From Parallel Grammar Development towards Machine Translation – A Project Overview –

Anette Frank
Xerox Research Centre Europe
6, Chemin de Maupertuis, 38240 Meylan, France

Abstract

We give an overview of a MT research project jointly undertaken by Xerox PARC and XRCE Grenoble. The project builds on insights and resources in large-scale development of parallel LFG grammars. The research approach towards translation focuses on innovative computational technologies which lead to a flexible translation architecture. Efficient processing of "packed" ambiguities not only enables ambiguity preserving transfer. It is at the heart of a flexible architectural design, open for various extensions which take the right decisions at the right time.

1 Introduction

Most of the existing high-performance MT systems are based on linguistic technology of the 60s, whereas research in NLP has established "higher-level" syntactic formalisms which allow for specification and processing of declarative, reversible grammars that assign rich structures to natural language sentences. Syntactic theories like Lexical-Functional Grammar (LFG), Head-Driven Phrase Structure Grammar (HPSG), or Generalized Phrase Structure Grammar (GPSG) are prime examples of declarative syntactic formalisms that assign natural language sentences, beyond a constituent structure, levels of representation which encode morphosyntactic, lexical and – most importantly – functional syntactic information. This high-level functional information is valuable for high quality NLP applications like machine translation, information extraction, or other scenarios where information about "who did what to whom" is relevant.

Over many years, research in Machine Translation has explored the potential of higher-level syntactic formalisms for new generation MT technology.

Linguistic research at Xerox is focusing on the theory and application of finite-state technology in NLP, as well as research in the linguistic, mathematical and computational foundations of Lexical-Functional Grammar. Theoretical contributions to research into LFG include the development of processing algorithms, principles of syntactic description for various linguistic phenomena (e.g. long distance dependencies, coordination), and extensions of the LFG projection architecture for semantic representation and translation (see Dalrymple et al 1995).

Since 1995, the ParGram (Parallel LFG Grammar Development) project, a joint initiative of Xerox PARC, XRCE Grenoble, and the University of Stuttgart (IMS), has investigated the potential of LFG for large-scale NLP applications. ParGram encompasses linguistic research in LFG-based parallel grammar development for different languages – English, French and German, and recently Norvegian with the University of Bergen as a new partner.

Along with the ParGram project, Xerox PARC has developed the XLE system (Xerox Linguistic Environment), a platform for large-scale LFG grammar development. XLE comprises interfaces to finite-state preprocessing modules for tokenization and morphological analysis, as well as an efficient parser and generator for LFG grammars. The XLE parser (and a new generator under development) are based on advanced computational technologies. The development of the XLE system constitutes an independent research project into computational algorithms for parsing, generation, and most recently, transfer.

2 Parallel Grammar Development for Multilingual NLP

Lexical-Functional Grammar (Bresnan 1982) is particularly well suited for high-level syntactic analysis in multilingual NLP tasks. The LFG formalism assigns natural language sentences two levels of linguistic representation – a constituent phrase structure (c-structure) and a functional structure (f-structure) – which are related in terms of a functional projection, or correspondence function (ϕ–projection). The c-structure encodes constituency (dominance) and surface order (precedence). The f-structure is an attribute-value representation which encodes syntactic information in terms of morphosyntactic features (NUM, GEND, TENSE, etc.) as well as functional rela-

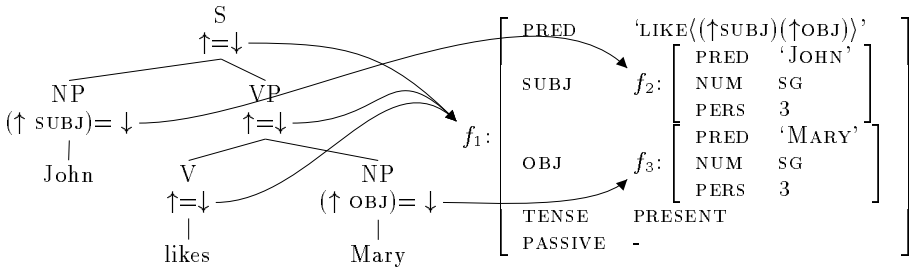

Figure 1: LFG projection architecture

tions between predicates and their arguments or adjuncts. The two levels of representation are related via the correspondence function ϕ, which maps partial c-structures to partial f-structures (see Fig.1).

The separation between the surface oriented c-structure and the more abstract representation of functional syntactic properties makes it possible to provide syntactic descriptions for typologically diverse languages which differ radically in terms of their c-structure properties, while relating them – via the ϕ-projection – to the level of functional representation, which encodes functional syntactic properties that are shared across typologically distinct languages. This makes the f-structure representation provided by LFG-based analysis attractive for multilingual NLP tasks, such as Machine Translation.

The ParGram project explores this potential of LFG as a framework for "parallel" syntactic description of various languages for multilingual NLP tasks. Large-scale LFG grammars have been developed for English, French and German, both under an engineering perspective (grammar engineering techniques for large-scale grammar development) and a linguistic research perspective (the development of principles for parallel f-structure representation across languages). Both aspects are documented in (Butt et al 1999) with further references on special issues in both areas. LFG grammars are declarative and thus reversible for generation. Both in parsing and generation mode, a constraint ranking mechanism provided by XLE filters syntactic and lexical ambiguities (Frank et al 1998). In generation mode, special constraint rankings are used to restrict surface order.

