A Typology of Translation Problems for Eurotra Translation Machines[1]

Ian Crookston[2] Jane Shelton[3] Andrew Way[4]

[1] **Acknowledgements**: We trust that the many erstwhile colleagues in the Eurotra project who contributed to the development of our ideas on the subject of translation problems will not object if we single out for specific mention Doug Arnold and Louisa Sadler. Finally, our considerable thanks to five anonymous reviewers whose comments and insightful discussion did much to shape the final article. Any remaining errors or omissions are of course our own fault.

[2] Address: Faculty of Speech & Language Sciences, Leeds Metropolitan University, UK

[3] Address: School of Modern Languages, University of Newcastle, UK

[4] Address: School of Computer Applications, Dublin City University, Ireland

Abstract

Keywords: 'Hard' Translation Problems, their Interaction, Consequences for Translation Notations.

This paper presents a detailed study of Eurotra Machine Translation engines, namely the mainstream Eurotra software known as the E-Framework, and two 'unofficial' spin-offs–the ⟨C,A⟩,T and Relaxed Compositionality translator notations–with regard to how these systems handle "hard" cases, and in particular their ability to handle combinations of such problems. The justification for this approach is that, at the end of the day, *any* system will be judged overridingly on one detail: how it handles the data.

Our findings present themselves as follows:

- In the ⟨C,A⟩,T translator notation, some cases of complex transfer are "wild", meaning roughly that they interact badly when present with other complex cases in the same sentence. For instance, the case *savoir* $\longrightarrow$ *know how* interacts badly with any other complex case. The effect of this is that each combination of a wild case and another complex case needs ad hoc treatment.

- The E-Framework is the same as the ⟨C,A⟩,T notation in this respect. In general, the E-Framework is equivalent to the ⟨C,A⟩,T notation for the task of transfer (this point is expanded upon in Crookston (1990)).

- The Relaxed Compositionality translator notation is able to handle each wild case with a single rule even where it appears in the same sentence as other complex cases.

0.1 The Translator Notation Problem

This paper is an extended discussion of one of the central problems of Machine Translation (MT), namely: given that translationally equivalent expressions may differ, what machine, that is, what notation, is necessary to create a target language representation on the basis of a source language one? We call this the translator notation problem. We examine this problem particularly in the context of a multilingual MT system, where for reasons of modularity (as will be discussed later), certain shortcuts along the lines of "tuning" languages towards eachother are not available. Other central and pressing problems of MT, such as the contribution of contextual and background knowledge, are not our concern here. We feel that any translation machine, no matter how sophisticated pragmatically, will need a linguistic undercarriage, because such phenomena as syntax are real, do differ from language to language, and are different in kind from contextual and background knowledge. This paper offers a discussion of some problems this linguistic undercarriage is faced with at the point where one language meets another. As will be seen, the problems are far from trivial.

It may well be rational to hope, as many do, that the translator notation problem will eventually go away; that when analysed deeply enough and in the right spirit, each language will turn out to be so similar to every other that "simple transfer" (essentially, word-for-word translation) will be all that is required. We submit that it is equally rational to hope that the translator notation problem may be solved, and perhaps in the present state of knowledge the latter hope is even a little more likely to be fulfilled, given the large distance we are currently away from the goal of simple transfer.

In this section, we outline the design decisions which lead to the existence of a translator notation problem, together with the reasons for them. We go on to document some of the manifestations of this problem in the literature at large before introducing Eurotra's main translator notations. Following this we outline a series of increasingly difficult translation problems. The next two sections then test these notations against this range of translational data: centrally, we show that the CAT and E-Framework (EF) notations faced with certain problems have a tendency to combinatorial explosion which is partly avoided by the Relaxed Compositionality (RC) notation. The significance of this is that CAT and EF have certain features which would appear in anyone's first approximation to a translator notation, based as they are on the local subtree of a linguistic representation, and it is these very features which cause some of the problems. Second approximations like RC can therefore be assessed as to how they address the problems revealed.

The discussion below often delves into a level of detail about translation problems which is usually avoided in the MT literature. We find the paucity of such detail regrettable, because it is often only the consideration of detail in these areas that reveals the existence and nature of problems: we therefore hope that this paper will provide a more solid foundation of detail than is usually

available about an MT notation.

0.2 The Translator Notation Problem

The most basic assumption that work in the Eurotra project has always made about machine translation (MT) is this:

(I) Translation is performed between linguistic representations.

This assumption has been argued for in the Eurotra literature, and the arguments and other important background can be found in the references cited at (0.4) below (see also Johnson (1987)): it is not necessary to repeat in detail the arguments supporting (I) here. The Eurotra approach is not of course uncontroversial in the perspective of the whole MT field: see Slocum (1985) and Hutchins (1986) for balanced surveys of the field[1].

Neither is (I) unique to Eurotra: rather, it places Eurotra in a family of current approaches to MT, some other exemplars of which will be mentioned below. All it means is that Eurotra differs from a "knowledge-based" approach, and from any other where the translation of a text passes through representations of its content in some notation that is independent of language[2].

(I) is usually held to offer two basic possibilities for the general outline or architecture of the translation process. The first is:

(A) 1 Source text TS $\longrightarrow$ Representation of source text RS $\longrightarrow$ Representation of target text RT $\longrightarrow$ Target text TT

This is known as the "transfer" architecture, and TS $\longrightarrow$ RS is usually known as the analysis mapping, RS $\longrightarrow$ RT the transfer mapping, RT $\longrightarrow$ TT the synthesis or generation mapping. Contrast this with the "interlingual" architecture:

[1] For the most part, Eurotra "linguistic" representations are "syntactic" in nature, although its treatment of such phenomena as Tense and Aspect, and Diathesis, are undoubtedly more semantic. We shall continue to use this potentially misleading terminology throughout the paper.

