantivo que iremos adicionar

inicialmente será “livro” e, como não existem paradigmas relacionados à palavra, é necessário definir um.

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>

   <alphabet>
     <!-- Caracteres definidos -->
   </alphabet>
   <sdefs>
     <!-- Símbolos definidos -->
   </sdefs>
   <pardefs>
      <!-- Paradigmas a serem definidos -->
   </pardefs>
   <section id="main" type="standard">
      <!-- Seções de lexemas a serem definidas -->
   </section>
</dictionary>
```

Código 3.7 Esqueleto básicos para definição de dicionários

Para o Português, estamos assumindo que para substantivos o lexema será citado na forma masculina e no singular. No singular a forma do substantivo é “livro” e no plural é “livros”. Então temos o Código 3.8:

```
<pardef n="livro__subs">
   <e>
     <p>
       <l/>
       <r><s n="subs"/><s n="sg"/></r>
     </p>
   </e>
   <e>
     <p>
       <l>s</l>
       <r><s n="subs”/><s n="pl"/></r>
     </p>
   </e>
</pardef>
```

Código 3.8 Definição de paradigma para o lexema “Livro”

Para auxiliar no entendimento das marcações recém utilizadas, o Quadro 3.3 detalha o significado de alguns elementos do formato XML utilizada pelo Apertium. No trecho de Código 3.8 vale notar o atributo “n” do elemento <pardef>, que apenas define um nome de referência para o paradigma. Este nome segue um padrão de nomenclatura definido pelos “pioneiros” do Apertium para facilitar sua identificação.

Elemento	Significado
E	Entrada de um elemento.
P	Par de direções
L	Vem do inglês *Left* que significa a entrada esquerda
R	Vem do inglês *Right* que significa a entrada direita

Quadro 3.3 Linguagem de Marcação

Dentro do par de direções, o conhecimento é construído com a orientação da direita para a esquerda, conceitualmente representando o sentido da direção do processo de geração. Esta definição existe, pois as bases de conhecimento, dicionários e regras, são compiladas para uma máquina de estado finito. No caso de um dicionário, a execução da esquerda para a direita corresponde ao processo de análise de palavras, e no sentido da direita para a esquerda construção das palavras, como mostra o Quadro 3.4.

Entrada	Direção	Saída	Processo
livros	da esquerda para a direita	livros<subs><pl>	Análise
livros<subs><pl>	da direita para a esquerda	livro	Geração

Quadro 3.4 Processos de Análise e Geração de palavras

As definições dos outros paradigmas a serem usados seguem o mesmo princípio que utilizamos com substantivos e iremos defini-los mais rapidamente na sequência.

No caso de adjetivos como “branco” a flexão existe em gênero e número; o paradigma é definido como segue.

```
pardef n="branc/o__adj">
   <e>
     <p>
       <l>o</l>
       <r><s n="adj"/><s n="m"/><s n="sg"/></r>
     </p>
   </e>
   <e>
     <p>
       <l>os</l>
       <r><s n="adj"/><s n="m"/><s n="pl"/></r>
     </p>
   </e>
   <e>
     <p>
```

```
            <l>a</l>
            <r><s n="adj"/><s n="f"/><s n="sg"/></r>
          </p>
        </e>
        <e>
          <p>
            <l>as</l>
            <r><s n="adj"/><s n="f"/><s n="pl"/></r>
          </p>
        </e>
      </pardef>
```

Código 3.9 Definição de paradigma para o lexema "Branco"

A flexão de verbos torna-se bastante extensa devido às flexões combinadas de cada tempo, pessoa e número. Para ilustrar, em nosso exemplo definimos apenas a flexão que será usada.

```
      <pardef n="limp/ar__verbreg">
        <e>
          <p>
            <l>o</l>
            <r>ar<s    n="verbreg"/><s    n="pres"/><s    n="p1"/><s
n="sg"/></r>
          </p>
        </e>
      </pardef>
```

Código 3.10 Definição de paradigma para o lexema "Limpar"

Neste momento os paradigmas que serão usados já foram definidos e podemos ligá-los aos lexemas correspondentes. Colocaremos esta "ligação" de referências dentro o elemento <section> que já foi definido anteriormente.

Dentro do elemento em questão, a definição do lexema "livro" será feita como o Código 3.11 e os significados dos novos elementos da estrutura são descritos no Quadro 3.5.

```
      <e lm="livro">
            <i> livro </i>
            <par n="livro__subs"/>
      </e>
```

Código 3.11 Definição do lexema Livro

Com o trecho de Código 3.11 definimos o lexema "livro" ligado ao paradigma que o flexiona, "livro_subs". Para os lexemas "branco" e "limpar" a definição é feita no trecho de Código 3.12 e o arquivo completo do nosso pequeno Dicionário Monolíngüe de Português pode ser visto no Apêndice A.

```
<e lm="branco">
      <i>branc</i>
      <par n="branc/o__adj"/>
</e>
<e lm="limpar">
      <i>limp</i>
      <par n="limp/ar__verbreg"/>
</e>
```

Código 3.12 Definição de Lexemas

Elemento	Significado
LM	Lexema
I	Identidade, ambos os sentidos, direita e esquerda, têm o mesmo identificador
PAR	Paradigma

Quadro 3.5 Linguagem de Marcação

Para o Dicionário Monolíngüe de Inglês será feita uma rotina similar à descrita para o Português e o resultado se encontra no Apêndice B. Vamos apenas descrever a definição do pronome pessoal reto do Inglês "I", pois este tem uma estrutura ligeiramente diferente daquela do exemplo exposto.

Temos que este tipo de pronome não tem flexão, logo não tem raiz de conjugação. Será criado para ele o seguinte paradigma.

```
<pardef n="prret__prn">
   <e>
    <p>
      <l>I</l>
      <r>pronpes<s n="prn"/><s n="pess"/><s n="p1"/><s n="sg"/>
      </r>
    </p>
   </e>
</pardef>
```

Código 3.13 Definição de paradigma para o Lexema "I"

Dentro do elemento <section> ficará a seguinte entrada.

```
<e lm="Pronome Pessoal Reto">
    <i/>
    <par n="prret__prn"/>
</e>
```

Código 3.14 Definição do Lexema "I"

Devemos notar que este pronome em nosso exemplo não tem seu corresponde no dicionário da Língua Portuguesa, pois ele será expandido na conjugação do verbo usado, e sua

resolução será apresentada como um exemplo de uso de regras de tradução.

3.4 Dicionário Bilíngüe

Agora que temos os dois dicionários monolíngües definidos, precisamos definir o dicionário bilíngüe. Sua estrutura é igual à dos dicionários monolíngües, usará o mesmo "esqueleto básico" do definido pelo Código 3.7, mas seu objetivo é registrar o mapeamento entre lexemas de dois dicionários monolíngües diferentes. Seguindo a nomeclatura do Apertium, nosso novo arquivo terá o nome apertium-pt-en.en-pt.dix.

Os elementos <sdef> deverão delimitar o conteúdo com todos os símbolos dos dois dicionários monolíngües envolvidos. Em nosso caso temos:

```
<sdefs>
   <sdef n="subs" c="substantivo"/>
   <sdef n="adj" c="adjetivo"/>
   <sdef n="verbreg" c="verbo regular"/>
   <sdef n="sg" c="singular"/>
   <sdef n="pl" c="plural"/>
   <sdef n="m" c="masculino"/>
   <sdef n="f" c="feminino"/>
   <sdef n="mf" c="semgenero"/>
   <sdef n="p1" c="primeira pessoa"/>
   <sdef n="tpres" c="presente"/>
   <sdef n="prn" c="Pronome"/>
   <sdef n="pess" c="Pessoal"/>
</sdefs>
```

Código 3.15 Definição de símbolos em Dicionário Bilingue

O elemento <section> contem as relações entre os lexemas, e sua função é gerenciar a troca de padrões. Estes padrões fazem o papel do tradutor "palavra-a-palavra", porém com uma análise mais fina de resultado. Em momentos de ambigüidade como, por exemplo, na definição de gêneros indefinidos, são usados recursos de marcadores que são resolvidos por outras etapas de tradução não tratadas neste trabalho.

Segue a codificação do mapeamento "palavra-a-palavra" do dicionário bilíngüe necessário para nossa frase de exemplo.

```
<e><p>
      <l>limpar<s n="verbreg"/></l>
      <r>clean<s n="verbreg"/></r>
</p></e>
<e><p>
      <l>telcado<s n="subs"/></l>
      <r>keyboard<s n="subs"/></r>
</p></e>
<e><p>
      <l>branco<s n="adj"/></l>
      <r>white<s n="adj"/></r>
</p></e>
```

Código 3.16 Mapeamento de palavras em Diconário Bilingue

3.5 Regras de transferência

Agora que temos todos os dicionários, precisamos ainda dos arquivos com as regras de transferência, deve existir um arquivo de regras para cada sentido da tradução. A seguir vamos exemplificar a criação de uma regra de transferência no sentido de português para inglês. Este exemplo mostra um mínimo dos recursos disponíveis e que podem ser utilizados. Este arquivo é dividido em quatro partes, e seu "esqueleto" básico é dado por:

```
<?xml version="1.0" encoding="UTF-8"?>
<transfer>
   <section-def-cats>
      <!-- Definição de Categorias -->
   </section-def-cats>
   <section-def-attrs>
      <!-- Definição de Atributos -->
   </section-def-attrs>
   <section-def-vars>
      <!-- Definição de Variáveis -->
   </section-def-vars>
   <section-rules>
      <!-- Definição de Regras -->
   </section-def-rules>
</transfer>
```

Código 3.17 Esqueleto básico do arquivo de regras de tranferência

Iremos construir a regra responsável por explicitar o pronome pessoal implícito na forma verbal conjugada "Limpo" em Português.

Primeiro precisamos definir categorias e atributos que nos permitirão agrupar os símbolos

gramaticais. Categorias nos permitem agrupar símbolos com casamento de padrões, como em "subs.*", que é o padrão para todos os substantivos. Atributos são um conjunto de símbolos que podem ser escolhidos dentro de uma determinada opção. Os símbolos "singular" (sg) e "plural" (pl) podem ser agrupados no atributo "numero" como o Código 3.18

```
<def-cat n="pronpes">
   <cat-item lemma="pronpes" tags="prn.*"/>
</def-cat>
<def-cat n="vrb">
   <cat-item tags="verbreg.*"/>
</def-cat>
<def-attr n="tipo_prn">
   <attr-item tags="prn.subj"/>
   <attr-item tags="prn.obj"/>
</def-attr>
<def-attr n="numero">
   <attr-item tags="sg"/>
   <attr-item tags="pl"/>
</def-attr>
<def-attr n="tempo">
   <attr-item tags="pri"/>
</def-attr>
<def-attr n="pessoa">
   <attr-item tags="p1"/>
</def-attr>
```

Código 3.18 Definição de Categorias e Atributos

Em seguida temos que definir uma regra que, utilizando os atributos e categorias criados na seção anterior, fazem modificações na saída, ajustando estruturas gramaticais e fazendo uma análise diferenciada de ambigüidades. As regras são exceções que ajustam o resultado e resolvem ambigüidades catalogadas para que, utilizando o dicionário monolíngüe da língua de destino, a saída seja pelo menos mais adequada do que um resultado "palavra-a-palavra". A definição de boas regras é uma parte importante no auxílio à tradução.

A seguir listamos a regra aplicada a lexemas do tipo "verbo" (vrb). Em nosso exemplo ilustrativo, esta regra, sempre que encontrar um verbo em Português irá reordenar e incluir elementos na saída da interlíngua parcial. Em nosso exemplo, ela cria um novo lexema "pronpes" auxiliado por atributos de pessoa e número que estavam presentes na flexão do verbo em Português.