3 Computational technology for LFG-based NLP applications

XLE as a grammar development platform
The grammar development platform XLE (Xerox Linguistic Environment) is an efficient reimplementation of its precursor system, the Xerox LFG Grammar Writer's Workbench (Kaplan and Maxwell 1996).[1]

[1]The Grammar Writer's Workbench, written in Medley Lisp, provides a complete implementation of the LFG formalism, and is particularly useful for teaching purposes and small experimental grammars. The system can be freely downloaded from `http://www.parc.xerox.com/istl/groups/nltt/medley/`. The XLE system is implemented in C and Tcl/Tk, operating under Unix, with plans to port to Windows NT. It covers (basically) the same range of the LFG formalism as its precursor system, but is better suited for large-scale grammar development.

XLE as a grammar development platform comes with an interface to finite-state transducers for tokenization and morphological analysis (Kaplan and Newman 1997). A cascade of tokenizers and normalizers segments the input string into tokens, which are then "looked up" in finite-state morphological transducers. The integration of morphological analysis allows to automatically generate large LFG lexica for open class categories like nouns, adjectives, adverbs, etc. They are created by generic LFG lexicon entries which specify f-structure annotations for morphological and lexical information provided by the morphology. While each grammar comes with hand-coded core LFG lexica for closed class "syntactic" lexical items, XLE supports integration and processing of large-size subcategorization lexica, which are extracted and converted from machine-readable dictionaries (Brazil 1997), or obtained by use of corpus analysis tools (Kuhn et al 1998).

Algorithms and architectures for unification-based grammar processing
The parsing and generation algorithms realized in XLE are based on insights from research into efficient processing algorithms for parsing and generation with unification-based grammars, in particular (Maxwell and Kaplan 1989, 1993, 1996) and (Shemtov 1997).

While context-free phrase structure grammars allow for parsing in polynomial time, grammar formalisms that in addition specify feature constraints can be NP-complete or undecidable, and parse in worst-case exponential or infinite time.

Maxwell and Kaplan (1993) investigate the computational properties of standard hybrid parsing architectures for unification-based grammars. They propose alternative *non-interleaved* processing architectures, which exploit various computational interface properties that support sub-exponential parsing.

The unification algorithm described in (Maxwell and Kaplan 1996) automatically takes advantage of

simple context-free equivalence in the feature space. As a result, sentences parse in cubic time in the typical case, while still being exponential in the worst case.

The theoretical findings of Maxwell and Kaplan (1989,1993,1996) and (Shemtov 1997) are realized in the XLE parsing and generation algorithms, which both exploit sub-exponential computational properties in a non-interleaved architecture of *factored pruning*, combined with *contexted unification* (see below).[2] Through various optimizations and careful implementation XLE has evolved to a high-performance parsing and generation system for LFG grammars.

Contexted constraint satisfaction and processing of packed ambiguities

Disjunctive statements of linguistic constraints allow for a transparent and modular specification of linguistic generalizations. Yet, the resolution of disjunctive feature constraint systems is expensive, in the worst case exponential, whereas conjunctive constraint systems can be solved by standard unification algorithms which do not present a computational problem.

In standard approaches to disjunctive constraint satisfaction, disjunctive formulas are converted to disjunctive normal form (DNF). Conjunctive constraint solving is then applied to each of the resulting conjunctive subformulas. The possibly exponential number of such subformulas results in an overall worst-case exponential process. It is important to note that by conversion to DNF individual facts are replicated in several distinct conjunctive subformulas. This means that they have to be recomputed many times.

$$\begin{array}{ll}\text{DNF} & (a \vee b) \wedge x \wedge (c \vee d) \\ \Rightarrow & (a \wedge x \wedge c) \vee (a \wedge x \wedge d) \vee (b \wedge x \wedge c) \vee (b \wedge x \wedge d)\end{array}$$

Maxwell and Kaplan (1989) observe that – though the number of disjunctions to process grows in rough proportion to the number of words in a sentence – most disjunctions are independent of each other. The general pattern is that disjunctions that arise from distinct parts of the sentence do not interact, as they are embedded within distinct parts of the f-structure. If disjunctions are independent, they conclude, it is in fact not necessary to explore all combinations of disjuncts as they are rendered in DNF, in order to determine the satisfiability of the entire constraint system.

On the basis of these observations, Maxwell and Kaplan (1989) devise an algorithm for contexted constraint satisfaction that reduces the problem of disjunctive constraint solving to the computationally much cheaper problem of conjunctive contexted constraint solving. The disjunctive constraint system is converted to a contexted conjunctive form (CF), a flat conjunction of implicational (contexted) facts,

$$\begin{array}{ll}\text{CF} & (a \vee b) \wedge x \wedge (c \vee d) \\ \Rightarrow & (p \rightarrow a) \wedge (\neg p \rightarrow b) \wedge x \wedge (q \rightarrow c) \wedge (\neg q \rightarrow d)\end{array}$$

[2]For generation this holds for a new generation algorithm, designed by John Maxwell and Hadar Shemtov.

based on the **Lemma:**
$\phi_1 \vee \phi_2$ is satisfiable iff $(p \rightarrow \phi_1) \wedge (\neg p \rightarrow \phi_2)$ is satisfiable, where p is a new propositional variable.

Context variables p, q and their negations are used to specifiy the requirement that for a disjunction of facts $\phi_1 \vee \phi_2$ at least one of the disjuncts is true.