[2] An interesting though isolated counterexample to (I) encountered in the present study is *au fur et à mesure*. Dictionaries give examples such as

Je lui paie au fur et à mesure $\Longrightarrow$ I'm paying him as fast as the work is done
On les emballe au fur et à mesure $\Longrightarrow$ They are packed as fast as they are ready

It is clear that the translation is *as fast as X* (or perhaps *at the rate that X*) where X can be completely undetermined by the source linguistic material, having sometimes to be created by the translator on the basis of completely non-linguistic knowledge. This counterexample appears to be insuperable. It is different in kind from most disambiguation problems, where the source linguistic material offers a narrow choice of target equivalents which might need to be made from non-linguistic knowledge. Here what has to be done from non-linguistic knowledge is creatively fill a complete blank in the target sentence.

(B) 2 Source text $\longrightarrow$ interlingual representation of text $\longrightarrow$ Target text

The interlingual representation in such an approach is completely neutral between the two languages, unlike RS and RT in (A). Of these Eurotra chooses the transfer architecture, again not uncontroversially, and, again, arguments are to be found in the references given at (0.4).

(A) is further elaborated in Eurotra, as commonly in this family, in the following way. The analysis and synthesis mappings are broken up into stages, with intermediate representations:

(C) 3

$$\begin{array}{lll} & \text{TS} \longrightarrow \text{RS1} \longrightarrow \text{RS2} \longrightarrow \ldots \longrightarrow & \text{RSn} \\ & & \downarrow \\ & \text{TT} \longleftarrow \text{RTn} \longleftarrow \ldots \longleftarrow \text{RT2} \longleftarrow & \text{RT1} \end{array}$$

RSn and RT1 are known as the "interface" representations (or IS's for "interface structures") of the source and target languages respectively. A good presentation of the arguments for this kind of staging of translation is given in Leermakers & Rous (1986).

A further assumption that Eurotra makes puts it in a small subfamily within the larger family which adopts (I), (A) and (C):

(II) Translation does not pass through any representation which is not in every respect a representation of either the source or the target language.

What (II) says is that IS's cannot be "tuned" to each other, i.e. they cannot be adjusted on bilingual criteria so as to make the process of mapping between them simpler. Every aspect of RS must be justified on the basis that it is an appropriate *monolingual* representation of TS within the wider grammar of the source language, and similarly for RT in the target language.

Many MT systems do not observe this restriction. One of the clearest examples in the literature is one which claims to observe it (cf claim (d) in Guilbaud (1987, p.278)) but does not (cf the RT in op cit, p.309, which has French words in German surface word order). Another major example is the Mu system (Nagao & Tsujii (1986)). This system explicitly divides the mapping between the two IS's into three stages, and thus passes through representations which are mixed source and target representations. For example, in the first of the three phases, a representation of the Japanese sentence that means roughly *That John goes is a tendency* is converted into a representation that might be glossed *John goes*, annotated for "tendency modality". It eventually becomes the English *John tends to go* (ibid:103). The intermediate representation would not be proposed as a possible representation of the Japanese sentence per se: it is a representation of Japanese skewed towards English. But these are mentioned here simply as examples of a large class of systems which fall outside the subfamily we are interested in here.

The justification for (II) is given in Krauwer & des Tombe (1984). Essentially, in a nine-language system such as (to take an example at random) Eurotra, if a source IS were to be tuned to its eight targets, maintenance of the mechanisms governing analysis and generation would become impossible. It would come to require mastery of the linguistics of all nine languages, the source or target plus all of those to which one endpoint of the mapping was being tuned. Perhaps the best known system which advocates tuning is ROSETTA, which is an interlingual-based system designed to translate between English, Dutch and Spanish. The authors of this system attempt to refute these arguments against tuning of grammars as follows (Rosetta 1994, pp. 97-99):

> "(Such) arguments are based on the assumption that tuning requires a large extra effort that will lead to unnatural grammars from a monolingual point of view. Experience, however, shows proof to the contrary ... Notwithstanding, 'relics' of one grammar do occur in another grammar ... However, in our experience the frequency of these phenomena is low for Romance and Germanic languages ... so the extra effort needed for tuning is relatively small ... The endeavour of attuning grammars is in fact very challenging from a theoretical point of view ... (as) it stimulates the grammar writers to look for properties that languages have in common and to develop a common view on grammars."

On this latter point, the properties that languages have in common tend to be relatively unproblematic for the purposes of translation; it is where languages *differ* that causes the problems. Despite this, the ROSETTA methodology forces its grammar writers to introduce unnatural rules into what are deemed monolingual modules to cope with language differences, rather than deal with these in a separate, translation module. The defence of this position culminates in statements such as the following (op cit, p.96):

> "It is suspected that the trouble of developing interface structures with the objective to reach simple transfer, but often still having to write complex transfer rules, is at least comparable to the extra effort needed to make compositional grammars isomporhic."

although they admit to having no evidence that this is indeed the case.

One further criticism of tuning is that systems which employ such a strategy will be difficult to extend with further languages, in that if these languages make distinctions which were not previously made in the grammars, then these will impinge on those grammars and cause them to be rewritten. ROSETTA's authors expect that (op cit, p.98):

> "adding Germanic and Romance languages to the ROSETTA system ... will mean only a few additions on the structural level. But

> if this turns out to be wrong, the alternative as presented in Landsbergen (1989) is possible: isomorphic grammars with transfer parts, in which a clear distinction is made for each grammar between the general aspects and the parts that are inserted because of translation into or from a particular language."