```
<rule>
   <pattern>
      <pattern-item n="vrb"/>
   </pattern>
   <action>
      <out>
         <lu>
            <lit v="pronpes"/>
            <lit-tag v="prn"/>
            <lit-tag v="pess"/>
            <clip pos=1 side=tl part="pessoa"/>
            <clip pos=1 side=tl part="numero"/>
         </lu>
         <b/>
         <lu>
            <clip pos=1 side=tl part="lem"/>
            <clip pos=1 side=tl part="a_verb"/>
            <clip pos=1 side=tl part="temps"/>
         </lu>
      </out>
   </action>
</rule>
```

Código 3.19 Definição de regra de exemplo

3.6 Considerações finais

O processo descrito apresenta uma pequena parcela dos desafios da tarefa de especificar um novo par de línguas no Apertium. É importante reforçar que todo o processo é baseado em arquivos textuais e que referências entre etiquetas de marcação, por exemplo, devem ser mantidas corretas de maneira manual.

No caso de dicionários e arquivos de tamanho reduzido é possível manter e gerenciar de maneira razoável o conceito associado às línguas, mas atualmente, em se tratando de dicionários monolíngües com mais de 10 mil lexemas e mais 80 mil linhas, gerenciar todo este conhecimento se torna inviável ou limitado sem ferramentas de auxílio.

O processo é mais complexo do que mostrado, compreendendo outras fases, arquivos e estruturas, mas focamos nosso trabalho em resolver o desenvolvimento dos passos iniciais da especificação de um par de línguas. Com estes cinco arquivos já é possível efetuar os primeiros passos na tradução de sentenças.

CAPÍTULO 4

WiKLaTS – Wiki-Knowledge for Language Translation Systems.

Tentamos mostrar até aqui um panorama sobre as MTAs e o processo de criação de línguas com o Apertium. Neste contexto, e com oportunidades já identificadas, vamos propor um projeto mais amplo que visa, no futuro, gerenciar por completo as bases de conhecimento que alimentarão o Apertium e outras máquinas de tradução automática.

Este é o projeto WiKLaTS, que apresentaremos de maneira panorâmica e é o grande motivador deste trabalho. Sua concepção é uma contribuição e faz parte do estudo. No entanto, devido à complexidade envolvida, colocaremos aqui apenas uma visão da arquitetura proposta, os objetivos gerais e alguns dos seus requisitos.

4.1 O projeto

No capítulo anterior foram mostrados alguns detalhes do processo de criação de pares de língua no do contexto do Apertium. Este processo, como disponível atualmente, torna ainda mais complexo o entendimento da estrutura dos arquivos ou mesmo para a sua manipulação. Isto determina uma carência de ferramentas que auxiliem o usuário no processo de criação e de desenvolvimento de pares de línguas. A necessidade já é notada mesmo para pares simples, mas é intensificada com a evolução de pares a um nível de complexidade alto quando pensamos em pares de tradução estáveis e com resultados mais próximos dos tradutores atuais.

O projeto WiKLaTS se propõe a preencher esta lacuna, devendo ser um ambiente integrado e interativo, voltado para usuários leigos em Informática e com conhecimentos

mínimos em Lingüística para desenvolvimento e gerência de pares de línguas. Ele visa possuir todos os recursos necessários à criação e à manipulação dos repositórios de conhecimento sobre línguas e regras de transferência sintáticas que tornem possível a tradução entre pares de línguas a partir de uma MTA baseada em regras.

Vislumbramos um ambiente com interfaces usuário que guiem o desenvolvimento de pares de línguas, sendo que o usuário deixará de necessitar de conhecimentos ligados à Informática, como linguagem de marcação XML, além de procurar diminuir a dependência de conhecimentos em Linguistica. Isto aumentará a base de usuários em potencial, com a possibilidade de que pessoas leigas em Computação possam contribuir com as bases de conhecimento, além de diminuir a curva de aprendizado sobre o processo.

Pensamos no conceito de Wiki como uma alternativa de sucesso no campo de construção colaborativa de conhecimento. É importante citar que o projeto espelhe-se nas características deste conceito e não é uma variação do software atual.

Almeja-se que sua base de conhecimento e funcionamento seja independente de qualquer máquina de tradução, sendo aberta e passível de integração a qualquer software que respeite suas interfaces de comunicação. Apesar da expectativa de independência da máquina de tradução, é claro que as estruturas e sua organização e o seu processo são intimamente ligados aqueles do Apertium, ferramenta que motivou o presente trabalho.

A representação de conhecimento e os processos documentados são concebidos com base no uso empírico da ferramenta, mais especificamente por meio da criação de pares de línguas experimentais e de troca de mensagens com grupos de lingüística, profissionais e pesquisadores de computação e lingüística que desenvolvem para o Apertium alem da documentação oficial disponibilizado pela plataforma.

Com base no conhecimento construído, esperam-se desenvolver um conjunto de técnicas, abstrações e fluxos que tornem a criação e a evolução destes tipos de conhecimento, tarefas

mais estruturadas, padronizadas e fáceis.

O WiKLaTS deve contemplar os seguintes requisitos:

- Prover um processo unificado e padronizado para a especificação do conhecimento sobre pares de línguas;
- Proporcionar ferramentas para auxílio à ampliação do conhecimento associados aos pares existentes;
- Permitir a exportação do conteúdo gerado para utilização em MTA (Apertium).
- Unificar os Dicionários Monolíngües de pares de línguas relacionadas entre si. Atualmente é comum que sejam especificadas várias versões do Dicionário Monolíngüe de uma língua, uma para cada par de línguas existente.
- Tornar possível a padronização dos arquivos XML fontes que representam o conhecimento;
- Proporcionar ferramentas de classificação sobre a qualidade da tradução entre os pares de línguas;
- Prover ambiente de testes de tradução baseado nas alterações feitas.
- Permitir controle de versões sobre os pares de língua desenvolvidos;
- Permitir o controle de alterações baseado em autoria e permissões de acesso;
- Permitir que remotamente usuários cadastrados possam dar contribuições.

4.2 A Arquitetura do WiKLaTS

Existem duas arquiteturas principais que se comunicam, o Projeto Apertium e o Projeto WiKLaTS, sendo que podemos dividir estas arquiteturas em módulos menores com funções bem definidas. Primeiramente vamos quebrar estas arquiteturas em blocos menores para uma melhor análise da proposta (Figura 4.1).

O Apertium possui duas bases de conhecimento: uma aberta, pois é definida por arquivos

XML e é aberta à edição do usuário, e a segunda binária, pois é a compilação da base aberta XML na máquina de estados finita que é utilizada pela MTA. Existe um pacote de utilitários chamado “lttoolbox” que fazem este processo de compilação e ajuste da base aberta para a base binária. Estes três pacotes, juntamente com a máquina de tradução genérica já detalhada na Figura 2.2, são os necessários para o funcionamento do WiKLaTS independente do Apertium.

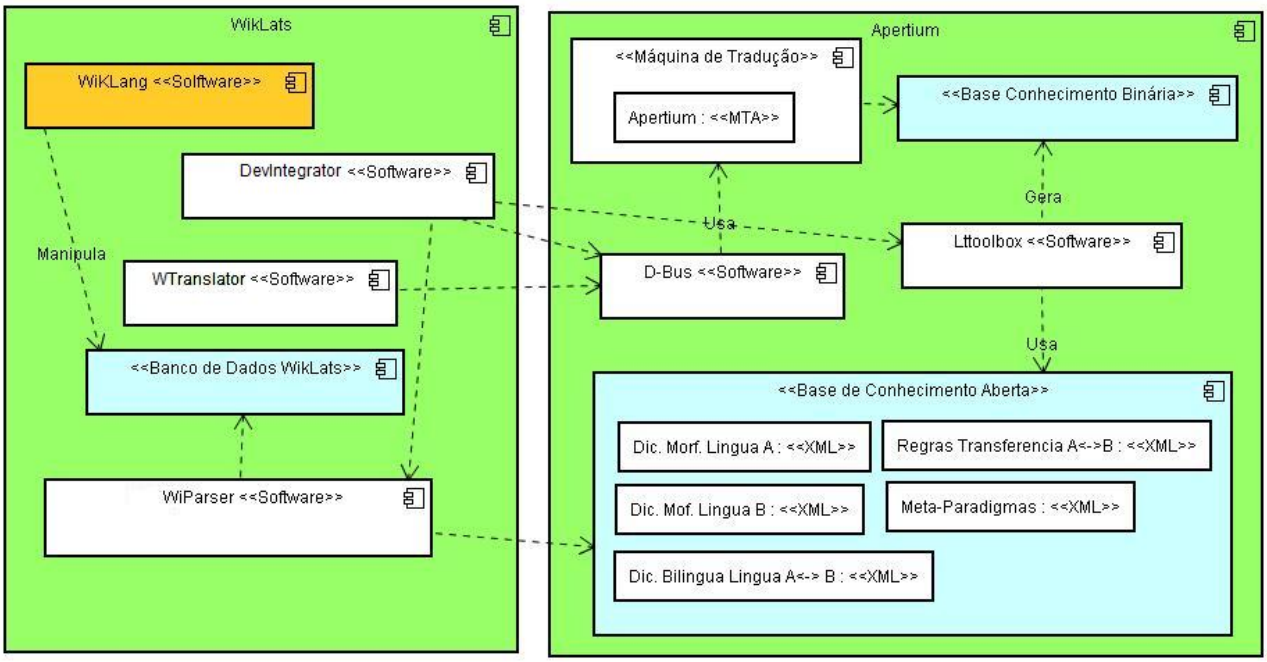

Figura 4.1 Arquitetura WiKLaTS

O módulo chamado “D-Bus Services” é a primeira iniciativa que possibilita chamadas e comunicação do Apertium com outros programas. Existem alguns projetos ativos abordando o desenvolvimento de módulos de comunicação baseados em WebServices e que podem ser muito importantes no futuro, mas ainda não é uma realidade.

Uma descrição mais detalhada desta arquitetura pode ser encontrada no *site* oficial do projeto Apertium (http://www.apertium.org).

O WiKLaTS é constituído de quatro módulos utilitários e uma base de dados própria que prima pelo controle do conhecimento que dará origem aos arquivos que serão usados pela MTA. Os módulos “WTranslator”, “WiParser”, “DevIntegrator” e “WiKLang” estão