Conversion to CF has the advantage that each fact appears only once, and is processed only once. The resulting formula is a flat conjunctive constraint system. To resolve this contexted constraint system, it is first turned into a normalized form that makes it easy to identify unsatisfiable combinations of constraints. The conversion builds on well-known algorithms for satisfaction of conjunctive systems. After detection of all unsatisfiable base constraints the resulting constraint system is checked for satisfiablity. For any unsatisfiable base constraint $P \rightarrow \phi$, $\neg P$ is called a *nogood*. The constraint system is satisfiable if the conjunction of all the nogoods, the *disjunctive residue*, is satisfiable.
If in the example above $a \wedge d$ is inconsistent, we derive an inconsistent base constraint in the normalized form: $p \wedge \neg q \rightarrow$ FALSE. $\neg(p \wedge \neg q)$ is thus a nogood. The disjunctive residue, here simply $\neg(p \wedge \neg q)$, is satisfiable in this constraint system for $p =$ F or $q =$ T.

While the first steps of the algorithm, conversion to CF and normalization of constraints are linear and polynomial, respectively, solving the disjunctive residue can be exponential. However, if the disjunctions are mostly independent, the residue breaks down into a number of independent problems, each of which is still exponential, but with much smaller exponents. The complexity will thus be closer to $k2^m$ than 2^n, where $m \ll n$. The performance experiments carried out in Maxwell and Kaplan (1993) confirm the observation that disjunctions are for the most part independent: contexted unification increases the overall performance in all interface architectures, as compared to (optimized) standard DNF-based unification.

After resolution of the disjunctive residue, the resulting constraint system is kept in conjunctive contexted form, i.e. in a *packed* representation format, where disjunctive facts are not compiled out and duplicated. In the packed f-structure representation local disjunctions are directly accessible through their context variables. This is illustrated in Fig.2, the *packed f-structure chart* for the ambiguous sentence *Unplug the power cord from the wall outlet.* The PP-attachment ambiguity is spelled out in the corresponding *unpacked* c- and f-structure pairs of Fig.3.
The ambiguity resides in the attachment of the PP as a VP- or NP-adjunct. While this ambiguity affects the entire c-to-f-structure mapping down from the level of VP, it is captured in terms of the local disjunctive contexts a_1 and a_2 in Fig.2. All remaining f-structure constraints are conjoined in the TRUE context.

$$\begin{array}{ll}a_1 & \rightarrow (f_2 \text{ ADJUNCT } \in) = f_{64} \\ a_2 & \rightarrow (f_{15} \text{ ADJUNCT } \in) = f_{64} \\ TRUE & \rightarrow (f_2 \text{ OBJ}) = f_{15} \ldots\end{array}$$

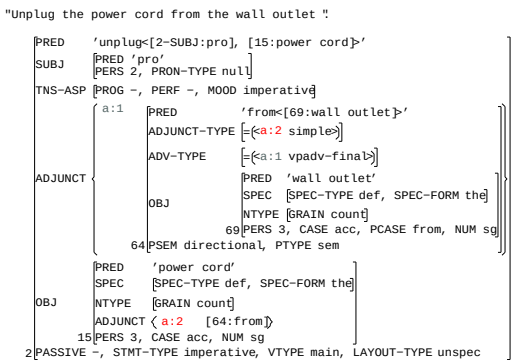

Figure 2: F-structure chart with disjunctive contexts $a1, a2$ for *Unplug the power cord from the wall outlet.*

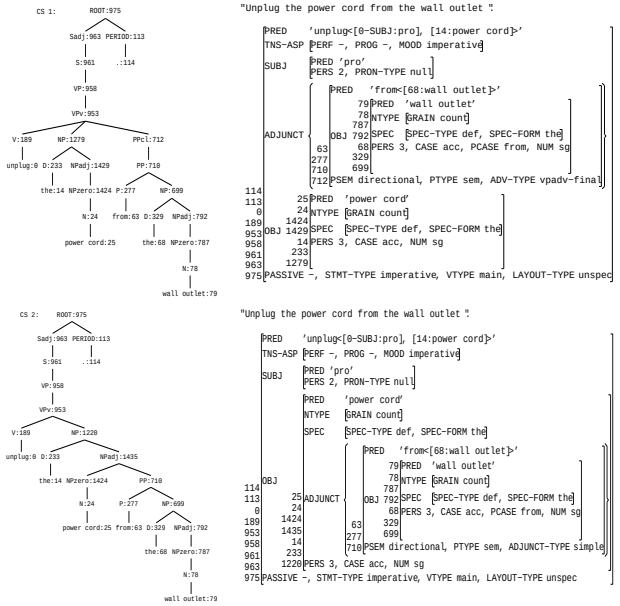

Figure 3: C-structure/f-structure ambiguity for *Unplug the power cord from the wall outlet.*

This compact representation can be transferred to subsequent processing modules, namely transfer and generation. Generation from packed structures in (Shemtov 1997) makes use of contexted constraints in the generation chart, to allow for efficient generation of all alternative expressions corresponding to an ambiguous input structure, without recurring to DNF, and thus enumerating and multiplying solutions.

4 MT as an LFG-based NLP application

Machine Translation is one of the most obvious multilingual NLP applications that can be built on top of the linguistic and computational resources that have been created in the ParGram and XLE projects. Other possible applications include (multilingual) information extraction, alignment components in translation memories, or multilingual authoring systems.