They admit, however, that this solution is "of course, the philosophy behind transfer systems" (ibid). The point remains, of course, that such a solution is the raison d'être of a transfer system; here, the insertion of translation-specific rules into monolingual components, whilst perhaps leading to the resolution of translation problems, continues to pollute the system, and cast doubt on the chosen methodology.

One consequence of (II) forms the subject matter for this paper. (II) creates a problem, which we shall call the translator notation problem. The representations in (C) are licensed, being formal linguistic representations like any other, by grammars written in "grammar notations". "Notation" here is intended to include both the notation proper and its interpretation. The notation proper is a "syntax" of grammatical symbols and their combinatorial possibilities, and the interpretation a "semantics" giving what that syntax means, but these terms will be avoided as being potentially confusing. The translator notation problem is this: what notation (including associated interpretation) is necessary to express the mapping between translationally equivalent representations? We have a source IS of a given sentence, and an IS of its translational equivalent in the target language, and the two are rigidly kept apart by assumption (II). What notation is needed to relate them explicitly, so that a computer could create one on the basis of the other?

This problem emerges wherever assumptions (I) and (II) are adopted. Where (II), in particular, is not adopted, the translator notation is not a problem. One can use any arbitrary translator notation, and "tune" the IS's until they are "close enough" for the chosen translator notation to work. One nice clear example of this is the source IS in Maas (1987, p.235), containing a representation of the German sentence *Ich arbeite gerne*. The representation has *gerne* as its highest governor, and the rest of the sentence as a clausal object of *gerne*. A relatively simple translator notation can then relate this representation to, say, the English representation of *I like working*, where *like* is indeed the highest governor, and *working* a clausal object of it. But this German representation is not a representation "of German" in the sense relevant to (II). It is a mixed representation of German words in a configuration appropriate for certain target languages: in other words, it is the non-adoption of (II) which makes it possible to adopt a simple translator notation and bypass the translator notation as a serious problem. And we feel that the non-adoption of (II) is not a live option for a large multilingual project[3].

[3]However, someone from a more semantic background might say that Maas' representation is in every respect a legitimate representation of German, namely its logical form. Given

If (II) is followed, *gerne* in a German IS must be represented in the same way as German adverbs in general (or in the same way as some German-internal subclass of adverbs) and will not take clausal objects. The mapping from the IS of *Ich arbeite gerne* to that of *I like working* will then become a "hard case" of transfer for the translator notation to solve. As stated, this paper will investigate the Eurotra translator notations, and the chosen methodology will be to identify "hard cases" of transfer and to examine how well they map such cases. This amounts, of course, to nothing more original than testing the theory of transfer against the data of transfer. However, taking into account the ability of a system to handle the data, together with the complexity of a notation's T-rules in attempting this task, provides a simple, effective evaluation metric against which to judge competing proposals, as well as providing a level of abstraction at which linguists and implementors can communicate. However, perhaps even more important is that this decomposition of the problem of MT provides a framework for the investigation of interesting, manageable and not so manageable problems.

Relevant terminology includes the notions of "simple" and "complex" transfer. Transfer is "simple" when the two IS's related by it differ only in their lexical units. All the translator notation has to do in that case is replace lexical units. Transfer is "complex" when it is not simple. Complex transfer examples are to differing degrees "hard cases" useful in investigating translator notations.

To summarize, it should be stressed that the approach taken here is limited to (essentially) syntactic structural transfer. Notwithstanding the renewed interest in semantic representations as a possibility for avoiding complex structural transfer, as well as approaches which postpone the solution of such problems to the generation stage, we exclude any comparison here on the grounds that it has yet to be shown that such "solutions" to the problems we discuss scale up to the size of the task faced in Eurotra. In that respect, it should be noted that what the Eurotra systems amount to are constrained formalisms, utilising a low-level syntactic representation and a one-shot transfer methodology. Admittedly, many of the problems discussed below result from the adoption of such choices. Given this, whilst our discussion is of direct import to systems which expound a similar viewpoint, we feel that any other type of system should be capable of handling the range of phenomena, for the range of languages, that we show RC to be capable of. At the same time, any system incapable of achieving the level of translation success of CAT, given its simplicity, must be called into question.

Finally, one final restriction that the Eurotra approach makes is in the generation of *one* translation, unless the input sentence is itself ambiguous. Conse-

---

the relative absence of semantics in the Eurotra formalisms, we feel justified in maintaining the claim that Maas' representation nevertheless violates axiom (II), although we remain open to the criticism that what we are arguing for is more for hybrid representations of monolingual syntax and semantics, rather than against hybrid representations of source and target languages.

quently, we do not discuss at all here the notion of valid alternative translations which demand rankings in desirability. We acknowledge that this is in direct contradiction to translation theory per se, where there is (normally) no such things as *the* translation, let alone the *best* translation, as the merit of different translations will depend on various criteria.

0.3 The problem outside Eurotra

Although this paper has its roots in the translator notation problem as Eurotra attempts to solve it, it is worth looking at other systems that have encountered it. One problem in attempting this is the paucity of the relevant details in published sources, which renders any conclusions drawn very tentative.

Describing the TAUM-AVIATION English-French system, Isabelle (1987, p.251) says "both languages share the same set of context-free 'base rules' but have different lexicons". This means that RT and RS, lexical items apart, are governed by the same grammar. However, this does not mean that translational equivalents will have the same structure. The grammar will generate *Le directeur aime l'idée* but that does not mean that the latter is the desired translation of *The director likes the idea*. So, Isabelle says, "lexical transfer cannot simply substitute lexical items, leaving the tree structure unaffected" (op cit, p.255). In other words assumption (II) is held in this system—the tree structures of translational equivalents cannot be tuned to each other. Isabelle (op cit, p.256) gives such examples of complex transfer as:

(1) a. supply x with y $\Longrightarrow$ fournir y à x

b. reinstall x $\Longrightarrow$ remettre x en place

c. service x $\Longrightarrow$ faire l'entretien de x

but unfortunately gives very little detail of LEXTRA, the translator notation that addresses these problems in TAUM-AVIATION.