ilustrados na Figura 4.1

4.2.1 WTranslator – Módulo de integração com máquina de tradução

O WTranslator será o módulo responsável pela comunicação entre a MTA e o usuário final que quer usar o serviço de tradução. Deverá ser uma interface-usuário que permita para testar a tradução, utilizando uma ou várias versões de um conjunto de regras de tradução. Num momento de colaboração web para o desenvolvimento de pares, pensamos que um mesmo par de línguas pode ter várias versões ou autores. Esta mesma interface deve ser capaz de possibilitar ao usuário testar, classificar, distinguir e sugerir quais as melhores opções para tradução. Os seguintes requisitos são desejáveis para o WTranslator:

- Tornar possível a comparação entre diferentes conjuntos de regras.
- Verificar quais os pares de tradução escolhidos ou classificados como de melhor qualidade de resultado.
- Tornar possível ao usuário classificar o resultado obtido.
- Em caso de traduções não satisfatórias, permitir ao usuário sugerir a tradução correta. Esta sugestão poderá ser salva como pendência a ser verificada pela equipe responsável pelo conhecimento sobre o par de línguas em questão.

No contexto do Apertium a comunicação com a MTA se dá com o uso da interface de comunicação “D-Bus”. Atualmente existem iniciativas para transformar os serviços do Apertium em WebServices mas elas ainda não foram concluídas.

4.2.2 WiParser – Módulo de Comunicação entre Bases de conhecimento

O WiParser será o módulo responsável por importar e exportar as bases de conhecimento entre as máquinas de tradução e o projeto WiKLaTS. O intercâmbio entre as bases de conhecimento é um dos gargalos para tornar o projeto flexível e atrativo a qualquer outra

máquina de tradução.

As estruturas de dados internas a serem desenvolvidas para o WiKLaTS serão baseadas nas pesquisas e na experiência do projeto Apertium. No entanto, elas podem não ser equivalentes, pois como resultado de pesquisas futuras o portal pode conter conceitos que não sejam tratadas pelo projeto Apertium.

Acreditamos que a independência pode tornar o desenvolvimento de ambas as arquiteturas mais robusta, sendo desejável criar uma abertura para que se torne possível o intercambio com outras máquina de tradução ou ferramenta além do Apertium. Não descartamos a idéia de que outros projetos possam contribuir criando integrações com projetos existentes. Daí a necessidade de proporcionar uma interface de comunicação que permita contribuições.

4.2.3 DevIntegrator - Modulo de testes e aprimoramento

Este ambiente tem por objetivos incentivar e facilitar um ciclo interativo de melhoramento da qualidade do resultado final do processo de tradução.

Um dos caminhos pesquisados consiste em quebrar e explicar o processo de tradução passo-a-passo, de maneira a auxiliar o usuário mais avançado e identificar em qual etapa do processo houve falha ou ambigüidade. O processo de tradução tem várias etapas e é mais fácil corrigir uma tradução errônea sabendo-se em qual etapa aconteceu o erro.

É importante criar abstrações que encapsulem a complexidade do processo. Outra importante funcionalidade seria a possibilidade de intervir nas etapas de modo a criar alternativas que melhorem os resultados. Esta intervenção temporária mostraria mais claramente de que forma uma determinada solução iria influenciar no resultado.

Computacionalmente este seria o módulo mais complexo, uma vez que deve ter comunicação sincronizada com vários outros módulos. Seguem alguns requisitos do DevIntegrator:

- Permitir a visualização das etapas do processo passo-a-passo;

- Possibilitar e controlar alterações temporárias nas bases de conhecimento;
- Permitir ao usuário efetivar ou sugerir alterações que representem uma melhora no mecanismo;
- Permitir o teste online das alterações refletidas na MTA;
- Proporcionar interação guiada que auxilie o processo de construção da tradução;
- Ter capacidade de comunicação com os outros módulos envolvidos, por ser um módulo de integração do processo.

O projeto Apertium dá suporte à concretização destas idéias, uma vez que cada módulo gera artefatos parciais em formato texto que marcam o passo do processo. O ambiente de interação-usuário deve eximir o usuário da necessidade de dominar conceitos lingüísticos, de maneira que o usuário final possa compreender tudo o que acontece dentro da máquina de tradução.

Atualmente, para o Apertium, existe uma ferramenta chamada "ApertiumView" (http://javabog.dk:8080/apertium-viewer/launch.jnlp) que implementa rusticamente uma pequena parte das funcionalidades propostas para este módulo. Seu objetivo é a exibição em forma de texto não tratado as saídas parciais do processo de tradução, auxiliando a detecção de falhas no processo. A Figura 4.2 mostra a interface disponível no ApertiumView.

4.2.4 WiKLang - Modulo de manipulação da base de conhecimento

Este módulo, foco da presente dissertação, terá uma atenção especial. No próximo capítulo, iremos detalhar as interfaces desenvolvidas com o propósito de implementar estes requisitos e atender as necessidades reais dos usuários alvo, que são usuários com desejo de contribuir com o Apertium mas que não .

A principal função deste módulo consiste em intermediar o processo de gestão do conteúdo sobre línguas e regras de tradução para o desenvolvimento de pares. O ambiente deve proporcionar, a um usuário leigo em Informática e sem conhecimentos profundos de

Lingüística, a possibilidade de criar e manipular os conceitos e estruturas necessários ao processo de tradução entre pares de línguas.

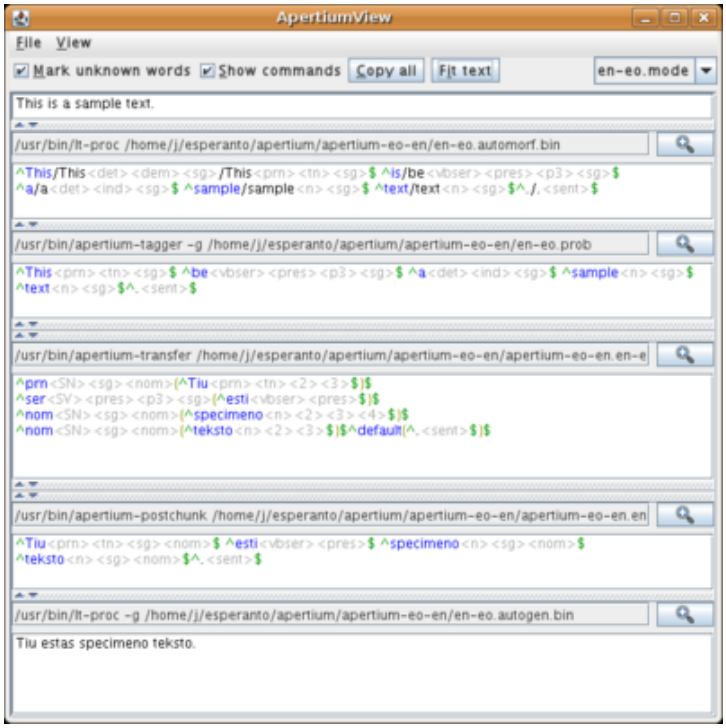

Figura 4.2 Apertium View

Ele se calca em simplificar o processo descrito no Capítulo 3, reduzindo a complexidade relacionada com conceitos computacionais de estrutura e processo manual. Com o conhecimento prévio do capítulo mencionado, foi possível quebrar a tarefa de criar uma base de conhecimento, inicialmente, em três tarefas fundamentais, cada uma das quais representa um grupo de arquivos que é base de conhecimento do Apertium: definição de dicionários monolíngües, definição de dicionários bilíngüe e definição de regras de tradução.

CAPÍTULO 5

O AMBIENTE DE ESPECIFICAÇÃO DE CONHECIMENTO.

No capítulo 3 descrevemos algumas características do processo de especificação do conhecimento para o Apertium com o intuito de mostrar sua complexidade.

Neste capítulo iremos descrever uma proposta de interface usuário que visa substituir parte do processo descrito no Capítulo 3. Tentamos aqui fazer uma alternativa ao processo atual do Apertium necessário à criação de arquivos nas etapas que representam a criação de dicionários, bilíngües e monolíngües.

Para o WiKLaTS, idealiza-se que o processo seja desempenhado em uma base de dados própria e que seu resultado seja apenas exportado para os arquivos do Apertium. No protótipo a atenção foi dirigida à edição direta dos arquivos XMLs, deixando a separação das bases para um esforço futuro. Esta escolha é colocada de maneira a tornar seu desenvolvimento mais próximo dos usuários.

5.1 Premissas

No processo descrito mostramos a criação de dicionários monolíngües, bilíngües e uma introdução às regras de tradução. No ambiente proposto vamos ignorar a criação de regras de tradução, focando o desenvolvimento de dicionários. A relevância deste recorte já se demonstra adequada devido ao foco das dificuldades com que os desenvolvedores de pares se deparam para desenvolver e manter detalhados dicionários nas bases XML que tendem a ser cada vez mais volumosos com o desenvolvimento do par.

É uma ambição que as interfaces propostas sejam suficientemente genéricas para criação