4.1 MT in the LFG Framework

Lexical-Functional Grammar has long been considered as a framework of linguistic analysis that is particularly well suited as an underlying theory for linguistically "informed" and modular MT architectures. Kaplan et al (1989) proposed an LFG architecture for transfer-based MT which takes advantage of LFG's division of linguistic representation into modular but related projection levels: the c-structure, f-structure, and optionally semantic (s-)structure. Source language representations are mapped to target language representations via a correspondence function, the τ-projection. This correspondence function can be defined as a mapping between source and target f-structures and/or semantic structures. Translation correspondences can thus be defined in a modular way, at various levels of linguistic abstraction.

Correspondence-based transfer has proven to allow for flexible and modular description of various typical translation phenomena. Due to the abstraction level of the f-structure representation, the description of transfer phenomena is largely independent from surface syntactic properties that vary across languages. Common transfer phenomena like grammatical function changes, and various kinds of structure changing transfer correspondences can be described in a modular way. Special kinds of structural mismatches however, in particular head-switching phenomena (*Hans schwimmt gerne – Hans likes swimming*), are difficult to define in terms of f-structure correspondences (Sadler and Thompson 1991). The treatment of these transfer phenomena is facilitated at the level of semantic representation, which further abstracts from structural differences between languages (Kaplan and Wedekind 1993). In this approach structural misalignment is dealt with in the syntax-semantics interface.

While in correspondence-based transfer a piecewise correspondence function maps a source structure node to a corresponding single target node, in transfer models based on term rewriting it is possible to relate a single source structure node to distinct target nodes. Transfer based on term rewriting has been proposed for syntactic and semantics-based transfer (Dorna and Emele 1996a; 1996b; Emele and Dorna 1998 and references therein).[3] The transfer component of Dorna and Emele is used in the VerbMobil MT project for transfer on underspecified semantic structures. The term rewriting algorithm operates on sets of terms of the semantic representation language.

Transfer on (underspecified) semantic structures has the obvious advantage that structural syntactic differences are neutralized in the more abstract se-

[3] For a comparison between f-structure- and semantics-based transfer in a term rewriting system see Dorna et al (1998). Closer comparison with correspondence-based f-structure transfer shows that term rewriting of f-structures is confronted with similar problems in treating head-switching. In both frameworks complex or multiple rules are needed to define appropriate target structures.

mantic representation. At the same time, it requires the definition of syntax-semantics interfaces both for parsing and generation. In the ParGram project we have chosen to explore translation on the basis of f-structures, where many – though not all – structural differences between languages are neutralized.

4.2 The Translation Architecture in XTE

The XLE system was extended to XTE (the Xerox Translation Environment) by addition of a transfer component, designed and implemented by Martin Kay. The underlying translation architecture is both traditional and innovative.

A source language string is parsed by the XLE system on the basis of rules and lexica of the source language LFG grammar. The transfer component rewrites the resulting source language f-structure into an underspecified target f-structure, by application of language-pair specific transfer rules. The target f-structure is input to XLE generation with the target LFG grammar, to produce target language strings.

While couched into a completely straightforward, traditional setup, the XTE translation architecture is innovative in generalizing contexted constraint processing and packed representation of ambiguities to all modules and interfaces of the translation chain.[4] As a major advantage of this processing scheme, ambiguities which arise in each of the processing modules – parsing, transfer, and generation – can be propagated forward within the translation chain.

Ambiguities that arise early in the translation chain will of course multiply whenever ambiguities arise in later processing modules. This leads to a computational complexity that is almost impossible to handle in conventional processing models. As a consequence, in most conventional translation architectures heuristic filters are applied early and throughout the processing chain – at the risk of pruning correct solutions too early, on the basis of poor evidence.

As we have seen, factored contexted constraint processing severely reduces the computational complexity in parsing and generation. The processing of contexted disjunctive constraints is now extended to the XTE transfer module, which operates on contexted, i.e. packed representations.[5] The output of transfer is a packed f-structure, which will be directly processed by a new generation algorithm which is designed to exploit the efficient XLE parsing and contexted constraint processing algorithms. Thus, while conventional systems are forced to resort to early pruning of ambiguities to avoid computational explosion, XTE's advanced processing technology allows to process ambiguities efficiently in a packed representation, in a uniform way, and in all processing modules. Since ambiguities can be carried along without harm, selections can be made flexibly, at various stages in the processing chain, whenever choices can be made on a justified basis, and with good evidence.

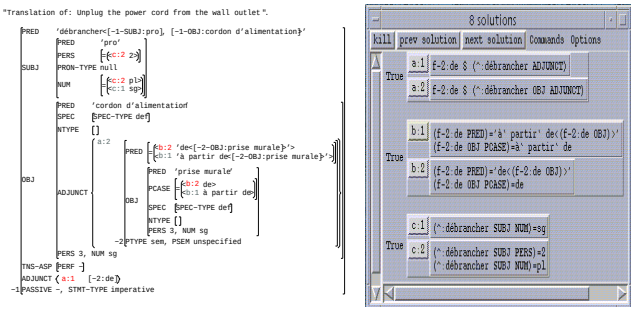

Figure 4: Ambiguity preserving translation – representation in chart and indexed by context variables

Moreover, transfer on packed representations of source language ambiguities allows for *ambiguity preserving* translation (Kay 1980, 1997) and (Shemtov 1997). Often an ambiguous sentence translates to a target sentence that displays the same ambiguity that is present in the source. As an example, reconsider Fig.2. The English sentence *Unplug the power cord from the wall outlet* can be translated into French as *Débranchez le cordon d'alimentation de la prise murale*, which displays the very same PP-attachment ambiguity that is present in the English sentence (see Fig.4).[6] With packed ambiguity processing in parsing, transfer and generation, this ambiguity can be carried over to the target without unfolding.