Sanfilippo et al. (1992) advocate a "strongly lexicalist treatment of translation equivalence" based on capturing mismatches due to diverging lexicalisation equivalence via translation links (*tlinks*). While this is fine for the chosen examples of verbs of movement, it seems not to work for more complex examples which involve having to "rearrange grammatical dependencies involving constituents other than those included in the same lexical equivalence ..." (headswitching, say, or examples such as (22)), "...for the *tlink* mechanism is not equipped to express structural equivalences which arch over lexical entries involved in distinct *tlinks*" (op cit, p.9). As we will show, numerous examples of just this type of thing exist, so any system which cannot deal with such cases must be called into question.

Sanfilippo et al.'s suggested 'solutions' to this problem would result in the yielding of "equivalences between lexically governed syntactic rules which are

unmotivated from the perspective of monolingual grammars" (ibid), and are thus rejected. Finally, they plump for a "generation process which does not rely on transfer of the (source) parse tree" (ibid), e.g. 'Shake and Bake MT', as a solution, although they state that "the eventual success and applicability of (this) approach ... rests on the development of computationally efficient versions of the Shake and Bake generation algorithm" (op cit, p.10), the likelihood of which is alluded to below.

With the CRITTER system, an intellectual descendant of the TAUM systems, Isabelle et al. (1988) give three specimens of the notation proper (op cit, p.265):

(2) a. eat $\leftrightarrow$ manger

b. miss(1: X, 2: Y) $\leftrightarrow$ manquer(1: Y′, 2: X′)

c. walk(inv-1: across(2: X)) $\leftrightarrow$
traverser(2: X′, inv-1: $manner(2: apied))

The interpretation of this notation is not given, and although one may speculate as a third party, it would be preferable to read the authors' account first-hand. Take (2)(c): "inv-", for instance, must mean something, but it is is left open to interpretation[4]; and similarly with the unexpected order of elements within the brackets on the *traverser* side. As it stands, it is impossible to evaluate thoroughly this notation with any certainty; all that can be said for certain is that the translator notation problem has arisen in CRITTER and has been addressed.

Estival et al. (1990) introduce "ELU", a general purpose grammar writing tool based on PATR-II, to which they add a transfer component in order to adapt the system to MT. They regard transfer as a relation between feature structures, and "require the transfer component to ... (have) the same declarative semantics as the other components of the system" (op cit, p.131). However, they later point out that "the system behaves differently when performing analysis/generation and when performing transfer" (op cit, p.142) in that, unlike in transfer, "the strategies used for analysis and for generation do not prevent the application of a general rule when a more specific one succeeds" (ibid). Hence, whilst their system will produce the correct, specific translation of *grey horse* as *Schimmel* (rather than *graues Pferd*), it would (presumably) produce two (or more) translations of ambiguous source sentences, one using the specific analysis rule and one by the default rules (as we will see, like CAT and EF, and unlike RC). They put this down to their making "a strong claim about translation, namely that unlike in the grammars, in the transfer component, more specific

[4]Presumably, if one side of an equation denoting an object L includes inv-N:X, then X denotes an object which includes N:L. All modifiers would be functors over their heads, resulting in the construction of a categorial-style functor argument tree. Of course, even if this interpretation is the correct one, it still says nothing about whether this is a useful way of describing the problem, nor whether it can be efficiently applied.

rules override less specific ones. This may prove to be too strong, and may have to be relaxed" (ibid). It would appear that such a relaxation may indeed be preferable, and given that the facility is built into the system, it should be availed of where required.

A further anomaly is found where they state that "rule bidirectionality is an *essential* property of the formalism" (op cit, p.132), but then almost immediately note that "it is not *necessary* to assume the bidirectionality hypothesis in an actual system built with ELU" (ibid) (our emphasis). Nevertheless, the detail shown in this approach is admirable, and a number of cases are worked through, including simple transfer, headswitching and shoehorning examples.

The linguistics formalism LFG (Kaplan & Bresnan, (1982)) has been proposed as an MT formalism (Kaplan et al., (1989)). LFG is a very powerful and elegant way of relating unlike representations when used as a pure-linguistics theory of syntax, and so its power in MT is worth investigating. Kaplan et al. illustrate the ability of LFG to cope with some "hard" examples using "codescription" (notably *venir de* $X \Longrightarrow$ has just X-ed). However, Sadler et al. (1989) and (1990) show that there are a number of cases which LFG cannot handle, some of which are solved by the introduction of the notion of "restriction" (Kaplan & Wedekind, 1993).

Zajac (1990) points out that is is difficult to see the approach of Kaplan et al. (1989) as transfer-based (op cit, p.246):

> "... since the grammar of the source language and the contrastive descriptions are tightly integrated. In fact, there is only one grammar which merges what is traditionally divided into source anguage grammar and contrastive transfer grammar, and there is only one lexicon. Such an integration makes any change or adaptation of the grammar difficult."

Zajac also notes problems with the extensibility of this approach (op cit, p.247):

> "Furthermore, for changing the target language or adding a new one, the only feasible solution would be to remove transfer information for the first language and extract the source grammar, and add co-descriptions for the second language. Thus, using the co-description approach, the modularity that could theoretically be achieved is lost in the actual implementation."

Zajac's work is to be found in the POLYGLOSS project (op cit). Here he reformulates the notion of codescription in a relational framework, leading to modular, reversible grammars. Zajac uses the same *venir de* $X \Longrightarrow$ has just X-ed example using a logic system using feature terms as basic data structures, embedded within a powerful typing mechanism.