de quaisquer dicionários, bilíngües ou monolíngües. Entretanto, existem muitos problemas particulares que não foram expostos durante o desenvolvimento que exigem que a proposta do presente trabalho seja estendida em trabalhos futuros.

Esta proposta foi desenvolvida sempre utilizando da estratégia de envios dos protótipos para um grupo de usuários em potencial e dos respectivos retornos, sendo um processo iterativo de avaliação e aperfeiçoamento dentro de cada versão. Ressaltamos novamente que este trabalho não abrangeu diretamente conceitos teóricos de IHC no desenvolvimento das interfaces, o nível de dificuldade se concentrou no entendimento do processo existente e da proposta de um novo conceito de interação com o usuário para o problema proposto.

5.2 O Ambiente

Analisando a interface proposta pela Figura 5.1, destacamos cinco elementos que compõem o ambiente.

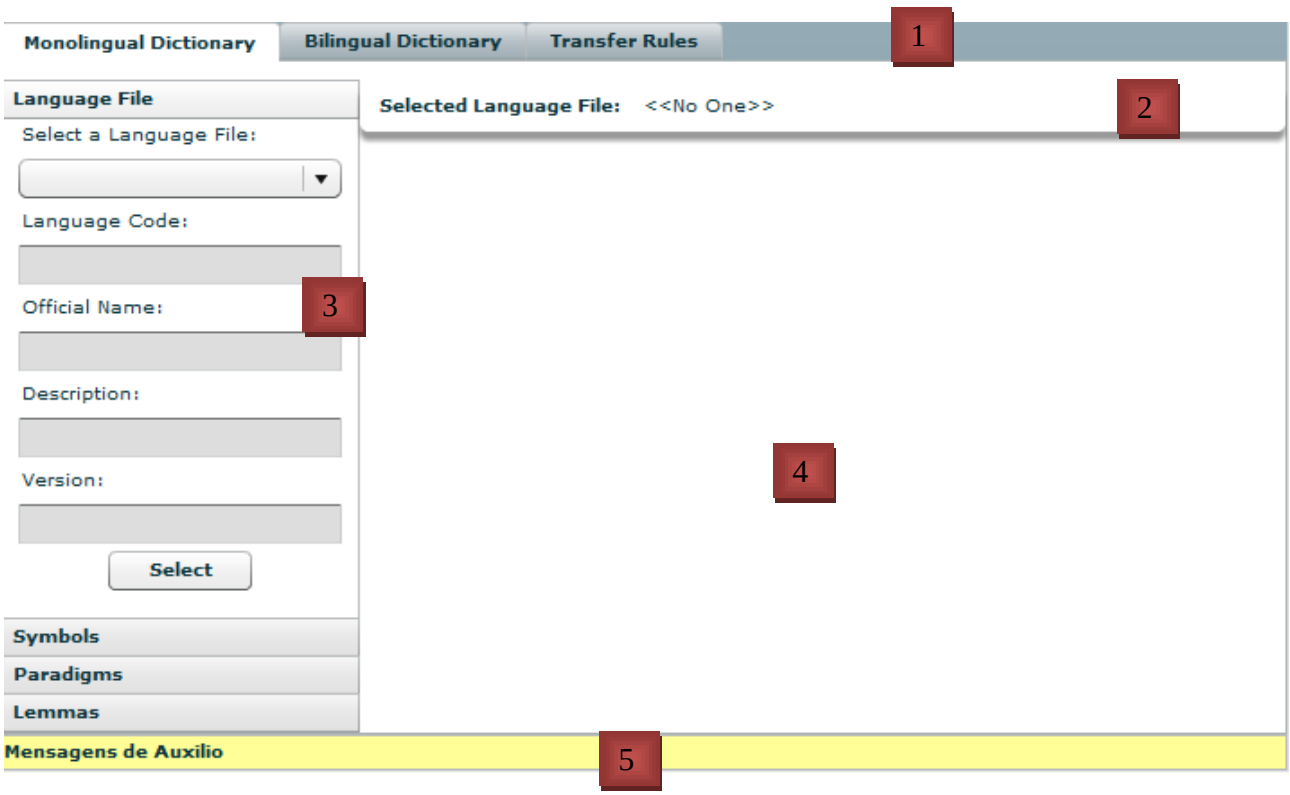

Figura 5.1 O ambiente proposto

1. Barra de páginas que separam o processo em três etapas distintas a serem descritas.
    a. Dicionário Monolíngüe
    b. Dicionário Bilíngüe
    c. Regras de Transferência. (Não contempladas no presente trabalho)
2. Barra de contexto, com as informações que indicam ao usuário o contexto do processo em que ele se encontra.
3. Menu flutuante contextualizado com os subprocessos de cada etapa.
4. Painel de ação, onde são executadas as ações de orientação do sistema e de entradas do usuário para cada subprocesso.
5. Barra de informação ao usuário, contendo conceitos, auxílio e sugestões de procedimento.

5.3 Dicionário Monolíngüe

O Dicionário Monolíngüe se tornou uma tarefa delicada no desenvolvimento de pares de línguas, principalmente devido à complexidade da estrutura empregada no Apertium e o volume de informação a ser representado. Dividimos o seu desenvolvimento em quatro etapas, que remetem justamente à distribuição estrutural dos XMLs resultantes.

5.3.1 Definições do Dicionário Monolíngüe

O primeiro subprocesso da definição de um Dicionário Monolíngüe foi definido como uma área de exibição de estatísticas e atributos estáticos da língua. Ao selecionar um dicionário, ou mesmo ao criar um novo, algumas características são importantes.

Atributos como código universal da língua, descrição e versão são apenas informativos, mas a definição do alfabeto influencia o conjunto de caracteres válidos nas interfaces associadas. O ambiente apresenta algumas estatísticas básicas que podem auxiliar no controle do desenvolvimento do conhecimento sobre a língua.

É necessária a execução desta etapa para permitir os novos passos. A área de contexto exibirá em todo momento qual a língua que está sendo alvo construção. A tela que representa o início desta etapa é representada pela Figura 5.2.

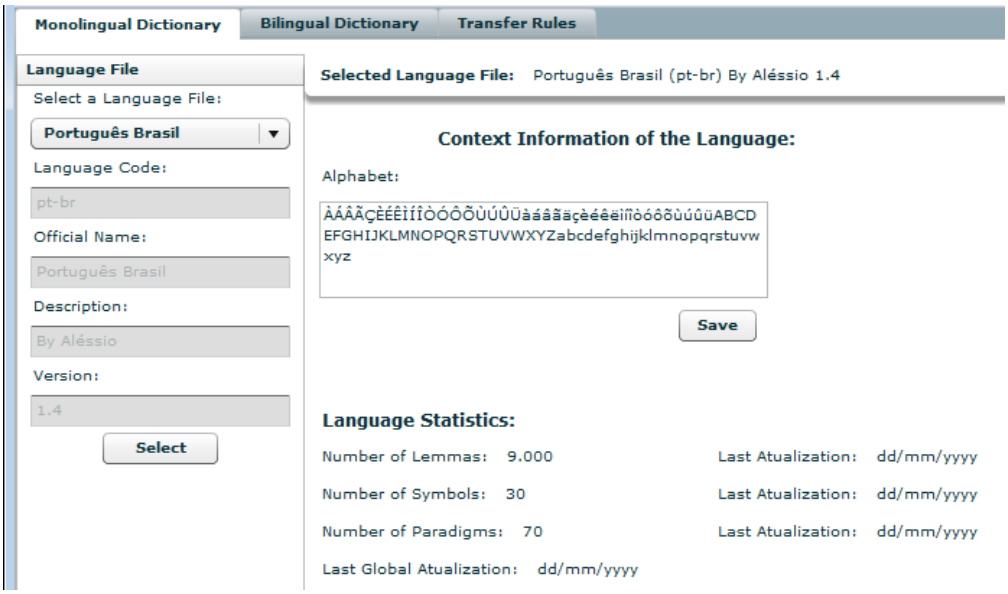

Figura 5.2 Estatísticas e atributos contidos na interação do dicionário Monolíngüe

5.3.2 Definição de Símbolos

No segundo subprocesso, a definição de símbolos deve gerenciar todas as etiquetas que podem ser utilizadas para fazer a identificação de lexemas. No Apertium não existem hierarquias entre etiquetas de dicionários monolíngües. Entretanto para o usuário lingüista tanto esta classificação como a existência de hierarquias são ferramentas importantes para auxiliar no processo. Logo, tratamos e classificamos símbolos utilizando modelos conhecidos, mas inicialmente não utilizados pela MTA tomada como base, servindo apenas de orientação na construção do conhecimento.

Ao selecionar um símbolo propusemos a definição de um código para o símbolo e uma descrição na língua de origem. São apresentadas estatísticas relacionadas às relações dos símbolos dentro das categorias. A interface possibilita a associação de símbolos com outros símbolos para a definição de hierarquias de classificação gramatical e de outra classificação baseada em agrupamentos por conceito. Este subprocesso é mostrado, de forma estática, na

Figura 5.3.

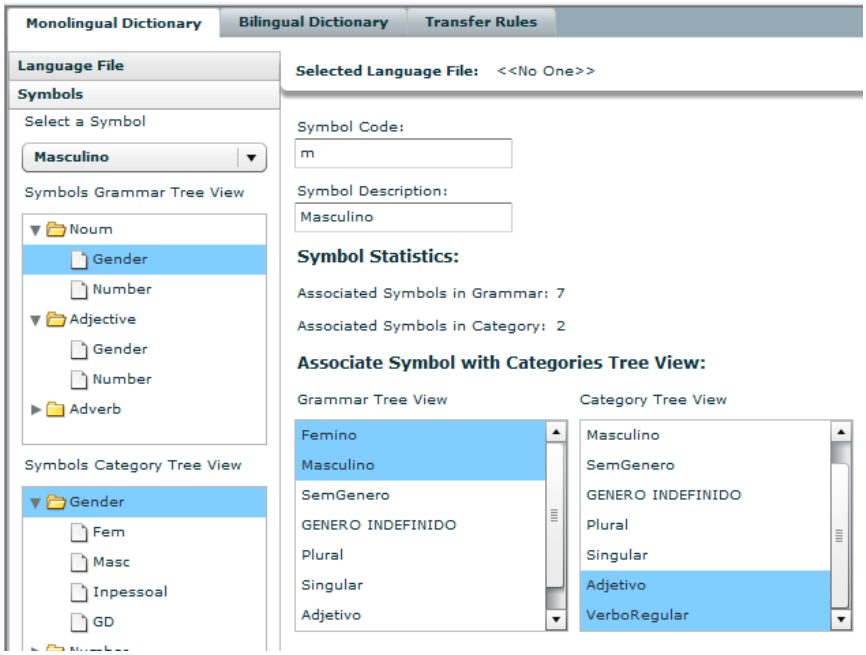

Figura 5.3 Definições associadas a um símbolo

5.3.3 Definição de paradigmas

A definição dos paradigmas é a parte mais complexa do processo de definição de dicionários monolíngües. O usuário inicialmente deve selecionar um paradigma existente, ou criar um novo, definir um nome para o paradigma e uma palavra de exemplo que tenha seu radical separado utilizando-se a nomenclatura padrão do Apertium.

O subprocesso é relativamente complexo e foi subdividido em páginas. A primeira mostra em uma grade todas as flexões proporcionadas pelo paradigma. Para auxílio, esta grade interage com um lexema de exemplo, definido pelo usuário, para que se tenha uma visão de como se dá as flexões (Figura 5.4).

Selecionando-se um elemento da grade é liberado o modo de edição de elementos do paradigma. Neste modo vemos um editor de expressões regulares que podem ocupar a parte direita do elemento e a definição de palavras e cadeias de símbolos que definem aquela classificação, sendo que a construção destas cadeias é feita por "*drag and drop*" de listas que eliminam erros de digitação e tornam as regras mais claras. É possível de se habilitar um

modo avançado de edição onde, além da utilização de lista, um usuário mais avançado pode editar diretamente a regra de flexão (Figura 5.5).

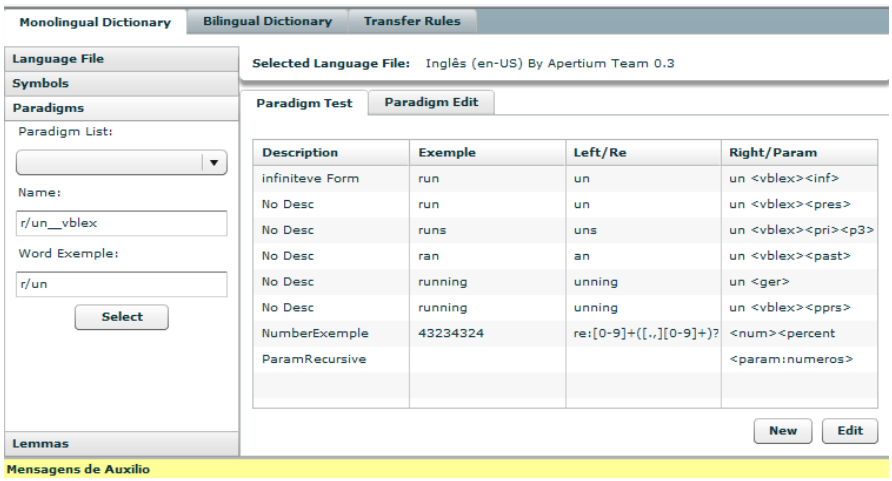

Figura 5.4 Páginas associadas ao subprocesso de Teste de um paradigma

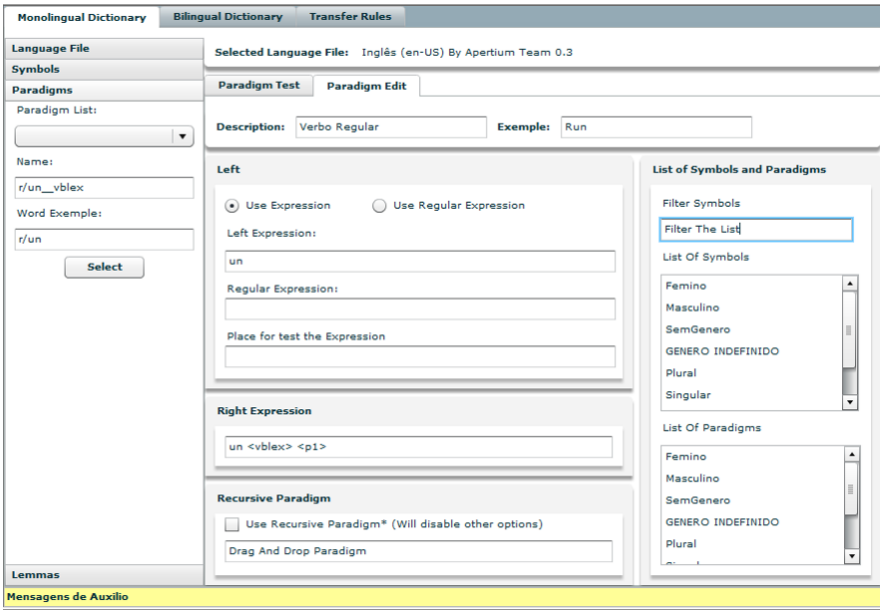

Figura 5.5 Páginas associadas ao subprocesso de edição de um paradigma

5.3.4 Definições de lexemas

A definição de lexemas em sua grande maioria consiste em ligar lexemas a paradigmas. Com este intuito, a proposta consiste em um guia para descobrir se já existem paradigmas para aquele lexema. Neste caso, o usuário entra com a palavra, destacando-se seu radical e até quatro flexões do lexema. O sistema irá varrer os paradigmas disponíveis e propor aqueles

que se validam de acordo com as flexões propostas. Cabe ao usuário escolher a melhor opção e complementar a idéia relacionada à criação do lexema com os dados necessários. A tela que exibe isto é representada pela Figura 5.6.

Caso não exista algum paradigma adequado às necessidades do usuário, ele é guiado à tela de definição de paradigmas, onde ele pode definir um novo paradigma genérico. Para lexemas que possuem uma flexão específica como, palavras compostas, existe a opção de criar um paradigma embutido, exclusivo ao lexema.

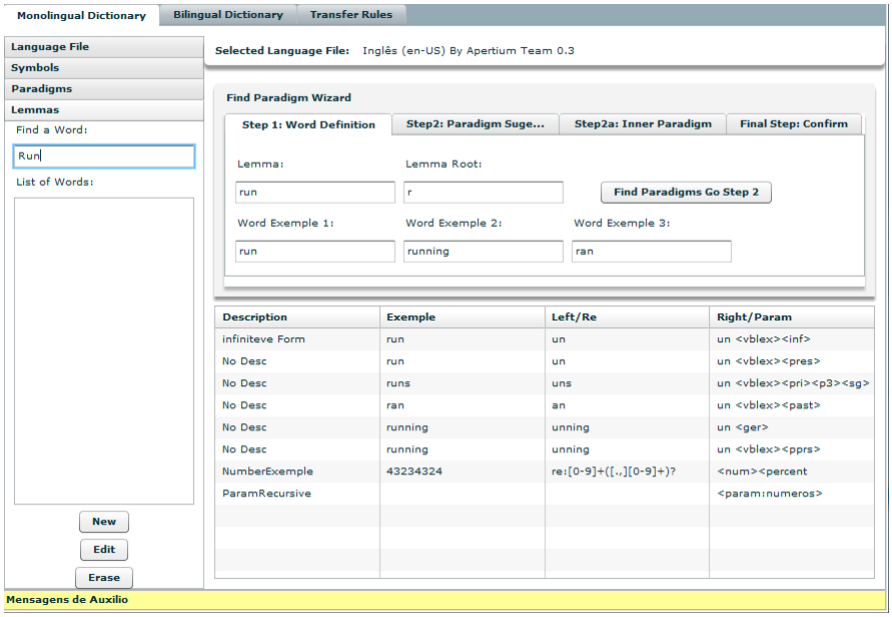

Figura 5.6 Definições associadas a um lexema

5.4 Dicionário Bilíngüe

Dividimos esta etapa em dois subprocessos, sendo eles: definição de Dicionário Bilíngue, e gerenciador de correspondências entre lexemas e expressões.

Após selecionados os dois dicionários Monolíngües que serão relacionados, o sistema prepara a definição do dicionário bilíngüe automaticamente, este momento chamado de definição do Dicionário Bilíngue. Logo na tela inicial estão disponíveis estatísticas sobre os Dicionários Monolíngües e a informações sobre a relação entre eles dentro do Dicionário

Bilíngüe. Neste momento o sistema gera também automaticamente, um conjunto de informações como o alfabeto e uma grade com a listagem de símbolos, resultantes do relacionamento das línguas.

Na grade com os símbolos estão disponíveis as seguintes informações em colunas: o código, a descrição e em qual Dicionário Morfológico o símbolo existe. Estas informações estão disponíveis para orientação na interface, mas também devem estar no arquivo XML resultante do Dicionário Bilíngue. A tela que representa este primeiro momento é representado pela Figura 5.6.

Depois de selecionado o Dicionário Bilíngue a ser trabalhado, temos que iniciar a gerência das relações entre lexemas e expressões. Estas relações são o conteúdo que realmente define o dicionário Bilíngue, o trabalho de construir estas relações se torna peculiar por que é feito apenas a partir de lexemas, simulando uma relação palavra a palavra, as relações são construídas a partir da interlíngua parcial gerada.