This translation model clearly follows the conception of Machine Translation advocated in Kay (1980, 1997). Machine Translation being a highly complex and poorly understood problem, a translation system mustn't take decisions which it is not well-prepared to take. The overall value of automatic translation is enhanced if such alternatives are left undecided. Ambiguities can be propagated towards the end of the translation chain, where examination on the output can provide useful hints for disambiguation. Moreover, the system must be designed in a flexible way, so as to allow for interactive guidance by a human. Interactive disambiguation can improve translation quality by avoiding chains of misguided decisions. Memory-based learning techniques can propagate human decisions for subsequent, similar decision problems. The translation architecture of XTE is designed for efficient propagation of ambiguities and *ambiguity management* (Shemtov 1997), and provides interfaces for interactive disambiguation (see below).

[4] In the current XLE implementation, generation requires unpacking of f-structures. See however (Shemtov 1997) for a sound generation algorithm for efficient generation from packed structures. Generation from packed f-structures is currently being implemented in XLE.

[5] See Dymetman and Tendeau (1998) for a variant of this approach. See also (Emele and Dorna 1998).

[6] Transfer introduces further ambiguities for preposition choice (*de/à partir de*) and morphological variants of the imperative. See below for disambiguation strategies.

4.3 The Transfer Component

The transfer component is a fairly general rewrite system that works on unordered sets of terms. In our application the terms represent f-structures, but the system lends itself to processing any kind of semantic (term) representation.[7]

In our translation scenario, the transfer component takes a packed f-structure as input, and delivers a packed f-structure as output. The attribute-value representation is first converted to a flat unordered set of f-structure terms. F-structure attributes with atomic values (f_1ATTR)=VAL are rewritten as `attr(var(1), val)`; attributes which take an f-structure node as value (f_1ATTR)=f_2 are rewritten as `attr(var(1), var(2))`. The f-structure terms are internally associated with their respective context variables.

The unordered set of terms is input to a cascade of rewrite rules that continuously rewrite subsets of (source language) f-structure terms into (target language) f-structure terms. The order in which the transfer rewrite rules are stated is crucial. Each rewrite rule applies to the current input set of terms, and yields an output set of terms. The output set constitutes the input for the next rewrite rule. A rule cannot reapply to its own output, but it applies to each distinct instantiation of the specified left-hand side terms that occur in the input set.

The left-hand side of rewrite rules specifies a set of terms `p`. If all these terms match a term in the input set, the matched terms are eliminated from the input, and the terms specified on the right-hand side of the rule are added to the input set. The left-hand side of a rule may contain positive `+p` and negative `-p` terms. A rule that specifies a positive constraint only applies if this term matches some term in the input. A rule that specifies a negative constraint only applies if the term doesn't match any term in the input. Positive terms are not eliminated from the input.

There are obligatory (`==>`) and optional (`?=>`) rules. Stated in an informal way, an obligatory rule that matches the input rewrites the left-hand side terms into the right-hand side terms. An optional rule that matches the input creates two output sets: in one output set the left-hand side terms are rewritten into the right-hand side terms, as in the application of an obligatory rule; the second output set is identical to the input set. Subsequent rules consider all alternative output sets created by preceding optional rules. The transfer component comes with a formalism that allows for a modular and generalized description of transfer patterns.

The operator (`&&`) unions two or more rewrite rules. If one of the unioned rules is an optional rule, the union will be an optional rule. If all of the rules are obligatory rules, the union is an obligatory rule.

[7] In much the same way as (Dorna and Emele 1996a)'s relational transfer system, as shown in (Dorna et al 1998).

Macros and templates provide means for stating hierarchies of recurring patterns of terms or rules. They can be (recursively) referenced in the definition of transfer rules. Templates define shorthands for optional, obligatory or unioned rewrite rules.

```
template_name(par1,par2):: lhs {==>|?=>} rhs.
```

Macros define shorthands for sets of terms and can be referenced in left- or right-hand sides of transfer rules, rule templates or in other macros.

```
null_pron(A):= pred(A,pro), pron_type(A,null).
```

Finally, left- or right-hand sides of transfer rules may state the empty set `0`. A rule `p ==> 0` with nonempty `p` deletes `p` from the input without introducing new terms in the output. Transfer rules with empty left-hand sides can be used in conjunction with rule unioning and redefinition of rule templates,[8] which allows for a compact definition of sequences of transfer rules. Below we first define two vacuous rule templates `restriction` and `opt`, the latter being optional. By union (`&&`) with these rule templates, the main template for verb transfer `v2v` is turned into an optional rule, which is called by the entry for *open*, to define transfer to *soulever*. Subsequent redefinition of `opt` as a vacuous obligatory rule effectively *redefines* the `v2v` template – with which `opt` is unioned – as an obligatory rule for subsequent template calls. *open* is thus alternatively transferred to French *ouvrir*. Finally, `restriction` – and thus `v2v` – is redefined to apply only in the absence of the term `obj`, in which case a macro for reflexive marking is called on the right-hand side. In this way we correctly transfer *appear* to French *s'afficher*.

```
restriction(A) :: 0 ==> 0.
opt:: 0 ?=> 0.
v2v(S,T):: pred(A,S), +vtype(A,_) ==> pred(A,T)
&& opt
&& restriction(A).
v2v(open,soulever).
opt :: 0 ==> 0.
v2v(open,ouvrir).
v2v(unplug,débrancher).
restriction(A) :: -obj(A,_) ==> refl(A).
v2v(appear,afficher).
```