Perhaps the first approach using LFG was the "Lexical-Functional Transfer" approach of Kudo & Nomura (1986), whose translator notation is closely based

on LFG. However, Kudo and Nomura do not make any claims about the ability of their system to do "hard cases" of transfer such as those addressed by TAUM-AVIATION, restricting their exemplification to an easy case.

The problem has also been addressed in "Shake & Bake" approaches to MT (Beaven 1992, Whitelock 1992, 1994). Whitelock's (1994) notational approach models itself on Unificational Categorial Grammar (Zeevat et al., 1987), thereby incorporating orthographical, syntactic and semantic information into signs (lexical entries). Examples of simple transfer, argument switching, headswitching and verbs of motion are illustrated in some detail in order to substantiate the claim that the "Shake & Bake" approach succeeds where structure-based methods do not. We will not go into this at length here, but merely quote Beaven himself (Beaven 1992) that "the process of Shake and Bake generation is NP-complete", and as such, it would appear that other methods (e.g. dynamic programming) seem to be necessary to "avoid the combinatoric explosion of Shake & Bake translation" (Alshawi, 1996).

Mention must finally be made of the ROSETTA approach (Landsbergen (1987a, 1987b); Rosetta 1994). This system is not within the family of systems that adopts (II), with its explicit acceptance of "attuned" grammars (Landsbergen 1987b, pp.352f). An example of this in ROSETTA is where English and Dutch first person pronouns are forced to bear a marking for gender, as in (Van Eynde, 1993, p. 24):

(3) a. EN: I (masc.) am ill $\longrightarrow$ ES: estoy enfermo

b. EN: I (fem.) am ill $\longrightarrow$ ES: estoy enferma

Given this, as well as the fact that ROSETTA is not a transfer system at all, having no close equivalent of the RS-RT mapping, why then consider it in this context at all? Because the same concern about the hard cases of translation, almost always including the Dutch adverb *graag*, emerges repeatedly in both the ROSETTA and the Eurotra literatures (see Landsbergen (1987b, pp.368-371), Appelo et al. (1987), Odijk (1989), Appelo (1993) and the Eurotra references given at (0.4) below). Not having a transfer mapping, ROSETTA faces no translator notation problem as such. However, it illustrates the tendency for MT problems, when squeezed out of one part of the scheme of things, to re-emerge in a different form in another.

In ROSETTA, the central concept is not really the representation but the derivation tree, in the Montagovian sense of a record of what rules are applied to what units in what order to generate a representation. Such rules might be "unit-building" rules, the addition of subject and object to a verb, say; or transformational-type rules altering the structure already built. Translational equivalence is specified by stipulating that two sentences are possible translations of each other when corresponding rules appear in the derivation tree of each at corresponding points. When the Dutch subject is added to the verb, then the English subject must be added to its verb, and so on. Given even this very brief outline, it is easy to see what will happen when "hard cases" of translation arise. If we take (1)(b) above, perhaps there will be an English rule adding subject and object to *reinstall*. This rule must presumably be made to correspond to a French rule adding subject, object, and *en place* to *remettre*. The hard cases of transfer reappear as hard cases of rule correspondence. Odijk (1989) is in fact dedicated to a discussion of such hard cases (though not including the *reinstall* type).

The major conclusion to be drawn from current literature on the translator notation problem is that it is no spurious problem. Many people who adopt (I) and (II) seem to find translator notations to be an issue. In addition, ROSETTA, approaching the MT problem from a very different direction, finds a corresponding problem. The translator notation problem, ROSETTA's rule correspondence cases, and the alternative practice of "tuning" RS and RT to avoid the translator notation problem, all appear to be reflexes of a fundamental fact about translation, namely that translational equivalents differ in certain awkward ways. It would be worthwhile to make a study of all translator notations, testing them against translation data, and arriving at well-supported conclusions as to strengths and weaknesses. This paper aims to begin that process by doing exactly that to the three main translator notations that have been proposed in Eurotra.

0.4 Eurotra's Translator Notations

The Eurotra literature has addressed the translator notation problem at length. There have been three main notations for the whole MT process proposed in Eurotra, and each of them has had its own translator notation:

- The "Constructor, Atom, Translator" (CAT or ⟨C,A⟩,T) notation of Arnold et al. (1986), Arnold & des Tombe (1987), Schmidt (1988a, 1988b), Sharp (1988, 1991)
- The "Eurotra Framework" (EF) notation of Bech & Nygaard (1988), Bech (1991)
- The "Relaxed Compositionality" (RC) notation of Arnold et al. (1988), Arnold & Sadler (1990) [5]

These three differ in status. Firstly, there are three distinct versions of the CAT notation:

- Arnold et al. (1986) and Arnold & des Tombe (1987) describe a "rule-to-rule" translator notation, whose translator rules say roughly: "if a representation (say a source IS) was built by rule R, attempt to build another (say a target IS) by rule R′".
- Schmidt (1988a, 1988b) describes a "representation-to-rule" version, in which the equivalent is roughly "if a representation is of shape S, attempt to build another by rule R′".
- Sharp (1988) mounts a "representation-to-representation" version: "if a representation is of shape S, attempt to build another of shape S′".

(The gloss in (0.4)(c), in particular, will be explicated further below). (0.4)(a) was never used for serious grammar writing as far as we know - it would presumably have suffered from the modularity problem noted in footnote 2. We will consider only (0.4)(c) below, because it is most directly comparable with (0.4)(b) and (c): this is of little significance, however, given the close similarity of all of (0.4).