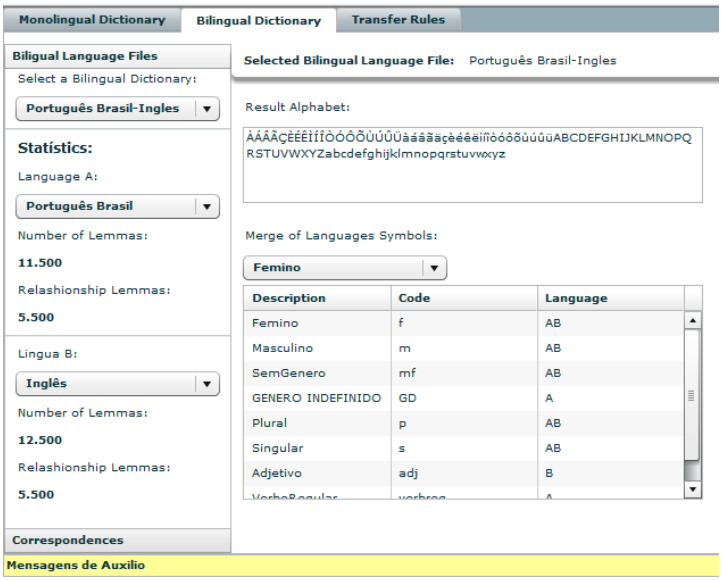

Figura 5.7 Definições associadas ao Dicionário Bilíngüe

A Figura 5.7 representa a interface com as relações existentes para um determinado lexema. O usuário tem disponível na barra lateral opções para filtrar e procurar as definições relacionadas ao lexema na “língua A” (origem) ou “língua B” (destino). Após escolher o

lexema desejado ele tem disponível duas grades com as informações.

A primeira possui todas as possíveis flexões do lexema escolhido e suas correspondências na fase de análise baseado no dicionário morfológico. A segunda mostra as regras associadas ao lexema. Estas são representadas por três colunas: a primeira representa o padrão aceito pela regra, a segunda a saída a ser gerada e a terceira o sentido válido para a regra.

A definição da primeira coluna segue o princípio de casamento de padrões com a regra mais específica ou o padrão mais completo. Este representa a raiz do lexema seguido da sua classificação morfológico gerada pela análise.

O conteúdo da segunda coluna é o conteúdo que substituirá o padrão encontrado, a fim de preparam a fase de geração para a língua de destino. A terceira é o sentido para qual a regra tem validade, pois determinadas regras podem fazer sentido apenas em uma direção do par de línguas.

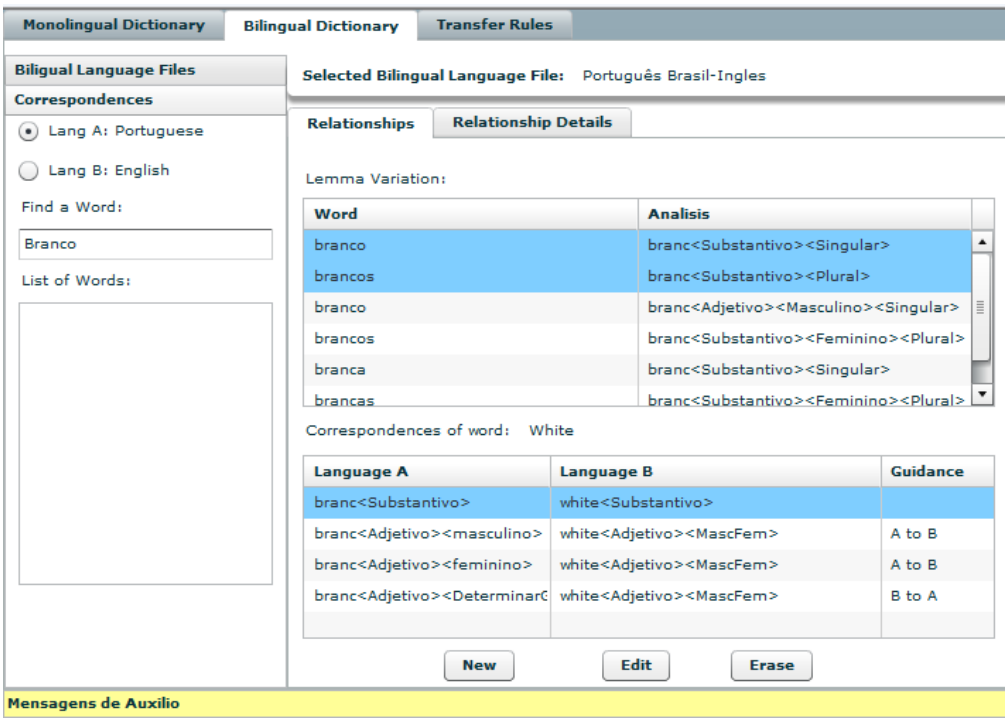

Figura 5.8 Relações entre lexemas e expressões para Dicionários Bilíngüe

CAPÍTULO 6

CONCLUSÃO E TRABALHOS FUTUROS

No presente trabalho, fizemos uma revisão sobre as máquinas de tradução automática, teorias e exemplos de software livre que se encaixam nas idéias propostas. Dentre as atuais alternativas julgamos que o Apertium é uma importante e forte iniciativa na área. Analisamos seus pontos negativos e propusemos alternativas para sua evolução, tanto do ponto de vista da arquitetura quanto uma solução para um dos módulos que consiste em um ambiente de interface-interação que facilita o processo de especificação do conhecimento.

O WiKLaTS pretende ser um sistema que visa à concentração organizada de conhecimento para ser reutilizada para vários fins, sendo que sua arquitetura, apesar de genérica, tem íntima relação com o Apertium. Apresentamos aqui uma visão daquilo que pode ser construído, sendo abertas várias lacunas nas mais variadas subáreas da Ciência da Computação, que vão desde representação de conhecimento até um Sistema especialista para a gestão da plataforma. Em cada módulo proposto há espaço para o desenvolvimento de idéias inovadoras e de funcionalidades que possam enriquecer o resultado final.

Analisamos os requisitos da primeira parte do processo, desenvolvimento de dicionários monolíngües e bilíngües, onde se encontra na versão disponível no Apertium, grande dificuldade ligada à entrada de conhecimento. Nosso protótipo de ambiente de interface desenvolvido para duas das três principais funções do módulo chamado inicialmente de “WiKLang” é a primeira alternativa documentada para a padronização de uma etapa do processo utilizado no Apertium.

Sabemos também que a implementação da presente proposta, ao tornar o uso deste ambiente real, determinará novos problemas e criará novos “gargalos” que deverão ser

tratados posteriormente. É necessária também uma análise mais aprofundada dos conceitos de IHC relacionados ao ambiente, uma vez que não foi possível estender estes conceitos no trabalho em questão.

Dentro dos possíveis trabalhos futuros podemos detalhar os seguintes pontos.

1. Detalhar uma arquitetura de engenharia de software que dê suporte à implementação do sistema com a arquitetura geral descrita e do protótipo criado, visando desafios como a comunicação com o Apertium;
2. Implementar o protótipo não funcional desenvolvido, com o objetivo de, por meio da interação mais exaustiva com a comunidade envolvida, extrair novos requisitos para expandir suas funcionalidades;
3. Estudar mais aprofundadamente a especificação do conhecimento, modelar um novo componente voltado a terceira e última parte não contemplada pelo protótipo anterior, a especificação de regras de tradução entre pares de línguas, que se apresenta com um nível de dificuldade considerável.

   Numa esperada extensão do prazo de concessão da bolsa de Doutorado, pretende-se dar continuidade ao trabalho, fechando o ciclo por meio das seguintes atividades;
4. Adicionar ao resultado da implementação do ciclo anterior, as novas funcionalidades do ambiente relacionadas à especificação de regras de tradução;
5. Avaliar e propor melhorias para o sistema desenvolvido, com base em algumas vertentes teóricas da IHC.
6. Análise das interfaces, pensando na adaptação e representação das mesmas, focando em como utilizá-las no desenvolvimento de pares de línguas da comunidade surdo muda, utilizando-se de recursos como o SignWrite.

APÊNDICE A – DIC. MONOLÍNGÜE PORTUGUÊS

Definição completa do arquivo apertium-pt-en.pt.dix, que define o dicionário monolíngüe de português.

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
   <alphabet>
      ÀÁÂÃÇÈÉÊÌÍÎÒÓÔÕÙÚÛÜàáâãäçèéêëìíîòóôõùúûüABCDEF
      GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
   </alphabet>
   <sdefs>
      <sdef n="subs"    c="substantivo"/>
      <sdef n="adj"     c="adjetivo"/>
      <sdef n="verbreg" c="verbo regular"/>
      <sdef n="sg"      c="singular"/>
      <sdef n="pl"      c="plural"/>
      <sdef n="m"       c="masculino"/>
      <sdef n="f"       c="feminino"/>
      <sdef n="mf"      c="semgenero"/>
      <sdef n="p1"      c="primeira pessoa"/>
      <sdef n="tpres"   c="presente"/>
      <sdef n="num"     c="Numero"/>
   </sdefs>
   <pardefs>
      <pardef n="numeros">
         <e>
            <re>[0-9]+([.,][0-9]+)?</re>
            <p>
               <l/>
               <r><s n="num"/></r>
            </p>
         </e>
      </pardef>
      <pardef n="livro__subs">
         <e>
            <p>
               <l/>
               <r><s n="subs"/><s n="sg"/></r>
            </p>
         </e>
         <e>
            <p>
               <l>s</l>
               <r><s n="subs"/><s n="pl"/></r>
            </p>
         </e>
     </pardef>
      <pardef n="branc/o__adj">
         <e>
            <p>
               <l>o</l>
               <r><s n="adj"/><s n="m"/><s n="sg"/></r>
```

```
                    </p>
                </e>
                <e>
                    <p>
                        <l>os</l>
                        <r><s n="adj"/><s n="m"/><s n="pl"/></r>
                    </p>
                </e>
                <e>
                    <p>
                        <l>a</l>
                        <r><s n="adj"/><s n="f"/><s n="sg"/></r>
                    </p>
                </e>
                <e>
                    <p>
                        <l>as</l>
                        <r><s n="adj"/><s n="f"/><s n="pl"/></r>
                    </p>
                </e>
            </pardef>
            <pardef n="limp/ar__verbreg">
                <e>
                    <p>
                        <l>o</l>
                        <r>ar<s   n="verbreg"/><s   n="tpres"/><s   n="p1"/><s
n="sg"/></r>
                    </p>
                </e>
            </pardef>
        </pardefs>
        <section id="main" type="standard">
            <e lm="livro">
                <i> livro </i>
                <par n="livro__subs"/>
            </e>
            <e lm="branco">
                <i>branc</i>
                <par n="branc/o__adj"/>
            </e>
            <e lm="limpar">
                <i>limp</i>
                <par n="limp/ar__verbreg"/>
            </e>
        </section>
    </dictionary>
```