4.4 A Transfer Grammar

With the extension of XLE to XTE, we built an experimental Translation Prototype that covers the entire translation chain, as a feasibility study for the newly designed transfer architecture, without aiming for large-scale coverage. A transfer grammar has been created for f-structure based transfer from English to French. As a corpus for translation we chose a text

[8] Templates and macros can be redefined at any point in the grammar. The new definition takes effect as soon as it is encountered. When a redefinition takes place, this causes an implicit redefinition of any other template or macro in whose definition it partakes, directly or indirectly.

from a technical domain, the user manual for the Xerox HomeCentre device. An arbitrary contiguous section of 99 sentences was selected from the corpus, the rationale being to ensure that a realistic collection of transfer problems would be encountered. We obtained correct translations for 94 sentences.[9]

Given the strictly order-sensitive nature of the rule rewriting system, a transfer grammar is globally structured into 3 major parts: A first set of rules defines various changes on the source f-structure, such as deletion of grammar specific features of the source f-structure, adjusting minor differences between source and target language features, as well as general transformations on f-structure encodings that facilitate the definition of transfer rules (e.g. set-valued `adjunct` features are converted to a set of atomic features).

The second part consists of genuine transfer rules, which continuously rewrite source language terms into target language terms. Internally, this part is again structured into a sequence of sections where transfer is defined for different parts of speech. The order of these sections is not arbitrary: for instance, transfer rules for verbs can be sensitive to nominal arguments or adjuncts in the source language. If these contextual constraints are stated in terms of source language predicates, transfer rules for verbs must precede transfer rules for nouns. Order restrictions can also occur within sections for the same part of speech.

The final part of the transfer grammar again performs general transformations on terms (e.g. converting atomic terms back to set-valued terms).

The prospects of parallel grammar development were confirmed in that the definition of transfer is clearly facilitated for many linguistic constructions, due to structural parallelism at the level of f-structure. The definition of transfer for standard syntactic constructions involving adverbials, negation, conjunctions, prepositions, adjectives, relative clauses, comparative clauses, etc. could be reduced to simple lexical transfer rules, while the (possibly complex) syntactic feature structures are left untouched as long as parallelism is preserved. This is illustrated by the following transfer rules. Due to uniform f-structure encodings for the specification of mood, sentence type, coordination and adjunct structures, etc., transfer of negation and conjunctions is covered by simple lexical rules that apply irrespective of the syntactic (f-structure) context, i.e. whether the material appears in declarative or imperative sentences, in relative clauses or conditional sentences. The adjective transfer rule, e.g., covers transfer of adjectives irrespective of their degree of comparison, which is specified by additional features that can be carried over to the target without changes. Even complex relative clauses can be transferred by simple lexical transfer rules for relative pronouns, the f-structures for relative clauses being specified in parallel across the grammars.

```
adv2adv(S,T):: pred(A,S) ==> pred(A,T).
   adv2adv(carefully,soigneusement).
   adv2adv(not, ne pas).
co2co(S,T):: conj_form(A,S) ==> conj_form(A,T).
   co2co(and, et).
   co2co(then, puis).
cj2cj(S,T):: comp_form(A,S) ==> comp_form(A,T).
   cj2cj(that, que).
   cj2cj(if, si).
a2a(S,T):: pred(A,S) ==> pred(A,T).
   a2a(good, bon).
+pr_type(A,rel), pr_form(A,that) ==> pr_form(A,qui).
```

There are of course transfer phenomena where source and target language exhibit distinct syntactic structures. Differences in argument structure or contextual restrictions on transfer are defined in a straightforward way in terms of lexicalized rule templates. More complex structural changes can be stated in a fairly modular way by exploiting a specific characteristics of the underlying transfer algorithm, the strictly ordered application of transfer rules which operate directly on the output of previous rule applications, as opposed to a rewrite scenario where the input set of terms is continuously rewritten into a *distinct* output set.[10]

This characteristics allows us to split up complex transfer definitions into a sequence of subsequent rules which define modular partial transformations in a stepwise fashion. Below we state the rule complex that defines nominalizations, such as *removing the print head – remplacement de la tête d'impression*.

The first rule performs lexical transfer of a verbal to a nominal predicate, jointly with a unioned (`&&`) rewrite operation that eliminates verbal and introduces appropriate nominal features (`nominal_to_verbal`). We further introduce the term `nominalized(A)`, as a handle, or trigger for the subsequent rules that will complete the nominalization transfer.

```
nominalization(SourceV,TargetN) ::
    pred(A,SourceV)
==> pred(A,TargetN), nominalized(A)
&&  verbal_to_nominal(A).
nominalization(replace, remplacement).
```

After lexical transfer, the argument structure of the originally verbal predicate is still unchanged. In nominalization, various kinds of relation changes occur, depending on the argument structure of the verb. A set of subsequent transfer rules defines these various relation changes. Below we state the rule for active transitive verbs, where the object of the lexical head is rewritten into a prepositional adjunct; the non-overt subject argument is deleted. The rules for relation changes are restricted to nominalization contexts by the constraint `+nominalized(A)`. In a subsequent, final rule this predicate is deleted from the set of terms.

```
      +nominalized(A), passive(A,-),
      obj_arg(A,B), subj_arg(A,C), null_pron(C)
==> adjunct_x(A,D), prepsem(de,D,B).
```

[9] The transfer grammar currently consists of 171 structural transfer rules and 76 lexicalized transfer rule templates with approximately 5 entries per template. The transfer lexicon is restricted to the chosen corpus.