[5] A full list would also have to include at least the "Constraint Logic Grammar" notation (Balari et al. (1990)) and the "MiMo2" notation (van Noord et al. (1991), Shieber et al. (1989, 1990)). Eurotra has been nothing if not notationally fertile. However, these are less interesting for our purposes. In the former, very little work has been done on the translator notation (the intention was for this to be a close copy of the EF translator notation); and the latter uses the translator notation of the RC system to express the transfer mapping.

Another important piece of background information is that almost all work in Eurotra, perhaps uniquely in its subfamily (at least until the appearance of Kaplan et al. (1989)), adds to the assumptions of section 1 the following further assumption regarding homogeneity[6]:

(III) The same translator notation as is used for transfer is used in the mappings that form parts of analysis and synthesis

This paper concerns itself with which of the Eurotra translator notations is better for transfer. The method, as said above, will be via examples of complex transfer. This leaves open the question of the justification for assumption (III). The latter ought in principle to be justified on the same sort of basis: find some analysis and synthesis mappings which the translator notation will have to perform, and see what translator notation is required to perform them. Crookston (1988) is a very small attempt at this task. It discusses the shortcomings of the CAT notation (i.e. Schmidt (1988a,b)) in extracting adverbs among the English auxiliaries in a c-structure type representation to make them canonically-ordered modifiers of the verb in very roughly an f-structure type representation.

A more important test case would be idioms. A given occurrence of an idiom will in general be scattered across the syntactic representations of its sentence (Schenk (1986) investigates the restrictions on "how scattered" an idiom may be). Consider (4), with the idiomatic parts in italics:

[6]The exception is the "MiMo2" notation, which uses a version of PATR (Shieber (1986)) to arrive at and generate from interface representations, while using the RC translator notation for transfer.

(4) The management always *lays* its failure straight *at* the workers' *door*

However, an idiom is naturally regarded as a semantic and translational unit of the source language, irrespective of what target language is being aimed at. It therefore can be represented as a single word-type unit in interface representation without "tuning". The translator notation therefore has to map between the scattered parts of a syntactic representation of (4), for instance, that represent *lays*, *at*, and *door*, and a single interface representation unit. Arnold & Sadler (1987) demonstrate that the RC translator notation is up to this task: what is not clear is how good the CAT and EF notations would be at it, or indeed whether something with altogether less power would be adequate.[7] It is in general puzzling that (III) has scarcely been defended in the Eurotra literature, given that it is one of the distinctive features of Eurotra.

A further essential piece of background to any discussion of translator notations is the nature of the representations[8] at each end of the mapping; for transfer, the nature of interface representations. The question of what type of interface representation most facilitates the transfer mapping is of the same order of importance as the question of the best translator notation. It is also on the whole just as underdiscussed, though it does form the main topic of Steiner et al. (1988). In the TAUM systems the interface representations were inspired by the deep structures of Standard Theory Transformational Grammar; in CRITTER, and in the DLT system of Schubert (1987), dependency representations are employed. In LFG-MT (Kudo & Nomura (1986), Kaplan et al. (1989)) transfer is between f-structures. The effects of such differing choices are not readily to be discovered.

Most Eurotra work assumes that interface representations are "deep syntactic dependency" representations. "Dependency" basically means here what it means standardly, that every word-like node is a potential governor of some other nodes and a dependant of another node (for a full explication of this notion see Allegranza et al. (1991)). As will become obvious below, some Eurotra work deviates from the standard in "lowering" the dependency governor.

[7] For example the PATR notation used for analysis and generation by MiMo2. One of the many trade-offs in the complex choice of an MT notation is that between an analysis-generation translator notation and a transfer one. The simpler and more constrained the former is, the more idioms it will be unable to coagulate at IS. The residue of idioms will have to be mapped onto their translation equivalents while still "scattered", simultaneously with the mapping of other transfer complexities, thus placing additional strain on the transfer translator notation. (III) is Eurotra's stab in the dark for the ideal trade-off.

[8] To clarify, the nature of the representations in question are not linguistic representations per se, but are *formalisms*.

A very useful rough guide to deciding what "deep syntax" means in Eurotra is that deep syntactic structures tend to be selectional structures. It is roughly true that governors are units that exercise selectional restrictions, and the other dependency roles are occupied by units that have selectional restrictions exercised on them. Moreover, a position which is subjected to the same selectional restrictions by the same governor is usually given the same dependency role regardless of its surface syntactic behaviour: for example, passive subject and active object have the same role in interface. Idioms are unitary governors in interface representation, which accords with the observation that they exercise selectional restrictions as units. We will later exemplify the disappearance of "strongly bound" prepositions in IS, which accords with their selectional irrelevance (see discussion of (**??**) below). This rough heuristic will be sufficient to establish sample IS representations on which translator notations can be tested. The results of such testing will not be affected when the tests are repeated on representations under a full working interface theory of the deep syntactic type.

A minor complication in a comparison of the Eurotra translator notations is this. The three translator notations referred to here–CAT, EF, RC–are parts of their respective MT notations as wholes. The grammar notation and translator notation are packaged together. Moreover, the CAT and EF grammar notations are normally used to generate "lowered-governor" dependency representations at dependency levels; while the RC grammar notation chooses to explore the possibility of "true" dependency structures where the governor is the mother. Ignoring many linguistic details for the sake of simply illustrating the notation, the interface representation of, say, (4), might be represented in CAT as:

(5)

```
(?,{cat=s}).[ (gov,{cat=v,word=lay-at-door}),
              (arg1,{cat=np}).[ (gov,{cat=n,word=management}) ],
              (arg2,{cat=np}).[ (gov,{cat=n,word=failure}) ],
              (arg3,{cat=np}).[ (gov,{cat=n,word=worker}) ]]
```

in EF as:

(6)

```
{cat=s}[ {role=gov,cat=v,word=lay-at-door},
         {role=arg1,cat=np}[ {role=gov,cat=n,word=management} ],
         {role=arg2,cat=np}[ {role=gov,cat=n,word=failure} ],
         {role=arg3,cat=np}[ {role=gov,cat=n,word=worker} ]]
```

and in RC as:

(7)

```
lay-at-door(cat=v).[ arg1=management(cat=n),
                     arg2=failure(cat=n),
                     arg3=worker(cat=n) ]
```

For convenience we adhere to the notations established in the literature for CAT and EF. The difference between a true dependency representation and a lowered-governor one is of no consequence: any of the three translator notations under discussion could be harnessed to any of the three grammar notations with at most minor adjustments. This is only what would be expected given that true and lowered-governor dependency are as near notational variants as makes no difference.