APÊNDICE B – DIC. MONOLÍNGÜE INGLÊS

Definição completa do arquivo apertium-pt-en.en.dix, que define o dicionário monolíngüe de inglês.

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
   <alphabet>
      ÀÁÂÃÇÈÉÊÌÍÎÒÓÔÕÙÚÛÜàáâãäçèéêëìíîòóôõùúûüABCDEF
      GHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
   </alphabet>
   <sdefs>
      <sdef n="subs"    c="substantivo"/>
      <sdef n="adj"     c="adjetivo"/>
      <sdef n="verbreg" c="verbo regular"/>
      <sdef n="sg"      c="singular"/>
      <sdef n="pl"      c="plural"/>
      <sdef n="m"       c="masculino"/>
      <sdef n="f"       c="feminino"/>
      <sdef n="mf"      c="semgenero"/>
      <sdef n="p1"      c="primeira pessoa"/>
      <sdef n="tpres"   c="presente"/>
      <sdef n="num"     c="Numero"/>
      <sdef n="prn"     c="Pronome"/>
      <sdef n="pess"    c="Pessoal"/>
   </sdefs>
   <pardefs>
      <pardef n="numeros">
         <e>
            <re>[0-9]+([.,][0-9]+)?</re>
            <p>
               <l/>
               <r><s n="num"/></r>
            </p>
         </e>
      </pardef>
      <pardef n="book__subs">
         <e>
            <p>
               <l/>
               <r><s n="subs"/><s n="sg"/></r>
            </p>
         </e>
         <e>
            <p>
               <l>s</l>
               <r><s n="subs"/><s n="pl"/></r>
            </p>
         </e>
     </pardef>
      <pardef n="white__adj">
         <e>
            <p>
```

```
                <l></l>
                <r><s n="adj"/><s n="mf"/><s n="sg"/></r>
             </p>
          </e>
          <e>
             <p>
                <l>s</l>
                <r><s n="adj"/><s n="mf"/><s n="pl"/></r>
             </p>
          </e>
       </pardef>
       <pardef n="clean__verbreg">
          <e>
             <p>
                <l></l>
                <r><s n="verbreg"/><s n="tpres"/></r>
             </p>
          </e>
       </pardef>
       <pardef n="prnpess_prn">
          <e>
     <p>
        <l>I</l>
        <r>prpers<s n="prn"/><s n="pess"/><s n="p1"/><s n="sg"/></r>
     </p>
  </e>
       </pardef>
    </pardefs>

    <section id="main" type="standard">
       <e lm="Pronome pessoal">
          <i></i>
  <par n="prnpess_prn"/>
       </e>
       <e lm="book">
          <i>book</i>
          <par n="book__subs"/>
       </e>
       <e lm="white">
          <i>white</i>
          <par n="white__adj"/>
       </e>
       <e lm="clean">
          <i>clean</i>
          <par n="clean__verbreg"/>
       </e>
    </section>
</dictionary>
```

APÊNDICE C – DIC. BILINGUE PORTUGUÊS INGLES

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
   <alphabet/>
   <sdefs>
      <sdef n="subs" c=”substantivo”/>
      <sdef n="adj" c=”adjetivo”/>
      <sdef n="verbreg" c=”verbo regular”/>
      <sdef n="sg" c=”singular”/>
      <sdef n="pl" c=”plural”/>
      <sdef n="m" c=”masculino”/>
      <sdef n="f" c=”feminino”/>
      <sdef n="mf" c=”semgenero”/>
      <sdef n="p1" c=”primeira pessoa”/>
      <sdef n="tpres" c=”presente”/>
      <sdef n="prn" c="Pronome"/>
      <sdef n="pess" c="Pessoal"/>
   </sdefs>
   <pardefs>
      <!-- Paradigmas a serem definidos -->
   </pardefs>
   <section id="main" type="standard">
      <e><p>
         <l>limpar<s n="verbreg"/></l>
         <r>clean<s n="verbreg"/></r>
      </p></e>
      <e><p>
         <l>livro<s n="subs"/></l>
         <r>book<s n="subs"/></r>
      </p></e>
      <e><p>
         <l>branco<s n="adj"/></l>
         <r>white<s n="adj"/></r>
      </p></e>
   </section>
</dictionary>
```

APÊNDICE D – REGRAS DE TRANSFERENCIA PT-EN

```
<?xml version="1.0" encoding="UTF-8"?>
<transfer>
   <section-def-cats>
      <!-- Definição de Categorias -->
      <def-cat n="pronpes">
         <cat-item lemma="pronpes" tags="prn.*"/>
      </def-cat>
      <def-cat n="vrb">
         <cat-item tags="verbreg.*"/>
      </def-cat>
   </section-def-cats>
   <section-def-attrs>
      <!-- Definição de Atributos -->
      <def-attr n="tipo_prn">
         <attr-item tags="prn.subj"/>
       <attr-item tags="prn.obj"/>
    </def-attr>
    <def-attr n="numero">
       <attr-item tags="sg"/>
       <attr-item tags="pl"/>
    </def-attr>
    <def-attr n="tempo">
       <attr-item tags="pri"/>
    </def-attr>
    <def-attr n="pessoa">
       <attr-item tags="p1"/>
    </def-attr>
   </section-def-attrs>
   <section-def-vars>
      <!-- Definição de Variáveis -->
   </section-def-vars>
   <section-rules>
      <!-- Definição de Regras -->
<rule>
   <pattern>
      <pattern-item n="vrb"/>
   </pattern>
   <action>
      <out>
         <lu>
            <lit v="pronpes"/>
            <lit-tag v="prn"/>
            <lit-tag v="pess"/>
            <clip pos=1 side=tl part="pessoa"/>
            <clip pos=1 side=tl part="numero"/>
         </lu>
         <b/>
         <lu>
            <clip pos=1 side=tl part="lem"/>
            <clip pos=1 side=tl part="a_verb"/>
            <clip pos=1 side=tl part="temps"/>
         </lu>
      </out>
```

```
   </action>
</rule>
   </section-def-rules>
</transfer>
```