[10] The latter conception is realized in the VerbMobil transfer component (Dorna and Emele 1996a).

Defining transfer in such a step-wise manner permits the statement of modular and more transparent transfer rules. There are cases where it is in fact impossible to state a single transfer rule for complex structural changes. A case in point are coordination structures where the relative scopes of coordination and an embedding conjunction are inverted in the target, as in: *You print a test page [when [you move the HomeCentre] or [replace an ink cartridge]] – Vous imprimez une page de test [[lorsque vous déplacez le HomeCentre] ou [(lors)que vous remplacez une cartouche d'encre]]*

In the source sentence, the coordination is embedded by a single conjunction *when*. In the target, coordination takes wide scope over the conjunction *lorsque*, which is reduplicated in each of the conjuncts. This structural relation corresponds to what is termed head-switching, but is more complicated in that it is combined with coordination, and thus can occur with an arbitrary number of conjuncts. While it is relatively easy to state a 'monolithic' transfer rule for this phenomenon, several variants would have to be defined to account for any possible number of conjuncts.[11]

Again, transfer can be defined in a more general way by a complex of subsequent transfer rules. The first rule rewrites the coordination conjunction `conj_form`, raising it to the higher level (node A) in the output. At the same time, the conjunction (`CompForm`=*when*) at node A is deleted from the input. However, the information about the lexical conjunction (`CompForm`), as well as the upper and lower nodes `A` and `Coord` is transmitted to the subsequent rule by means of the term `raise_conjuncts_to_compl`. The next rule creates, for *each* conjunct B of the node Coord (`element(B,Coord)`) a new conjunction structure `semconj(Bnew,CompForm,B)` that embeds the original conjunct referenced by B, and which is itself hooked into the raised coordination structure, at node A (`element(Bnew,A)`. Crucially, the second rule applies to *each* predicate `element` in the input set, raising it to A and inserting a conjunction *when* in the output. This is how the rule accounts for any arbitrary number of conjuncts. The two rules cannot be combined into a single rule, since the first rule is resource-sensitive to the occurrence of the *single* embedding conjunction (*when*) in the input. It could only be consumed once, thus not allowing for multiple application of the (combined) rule for an arbitrary number of conjuncts.

```
      semconj(A,CompForm,Coord),
      conj_form(Coord,ConjForm)
==>   conj_form(A,ConjForm),
      raise_conjuncts_to_compl(CompForm,Coord,A).

      +raise_conjuncts_to_compl(CompForm,Coord,A),
      element(B,Coord)
==>   element(Bnew,A), semconj(Bnew,CompForm,B).
      raise_conjuncts_to_compl(_,_,_) ==> 0.
```

The transfer algorithm realized in XTE's transfer component provides for a flexible way of encoding even complex structural changes in a modular and general way. The fact that any rule application changes the input for subsequent transfer rules requires a thorough organization of the transfer grammar.[12]

5 New Directions and Conclusion

The translation architecture and processing techniques realized in XTE constitute only a first, basic step towards a Machine Translation system. However, the system is designed in such a way as to allow for innovative extensions. The way in which extensions will be integrated into the overall system design, the way in which the system's characteristics are further exploited will be decisive for its overall value.

Ambiguity Preservation and Disambiguation: XTE's system architecture allows for a flexible design for ambiguity handling. Ambiguity preserving translation is inherently supported by the translation architecture. Propagation of ambiguities without filtering can be exploited in multilingual translation by triangulation (Kay 1980, Shemtov 1997) and for various techniques of ambiguity management (Shemtov 1997). Interfaces can be designed to allow for a flexible mixture of stochastic and interactive disambiguation, depending on specific applications and user needs. In the XTE prototype, a stochastic disambiguation model (Eisele 1999) assigns probabilistic weights to ambiguities present in source (and/or target) f-structures. Thresholds can be set for non-interactive n-best propagation of ambiguities. In interactive mode, ranked structures can be inspected by the user, to select f-structures for further processing. Ranked alternatives can be selected by reference to local ambiguities, indexed by their context variables (see Fig.4). This interactive model can be extended in various ways, e.g. to trigger user-intervention for predefined decision problems (which may presented in terms of stochastic ranking), and by integration of learning and propagation techniques for human-approved ambiguity resolution.

Acquisition of Transfer Knowledge: Techniques for automatic acquisition of transfer lexica from bilingual corpora were proposed e.g. by Turcato (1998). His approach can be generalized to packed f-structure processing, and seamlessly integrated within the XTE translation architecture. Extensions towards alignment models as proposed by Grishman (1994) can be exploited for automatic acquisition of transfer rules.

[11] Such a rule references the embedding conjunction and the coordination structure in the source, and inverts their structural relation in the target. However, the single embedding conjunction has to be reintroduced within *each* of the conjuncts of the 'raised' coordination structure. This can only be done by enumerating the correct number of conjuncts in the target. Seperate rules are thus required to account for a variable number of conjuncts.

[12] Obvious complications like translation cycles are dealt with in a straightforward way, by source and target language marking of lexical predicates.