Another detail which is irrelevant to the question of translator notation (though of importance when dicussing systems as a whole, of course) is reversibility. The RC translator notation is reversible, meaning that rules which can be used to create any representation R2 from another R1 can also be used to create R1 from R2. The CAT and EF notations are not reversible: separate sets of rules are needed for each direction[9]. These properties could be reversed (or abolished) without affecting the question of how good the notations are for complex transfer.

CAT and RC are significant points of comparison for the following reasons. CAT is an obvious, basic tree manipulation formalism. As such it forms a base line to which any linguistic MT notation can be compared. It is hard to conceive of a linguistic MT notation which cannot handle all that CAT[10] can handle, so that it is at the point at which CAT cannot handle them that problems become interesting. Any other tree-based MT formalism which does not have the same coverage as CAT can be rejected. RC is a specific attempt to put CAT right. It attempts to characterise the heart of CAT's problems and solve them radically.

[9]If a statement of equality is reversible, it is because the notation has a declarative semantics. It is this fact that makes possible the more constraint-based approaches to equivalence which account for the relative success of the RC notation. In CAT and EF, although analysis and synthesis grammars may indeed be reversible, the systems as a whole can never be, given that interlevel translators, both monolingual and in transfer, are unidirectional.

[10]The EF notation is carried along with CAT in discussions of this kind, since, as argued below, they have essentially the same effective capability for performing bilingual mappings.

Whilst it is beyond the task of this paper to include full syntax and semantics for each of the Eurotra MT notations presented, given that the internal workings of Eurotra remain something of a mystery to many, it is appropriate to comment briefly on some of the design features of these systems.

0.4.1 The CAT Notation

The CAT system (and by extension, the EF system) consists of two distinct formal sets of rules, *generators* and *translators*. Given that Eurotra takes a stratificational approach to translation, each level of representation is described fully by a grammar and separate feature definition (a set of attribute-value pairs). In CAT, each grammar consists of "...a pair ⟨C,A⟩ where C is a finite set of constructors ('rules' defining the class of complex expressions) and A is a finite set of atoms (basic expressions)" (Arnold et al. 1986). Translators (t-rules) interpret the mapping between adjacent levels of representation in a compositional manner, forming partial target level object desriptions in the process. The grammar at each level then attempts to validate (*consolidate*, to use the Eurotra terminology) these linguistic objects formed by the translators, guaranteeing that only well-formed target structures can be built.

Grammars contain three types of rules:

- structure-building rules
- feature rules
- filter rules

Structure building rules–embodying syntactic structures and lexical items–are of course obligatory, while all the others are optional. Feature rules add new information to the feature bundles, including the possibility of inserting feature bundles as sister nodes in a tree. Filter rules are used to delete unwanted objects using unification in two different ways: for *strict* filter rules, if unification fails, then the object is adjudged to be ill-formed, whereas for *killer* filter rules, if unification succeeds, then the object is adjudged to be ill-formed.

Translation of a source object proceeds by its unifying[11] in a top-down, non-deterministic fashion, with the left hand side (LHS) of a t-rule, be it a *default* or *explicit* rule, i.e. the LHS of the rule *subsumes* the object. Explicit t-rules override default t-rules, but if more than one explicit rule can pply, then different target representations are built. The right hand side (RHS) of the t-rule specifies the target object to be built, again by unification, provided that such an object is licensed by the target grammar. This manipulation of the source object is performed compositionally, given that only those rules whose LHS unifies with (matches, in EF) the appropriate part of the object contribute to its target description. This process is under the control of the user to some extent given that there is, in practice, no limit to the complexity of either side of t-rules, i.e. it may specify a tree to an arbitrary level of detail.

In sum, the CAT system can be categorised as "...in effect a context free grammar notation augmented by a simple feature theory based on unification, and (via the t-rules) the capacity for certain transformations" (Arnold et al, 1986).

0.4.2 The EF translator notation

The only significant difference between EF and CAT as regards translator notations is the notion of "descriptor", also known as "weak" or "unconsolidated" representation (Bech & Nygaard, 1988). Nevertheless, this feature barely affects transfer, for reasons we will examine directly.

In particular, (43) holds in EF just as much as in CAT, and consequently the same problems with combinations of certain hard cases can be observed, along with the same tendency for translator grammars to approximate to sentence dictionaries. The particular class of tame constructions for CAT alluded to in the preceding section (and discussed in depth below) would be tame for EF in exactly the same way, and the wild remainder live on with undiminished ferocity.