ANEXO A – DTDS XMLs DE DICIONÁRIOS

```
<!--      DTD for the format of dictionaries  -->
<!ELEMENT dictionary (alphabet?, sdefs?,
            pardefs?, section+)>
<!-- root element-->

<!ELEMENT alphabet (#PCDATA)>
<!-- alphabetic character list -->

<!ELEMENT sdefs (sdef+)>
<!-- symbol definition section -->

<!ELEMENT sdef EMPTY>
<!-- symbol definition -->
<!ATTLIST sdef
n ID #REQUIRED
>
<!-- n: symbol (tag) name -->
<!ATTLIST sdef
c CDATA  #IMPLIED
>
<!-- c: symbol (tag) comment -->

<!ELEMENT pardefs (pardef+)>
<!-- paradigm definition section -->

<!ELEMENT pardef (e+)>
<!-- paradigm definition -->
<!ATTLIST pardef
n CDATA #REQUIRED
>
<!-- n: paradigm name -->
<!ATTLIST pardef
        c CDATA  #IMPLIED
>
        <!-- c: comment about paradigm -->

<!ELEMENT section (e+)>
<!-- dictionary section -->
<!ATTLIST section
id ID #REQUIRED
type (standard|inconditional|postblank|preblank) #REQUIRED

>
<!-- id: dictionary section identifier -->
<!-- type: dictionary section type -->

<!ELEMENT e (i | p | par | re)+>
<!-- entry -->
<!ATTLIST e
r (LR|RL) #IMPLIED
lm CDATA #IMPLIED
a CDATA #IMPLIED
c CDATA #IMPLIED
```

```
i CDATA #IMPLIED
slr CDATA #IMPLIED
srl CDATA #IMPLIED
>
<!-- r: restriction LR: left-to-right,
                RL: right-to-left -->
<!-- lm: lemma -->
<!-- a: author -->
<!-- c: comment -->
<!-- i: ignore ('yes') means ignore, otherwise it is not ignored) -->
<!-- slr: translation sense when translating from left to right -->
<!-- srl: translation sense when translating from right to left -->
<!ELEMENT par EMPTY>
<!-- reference to paradigm -->
<!ATTLIST par
n CDATA #REQUIRED

>
<!-- n: paradigm name -->

<!ELEMENT i (#PCDATA | b | s | g | j | a)*>
<!-- identity -->

<!ELEMENT re (#PCDATA)>
<!-- regular expression identification -->

<!ELEMENT p (l, r)>
<!-- pair of strings -->

<!ELEMENT l (#PCDATA | a | b | g | j | s)*>
<!-- left part of p -->

<!ELEMENT r (#PCDATA | a | b | g | j | s)*>
<!-- right part of p -->

<!ELEMENT a EMPTY>
<!-- post-generator wake-up mark -->

<!ELEMENT b EMPTY>
<!-- blank chars block mark -->

<!ELEMENT g (#PCDATA | a | b | j | s)*>
<!-- mark special groups in lemmas -->
<!ATTLIST g
i CDATA #IMPLIED
>
<!-- i is used to co-index groups in the left with those -->
<!-- on the right of a pair -->

<!ELEMENT j EMPTY>
<!-- join lexical forms -->

<!ELEMENT s EMPTY>
<!-- reference to symbol (tag) -->
<!ATTLIST s
n IDREF #REQUIRED

>
<!-- n: symbol (tag) name -->
```

ANEXO B – DTDS XMLs DE REGRAS

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
    Draft of DTD for the structural transfer rule files
    Sergio Ortiz, Gema Ramírez-Sánchez, Mireia Ginestí, Mikel L.
Forcada,
    2005.07.29.
-->

<!ENTITY % condition "(and|or|not|equal|begins-with|begins-with-
list|ends-with|ends-with-list|contains-substring|in)">
<!ENTITY % container "(var|clip)">
<!ENTITY % sentence "(let|out|choose|modify-case|call-macro|append)">
<!ENTITY % value "(b|clip|lit|lit-tag|var|get-case-from|case-
of|concat)">
<!ENTITY % stringvalue "(clip|lit|var|get-case-from|case-of)">

<!ELEMENT transfer (section-def-cats, section-def-attrs, section-def-
vars, section-def-lists?, section-def-macros?, section-rules)>
<!ATTLIST transfer default (lu|chunk) #IMPLIED>
<!--
     'transfer' is the root element containing the whole structural
     transfer rule file. Attribute 'default' specifies if
     unmatched words have to be written as lexical units ("lu", this
is
     the default value) or as chunks ("chunk").
-->

<!ELEMENT section-def-cats (def-cat+)>
<!--
     The 'def-cats' section defines the categories used to build the
patterns used in rules
 -->

<!ELEMENT def-cat (cat-item+)>
<!ATTLIST def-cat  n ID #REQUIRED>
<!--
     Each 'def-cat' defines one category in terms of a list of
     category items and has a unique name 'n', which is mandatory
-->

<!ELEMENT cat-item EMPTY>
<!ATTLIST cat-item lemma CDATA #IMPLIED
                   tags CDATA #REQUIRED >
<!--
          Each 'cat-item' (category item) represents a set of lexical
forms
     and has a mandatory attribute 'tags' whose value is a sequence
of
     dot-separated tag names; this sequence is a subsequence of the
     tag sequence defining each possible lexical form. For example,
     tags="n.f" would match all lexical forms containing this tag
     sequence, such as "^casa<n><f><pl>$".
```

```
          In addition, an optional attribute, "lemma", may be used to
          define lexical forms having a particular substring in their
lemma
     -->

     <!ELEMENT section-def-attrs (def-attr+)>

     <!--
          The 'def-attrs' section defines the attributes that will be
          identified in matched lexical forms
     -->

     <!ELEMENT def-attr (attr-item+)>
     <!ATTLIST def-attr n ID #REQUIRED>
     <!--
          Each def-attr defines one attribute in terms of a list of
          attribute items and has a mandatory unique name n
     -->

     <!ELEMENT attr-item EMPTY>
     <!ATTLIST attr-item tags CDATA #IMPLIED>
     <!--
          Each 'attr-item' specifies a subsequence of the tags in
          that lexical form (attribute 'tags')
     -->

     <!ELEMENT section-def-vars (def-var+)>
     <!--
          The 'def-vars' section defines the global variables
          that will be used to transfer information between rules
     -->

     <!ELEMENT def-var EMPTY>
     <!ATTLIST def-var n ID #REQUIRED
                       v CDATA #IMPLIED>
     <!--
          The definition of a global variable has a mandatory unique name
'n' that
          will be used to refer to it. A value of initialization can also
be specified
          by means the 'v' attribute.    The default value of the
initialization is the
          empty string.
     -->

     <!ELEMENT section-def-lists (def-list)+>
     <!--
          Element 'section-def-lists' encloses a set of list definitions
     -->

     <!ELEMENT def-list (list-item+)>
     <!ATTLIST def-list n ID #REQUIRED>
     <!--
          The 'def-list' element defines a named list to search with the
'in'
          element.  Attribute 'n' sets the name of the list
     -->

     <!ELEMENT list-item EMPTY>
     <!ATTLIST list-item v CDATA #REQUIRED>
     <!--
```

```
          Attribute 'v' of 'list-item' element contains the value to be
added to
          the list being defined
     -->

     <!ELEMENT section-def-macros (def-macro)+>
     <!--

          The 'def-macros' section defines macros containing portions of
          code frequently used in the action part of rules

     -->

     <!ELEMENT def-macro (%sentence;)+>
     <!ATTLIST def-macro n ID #REQUIRED>
     <!ATTLIST def-macro npar CDATA #REQUIRED>
     <!--
          Macro definition:

          A macro has a mandatory name (the value of 'n'), a number of
parameters
          (the value of 'npar') and a body containing arguments and
statements.
     -->

     <!ELEMENT section-rules (rule+)>
     <!--
          The rules section contains a sequence of one or more rules
     -->

     <!ELEMENT rule (pattern, action)>
     <!ATTLIST rule comment CDATA #IMPLIED>
     <!--
           Each rule has a pattern and an action
           * attribute 'comment' allows to put in comments about the
purpose of
             the rule being defined
     -->

     <!ELEMENT pattern (pattern-item+)>
     <!--
     The pattern is specified in terms of pattern items, each one
     representing a lexical form in the matched pattern
     -->

     <!ELEMENT pattern-item EMPTY>
     <!ATTLIST pattern-item n IDREF #REQUIRED>
     <!--
            Each attribute to be activated is referred to by its name in
the def-cats section
     -->

     <!ELEMENT action (%sentence;)*>
     <!--
            Encloses the procedural part of a rule
     -->

     <!ELEMENT choose (when+,otherwise?)>
     <!--
          The choose statement is a selection statement (similar to a case
          statement) composed of one or more tested cases and an optional
```

```
          otherwise
     -->

     <!ELEMENT when (test,(%sentence;)*)>
     <!--
          Each tested case is a block of zero or more statements
     -->

     <!ELEMENT otherwise (%sentence;)+>
     <!--
          The otherwise case is also a block of one or more statements
     -->

     <!ELEMENT test (%condition;)>
     <!--
          The test in a tested case may be a conjunction, a disjunction,
or
          a negation of simpler tests, as well as a simple equality test
     -->

     <!ELEMENT and ((%condition;),(%condition;)+)>
     <!--
          Each conjuntion test contains two or more simpler tests
     -->

     <!ELEMENT or ((%condition;),(%condition;)+)>
     <!--
          Each disjunction test contains two or more simpler tests
     -->

     <!ELEMENT not (%condition;)>
     <!--
          The negation of a simpler test is a test itself
     -->

     <!ELEMENT equal (%value;,%value;)>
     <!ATTLIST equal caseless (no|yes) #IMPLIED>
     <!--
           The simplest test is an equality test. The right part and the
           left part of the equality may both be a clip (see below), a
           literal string ('lit'), a literal tag ('lit-tag') or the value
of
           a variable ('var') defined in the def-vars section.  When the
attribute
           'caseless' is set to 'yes', the comparison is made without
attending
           to the case.
     -->

     <!ELEMENT begins-with (%value;,%value;)>
     <!ATTLIST begins-with caseless (no|yes) #IMPLIED>
     <!--
           Tests if the left part contains the right part at the
beginning.
           Both parts of the test may both be a clip (see below), a
           literal string ('lit'), a literal tag ('lit-tag') or the value
of
           a variable ('var') defined in the def-vars section.  When the
attribute
           'caseless' is set to 'yes', the comparison is made without
attending
```

```
            to the case.
      -->

      <!ELEMENT ends-with (%value;,%value;)>
      <!ATTLIST ends-with caseless (no|yes) #IMPLIED>
      <!--
            Tests if the left part contains the right part at the end.
            Both parts of the test may both be a clip (see below), a
            literal string ('lit'), a literal tag ('lit-tag') or the value
of
            a variable ('var') defined in the def-vars section.  When the
attribute
            'caseless' is set to 'yes', the comparison is made without
attending
            to the case.
      -->

      <!ELEMENT begins-with-list (%value;,list)>
      <!ATTLIST begins-with-list caseless (no|yes) #IMPLIED>
      <!--
            Tests if the left part contains the right part at the
beginning.
            First parts of the test may be a clip (see below), a
            literal string ('lit'), a literal tag ('lit-tag') or the value
of
            a variable ('var') defined in the def-vars section. The second
part
            must be always a list.  When the attribute
            'caseless' is set to 'yes', the comparison is made without
attending
            to the case.
      -->


      <!ELEMENT ends-with-list (%value;,list)>
      <!ATTLIST ends-with-list caseless (no|yes) #IMPLIED>
      <!--
            Tests if the left part contains the right part at the end.
            First parts of the test may be a clip (see below), a
            literal string ('lit'), a literal tag ('lit-tag') or the value
of
            a variable ('var') defined in the def-vars section. The second
part
            must be always a list.  When the attribute
            'caseless' is set to 'yes', the comparison is made without
attending
            to the case.
      -->


      <!ELEMENT contains-substring (%value;,%value;)>
      <!ATTLIST contains-substring caseless (no|yes) #IMPLIED>
      <!--
            Tests if the left part contains the right part.
            Both parts of the test may both be a clip (see below), a
            literal string ('lit'), a literal tag ('lit-tag') or the value
of
            a variable ('var') defined in the def-vars section.  When the
attribute
            'caseless' is set to 'yes', the comparison is made without
attending
```