Statistical Methods and Robustness: Further extensions are required for transforming an MT prototype into a powerful and robust large-scale MT system. Knowledge-, or rule-based systems are not well-prepared to process unseen data not captured by grammar or transfer rules. Interfacing statistical processing models with rule-based systems is a challenge worth to explore. Corpus-driven stochastic parsing models in the LFG framework (Bod and Kaplan 1998), with possible extensions towards transfer architectures (Way 1998) take first steps into this direction.

Acknowledgements

The author gave a summarization of the research conducted by the project members Marc Dymetman, Andreas Eisele, Anette Frank, Ron Kaplan, Martin Kay, John Maxwell, Paula Newman, Hadar Shemtov, Annie Zaenen, and the grammar writer team. We are grateful for helpful comments and suggestions from our colleagues. Remaining errors are our own.

References

Bod, R. and Kaplan, R. 1998. A Probabilistic Corpus-Driven Model for Lexical-Functional Analysis. in *Proceedings COLING/ACL'98*, Canada.

Bresnan, J. (ed) 1982. *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Butt, M., King, T.H., Niño, M.E. and Segond, F. 1999. *A Grammar Writer's Cookbook*. Stanford, CSLI Publications.

Dalrymple, M., Kaplan R.M., Maxwell III, J.T. and Zaenen, A. 1995. *Formal Issues in Lexical-Functional Grammar*. Stanford, CSLI Publications.

Dorna, M. and Emele, M.C. 1996a. Efficient Implementation of a Semantic-based Transfer Approach. *ECAI'96*, Budapest, Hungary.

Dorna, M. and Emele, M.C. 1996b. Semantic-based Transfer. *Coling'96*, Copenhagen, Denmark.

Dorna, M., Frank, A., van Genabith, J. and Emele, M.C. 1998. Syntactic and Semantic Transfer with F-Structures. In *Proceedings of COLING'98*, Canada.

Dymetman, M. and Tendeau, F. 1998. An Algorithm for the Transfer of Packed Linguistic Structures. Unreleased internal Xerox publication.

Eisele, A. 1999. Representation and stochastic resolution of ambiguity in constraint-based parsing. Doctoral Thesis, University of Stuttgart.

Emele, M.C. and Dorna, M. 1998. Ambiguity Preserving Machine Translation using Packed Representations. In *Proceedings of COLING'98*, Canada.

Frank, A., King, T.H., Kuhn, J. and Maxwell III, J.T. 1998. Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. in: Butt, M. and T.H. King (eds): *Proceedings of the LFG98 Conference*, University of Queensland, Brisbane, CSLI Online Publications.

Grishman, R. 1994. Iterative Alignment of Syntactic Structures for a Bilingual Corpus. in *Workshop on Very Large Corpora 1994*.

Kaplan, R. and Maxwell III,J.T. 1996. LFG grammar writer's workbench. Technical Report. Xerox Parc.

Kaplan, R., Netter, K., Wedekind, J, and Zaenen, A. 1989. Translation by Structural Correspondences. *EACL'89*, Manchester, UK, 272–281.

Kaplan, R. and Newman, P. 1997. Lexical resource reconciliation in the Xerox Linguistic Environment. in Estival, D., Lavelli, A., Netter, K., and Pianesi, F. (eds) *Computational environments for grammar development and linguistic engineering*. Proceedings of a workshop sponsored by ACL, Madrid, Spain, 54–61.

Kaplan, R. and Wedekind, J. 1993. Restriction and Correspondance-based Translation. *EACL'93*, Utrecht, The Netherlands, 193–202.

Kay, M. 1980. The Proper Place of Men and Machines in Language Translation. Xerox PARC Working Paper. Published in *Machine Translation*, 12, 1997, Kluwer, Netherlands, 3–23.

Kay, M. 1997. It's Still the Proper Place. In *Machine Translation*, 12, 1997, Kluwer, Netherlands, 35–38.

Kuhn, J., Eckle-Kohler, J. and Rohrer, C. 1998. Lexicon Acquisition with and for Symbolic NLP-Systems – a Bootstrapping Approach. in *Proceedings of the First International Conference on Language Resources and Evaluation (LREC98)*. Granada, Spain, 89–95.

Maxwell III, J.T. and Kaplan, R.M. 1989. An overview of disjunctive constraint satisfaction. In *Proceedings of the International Workshop on Parsing Technologies*, 18–27. (Also published as: A method for disjunctive constraint satisfaction. in: Tomita, M. (ed): *Current Issues in Parsing Technology*, Kluwer Academic Publishers, 1991).

Maxwell III, J.T. and Kaplan, R.M. 1993. The interface between phrasal and functional constraints. In *Computational Linguistics*, 19(4):571–590.

Maxwell III, J.T. and Kaplan, R. 1996. Unification-based Parsers that Automatically Take Advantage of Context Freeness. Paper presented at the *LFG96 Conference*, Grenoble, France. Ms. Xerox PARC.

Sadler, L. and Thompson, H.S. 1991. Structural Non-correspondence in Translation. *EACL'91*, Berlin, Germany, 293–298.

Shemtov, H. 1997. *Ambiguity Management in Natural Language Generation.* Doctoral dissertation, Stanford University.

Turcato, D. 1998. Automatically Creating Bilingual Lexicons for Machine Translation from Bilingual Text, in *Proceedings of COLING'98*, Canada.

Way, A. 1999. A Hybrid Architecture for Robust MT using LFG-DOP. To appear in *Journal of Experimental and Theoretical Artificial Intelligence*.