```
            to the case.
      -->


      <!ELEMENT in (%value;, list)>
      <!ATTLIST in caseless (no|yes) #IMPLIED>
      <!--
          'in' performs a search of a value in a list.  If 'caseless' is
set to yes,
          this search is performed without attending to the case
      -->

      <!ELEMENT list EMPTY>
      <!ATTLIST list n IDREF #REQUIRED>
      <!--
          'list' refers, with the name in attribute 'n', a list defined
before in
          the 'section-def-list' section
      -->

      <!ELEMENT let (%container;, %value;)>
      <!--
            An assignment statement ('let') assigns the value of a clip
(see
            below), a literal string ('lit'), a literal tag('lit-tag') or
the
            value of a global variable ('var') to either a global variable
('var')
            or a clip
      -->

      <!ELEMENT append (%value;)+>
      <!ATTLIST append n IDREF #REQUIRED>
      <!--
            This instruction appends the value of a clip (see
            below), a literal string ('lit'), a literal tag('lit-tag') or
the
            value of a global variable ('var') to either a global variable
('var')
            or a clip, identified by the "n" attribute
      -->


      <!ELEMENT out (mlu|lu|b|chunk|var)+>
      <!--
            'out' is an output statement; it may output any sequence of
            clips, literal strings, literal tags, variables, and whitespace
items
            (see below)
      -->

      <!ELEMENT modify-case (%container;, %stringvalue;)>
      <!--
            The first argument of 'modify-case' copy the case of the second
            argument.
      -->

      <!ELEMENT call-macro (with-param)*>
      <!ATTLIST call-macro n IDREF #REQUIRED>
```

```
      <!--
            A macro may be called anywhere by name with one or more
            arguments
      -->

      <!ELEMENT with-param EMPTY>
      <!ATTLIST with-param pos CDATA #REQUIRED>
      <!--
            The attribute pos in each argument is used to refer to a
lexical
            form in the current rule. For example, if a 2-parameter macro
            has been defined to perform noun-adjective agreement
operations,
            it may be used with arguments 1 and 2 in a noun-adjective rule,
            with arguments 2, 3 and 1 in a determiner-noun-adjective rule,
with
            arguments 1 and 3 in a noun-adverb-adjective rule, and with
            arguments 2 and 1 in an adjective-noun rule
      -->

      <!ELEMENT clip EMPTY>
      <!ATTLIST clip pos CDATA #REQUIRED
                     side (sl|tl) #REQUIRED
                     part CDATA #REQUIRED
                     queue CDATA #IMPLIED
                     link-to CDATA #IMPLIED>
      <!--
            A 'clip' is a substring of a source-language or target-language
            lexical form, extracted according to an attribute:

            * 'pos' is an index (1, 2, 3...) used to select a lexical form
               inside the rule;

            * 'side' is used to select a source-language ('sl') or a
              target-language ('tl') clip

            * the value of 'part' is the name of an attribute defined in
              def-attrs, but may take also the values 'lem' (referring to
              the lemma of the lexical form), 'lemh' (lemma head), 'lemq'
              (lemma queue) and 'whole' (referring to the whole lexical
form).

            * the value of 'queue' may be 'no' or 'yes'.  'yes' is assumed
by
              default.

            * 'link-to' causes the other attributes to be ignored in clip
evaluation
              when using 'clip' as a right hand side element (as value),
and
              returns its value.  When using as a left hand side (as
reference),
              the value of the 'as' attribute is ignored.
      -->

      <!ELEMENT lit EMPTY>
      <!ATTLIST lit v CDATA #REQUIRED>
      <!--
            A literal string value: the value of the literal is the value
of
            the 'v' attribute
```

```
      -->

      <!ELEMENT lit-tag EMPTY>
      <!ATTLIST lit-tag v CDATA #REQUIRED>
      <!--
            A literal string value: the value of the literal is the value
of
            the 'v' attribute
      -->


      <!ELEMENT var EMPTY>
      <!ATTLIST var n IDREF #REQUIRED>
      <!--
           Each 'var' is a variable identifier: the attribute n is the name
           of the variable. When it is in an 'out', a 'test', or the right
           part of a 'let', it represents the value of the variable; when
in
           the left part of a 'let' it represents the reference of the
           variable.
      -->

      <!ELEMENT get-case-from (clip|lit|var)>
      <!ATTLIST get-case-from pos CDATA #REQUIRED>
      <!-- Atención, falta modificar todos los comentarios donde intervenga
      get-case-from -->

      <!ELEMENT case-of EMPTY>
      <!ATTLIST case-of pos CDATA #REQUIRED
                     side (sl|tl) #REQUIRED
                     part CDATA #REQUIRED>
      <!--
            A 'case-of' is a value representing the case of a "clip".  This
value
            will be "aa" (all lowercase), "Aa" (first uppercase) and "AA",
            (all uppercase).

            * 'pos' is an index (1, 2, 3...) used to select a lexical form
               inside the rule;

            * 'side' is used to select a source-language ('sl') or a
              target-language ('tl') clip

            * the value of 'part' is the name of an attribute defined in
              def-attrs, but may take also the values 'lem' (referring to
              the lemma of the lexical form), 'lemh' (lemma head), 'lemq'
              (lemma  queue)  and  'whole'  (referring  to  the  whole  lexical
form).
      -->


      <!ELEMENT concat (%value;)+>
      <!-- Concatenates a sequence of values -->

      <!ELEMENT mlu (lu+)>
      <!-- Encloses a multiword -->

      <!ELEMENT lu (%value;)+>
      <!-- Encloses a word inside an 'out' element. -->

      <!ELEMENT chunk (tags,(mlu|lu|b|var)+)>
```

```
      <!ATTLIST chunk name CDATA #IMPLIED
                      namefrom CDATA #IMPLIED
                      case CDATA #IMPLIED>
      <!--
           Encloses a chunk inside an 'out' element.
           * 'name' the pseudolemma of the chunk.
           * 'namefrom' get the name from a variable.
           * 'case' the variable to get the uppercase/lowercase policy
              to apply it to the chunk name
      -->

      <!ELEMENT tags (tag+)>
      <!ELEMENT tag (%value;)>

      <!ELEMENT b EMPTY>
      <!ATTLIST b pos CDATA #IMPLIED>
      <!--
           'b' is a [super]blanks item, indexed by pos; for example, a 'b'
           with pos="2" refers to the [super]blanks (including format data
           encapsulated by the de-formatter) between lexical form 2 and
           lexical form 3. Managing [super]blanks explicitly allows for the
           correct placement of format when the result of structural
           transfer has more or less lexical items than the original or has
           been reordered in some way.   If attribute "pos" is not
specified, then
           a single blank (ASCII 32) is generated.
      -->
```

BIBLIOGRAFIA:

[1] J. HUTCHINS. Current commercial machine translation systems and computer-based translation tools: system types and their uses. *International Journal of Translation (ITJ-2005)* vol.17, no.1-2, Jan-Dec 2005, pp.5-38. Disponível em http://www.hutchinsweb.me.uk/IJT-2005.pdf

[2] M. L. Forcada, B. I. Bonev, S. Ortiz-Rojas, J. A. Pérez-Ortiz, G. Ramírez-Sánchez, F. Sánchez-Martínez, C. Armentano-Oller, M. A. Montava and F. M. Tyers. Documentation of the Open-Source Shallow-Transfer Machine Translation Platform Apertium, 2008. Acessado em 20/05/2009 e disponível em: http://xixona.dlsi.ua.es/~fran/apertium2-documentation.pdf.

[3] A. M. Corbí-Bellot, M. L. Forcada, S. Ortiz-Rojas, J. A. Pérez-Ortiz, G. Ramírez-Sánchez, F. Sánchez-Martínez, I. Alegria, A. Mayor, K. Sarasola. An opensource shallow-transfer machine translation engine for the Romance languages of Spain. *Proceedings of the 10th European Association for Machine Translation Conference*, Budapest, Hungary, 2005, 79–86a Disponível em: http://www.dlsi.ua.es/~mlf/docum/corbibellot05p.pdf.

[4] G. Ramírez-Sánchez, F. Sánchez-Martínez, S. Ortiz-Rojas, J. A. Pérez-Ortiz, M. L. Forcada. Opentrad Apertium open-source machine translation system: an opportunity for business and research. *Proceedings of Translating and the Computer 28 Conference, London*, 2006.

[5] L. P. OLIVEIRA. Escolhas pedagógicas do educador e identidade cultural dos aprendizes. *Linguagem e Ensino*. Vol. 3, n° 2, pp. 49-59, 2000.

[6] W. J. Hutchins and H. L. Somers. An Introduction to Machine Translation. London, 1992, Academic Press. ISBN 0-12-362830-X. Disponível em: http://www.hutchinsweb.me.uk/IntroMT-TOC.htm.

[7] P. SANTOS. "Tradução automática". *Engenharia da Linguagem*. Org. Maria Helena M. Mateus e António Horta Branco. Lisboa, Edições Colibri, 1995, pp. 121-128.

[8] S. NIRENBURG. Knowledge and Choices in Machine Translation. *Machine Translation*. Org. Sergei Nirenburg. Cambridge, Cambridge University Press, 1987, pp. 1-15.

[9] A. K. MELBY and C. T. Warner. The Possibility of Language. *Amsterdam/Philadelphia: John Benjamins Publishing Company,* 1995.

[10] ALPAC (1966). Language and machines: computers in translation and linguistics. Report by the Automatic Language Processing Advisory Committee, Division of Behavioral Sciences, National Academy of Sciences, National Research Council. Washington, DC: National Academy of Sciences, National Research Council.

[11] J. Slocum. A Survey of Machine Translation: Its History, Current Status, and Future Prospects. *Machine Translation Systems*. Org. Jonathan Slocum. Cambridge, Cambridge University Press, 1985, pp.1-41.

[12] A. G. Désilets. Translation Wikified: How will Massive Online Collaboration Impact the World of Translation?. *Opening Keynote at Translating and the Computer 29*, 2007.

[13] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg. Who is an open source software developer?. *Commun. ACM* 45, 2 (Feb. 2002), 67-72. Disponível em: http://doi.acm.org/10.1145/503124.503125

[14] B. Scott and A. Barreiro: OpenLogos MT and the SAL representation language. *Proceedings of the First International Workshop on Free/Open-Source Rule-Based*

Machine Translation / Edited by J. A. Pérez-Ortiz, F. Sánchez-Martínez, F. M. Tyers. Alicante, Spain: Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos. 2-3 November 2009, pp. 19-26

[15] C. Armentano-Oller and M. L. Forcada. Open-source machine translation between small languages: Catalan and Aranese Occitan. In Strategies for developing machine translation for minority languages (5th SALTMIL workshop on Minority Languages), pages 51–54, 2006, (conjunction with LREC 2006) Disponível em: http://www.dlsi.ua.es/~mlf/docum/armentano06p2.pdf.

ALÉSSIO MIRANDA JÚNIOR

WiKLaTS –UM AMBIENTE DE INTERFACE E INTERAÇÃO PARA MANIPULAÇÃO E FORMALIZAÇÃO DE CONHECIMENTO PARA TRADUÇÃO ENTRE PARES DE LÍNGUAS BASEADA EM REGRAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Profª. Drª. Laura Sánchez García.

CURITIBA

2009