[11] This shows a difference between CAT and EF. In the latter, matching rather than unification is used to prevent the potential introduction of new features into the object which could cause it to become illegal according to the source grammar. This is because "although the insertion of nodes in EF can be performed exactly as in CAT2, i.e. in t-rules, EF also allows for node insertion within generators." (Sharp, 1991)

A "descriptor" is a representation that can be processed by the target grammar. In the CAT notation as discussed above, the RHS of a translator rule results in a tree that must be checked by the target grammar, that is, roughly, accepted or rejected according to whether there are rules that license it. A descriptor is completely re-parsed by the target grammar in a process known as "consolidation", resulting in new leaves, mid-tree nodes and entire subtrees being added to the descriptor to form the final "consolidated" representation. A vastly simplified example would be the insertion of an English VP node in synthesis, via the following rule:

(8)
```
S:{cat=s}[ V:{cat=v},
           SUBJ:{cat=np},
           OBJ:{cat=np} ]
=>
S < SUBJ, V, OBJ >
```

where the angled brackets signify the dominance relation in a descriptorVP nodes occurring in surface structure are deleted at higher levels, given the desideratum of flattened, dependency structures at IS, where the V is promoted to the role of governor of the sentence. In generation, the converse is required, as in (9).. The descriptor on the RHS of this, assuming the presence of typical rules for English surface S's and VP's, will be consolidated into:

(9)
```
{cat=s}[ {cat=np},
         {cat=vp}[ {cat=v},
                   {cat=np} ]]
```

However, this power barely affects transfer. How can we be so sure of that? Simply, because a monolingual target IS grammar cannot process representations in any way which will solve bilingual translation problems.

Take *savoir* for example. Avoiding (for now) the wild complexity of inserting a *how* in the object clause in transfer, roughly the following descriptor would be produced for consolidation by the English IS:

(10)
```
{cat=s} < {role=gov,word=know},
          {role=arg1},
          {role=arg2,cat=s} < {role=gov},
                              {role=arg2} >>
```

To turn this into the correct translation of *savoir* plus infinitive, rules amounting to the following need to be present in the English IS grammar:

(11) Insert an AdvP headed by *how* in an infinitive clause that is the arg2 of *know*.

Such embellishments are in fact within the power of EF consolidation. But they cannot be performed on the output of transfer, because what is done to the output of transfer has to be monolingually motivated. Rules adding up to (11) are not rules of English. It involves a false claim that (for instance) *The director knows not to do that* is ungrammatical, and another that *The director knows how not to do that* is its nearest grammatical paraphrase. *The director knows not to do that* might well be required as the translation of some other expression from some language, and (11) would destroy it. (11) cannot be added to an IS grammar of English without gross tuning and accompanying loss of maintainability and inspectability.

It is significant in this respect that Bech & Nygaard limit their claims to a monolingual example (in fact a synthesis example). When the target grammar is in the same language as the source, the claims of the EF translator notation and of the distinctive device that is consolidation deserve close scrutiny. Consider a close parallel of (11):

(12) Shoehorn a PP headed by *on* around the NP after *rely*.

This is monolingually valid for English surface constituent structure, and if written into the appropriate grammar simplifies the translation mapping of synthesis. But transfer is a different problem.

0.4.3 The Relaxed Compositionality Notation

As will be seen, the RC notation adopts quite a different approach to translation than either the CAT or EF notations.

Translation is achieved by the successive application of a set of t-rules to a source object. The LHS of each t-rule removes some part(s) from the object and leaves the rest of the object behind, to act as input to the next appropriate t-rule, which will extract some part(s), reducing the source object even more. The RHS, conversely, builds a target object "brick-by-brick", by combining the translations of these pieces (or by directly inserting material); each t-rule adds more information to the target object until it is complete.

Those parts which have been extracted by the LHS of the t-rules are either discarded (indicated by "!") or linked to a place in the target object by means of indices (e.g. "a!"). The latter are themselves considered new objects by the translators and are treated in exactly the same fashion as outlined above. The corresponding indices on the RHS of the t-rule indicate where the output of these subsequent t-rules is to be positioned.

In sum, then, translation proceeds by a step-by-step reduction of a source object until it has been dismantled, mirrored by a step-by-step construction of the target object until complete. Importantly, the number of steps involved is variable and there is no requirement that all daughters of any one node must be dealt with simultaneously by one t-rule. Consequently, in contrast to the CAT and EF notation, the grammar writer using the RC notation is not limited by the "local tree" restriction as stated in (43) above.

The essence of the RC notation is that the way in which the source object is decomposed into parts is under the control of the user and need not necessarily reflect the way in which the object was built nor must the construction of target object by the RHS of the t-rules necessarily reflect the way the target grammar would have built it.

Examples of such parts might be:

a) A node and all its substructure plus a feature on another node
b) Several features scattered over several nodes
c) A node and part of its substructure

and so on.

All that is required is that, once complete, the target objects are legal in the sense that they *could* have been built by the monolingual grammars. It is in this sense that the compositionality of the translation process is "relaxed". Another way of looking at this is that the "whole-local-tree" restriction which causes so many problems in CAT is removed in RC. The way the source object is decomposed is tightly constrained in CAT by this restriction so that decomposition closely reflects monolingual construction[12].

[12]In fact, the only practical restriction on t-rules is that the t-rule must actively affect the object in some way: either by division of the input into a minimum of two bound parts or by extracting and discarding a minimum of one feature or one piece of structure. This is to prevent the system looping. Once an object is subjected to a given t-rule, the remainder of that object is then offered to the set of t-rules, including this t-rule, again. In this way, if the object remains unchanged by any one t-rule, the t-rule will continue to match the object indefinitely, and the system will loop.

It is appropriate here to note an observation from Arnold & Sadler (1989, p.19):

> "Notice that relaxed compositionality does not result in anything like 'transformational' power ... In addition, the output of a t-rule t1 which relates L1 to L2 is always an object in the target language L2, and so cannot be input to another t-rule from the same translator (t-rules from one translator can collaborate in decomposing an object, but cannot operate "in sequence" with one taking the output of the other). Thus, it is not possible to use t-rules to perform familiar transformations involving unbounded movements, or successive cyclical raising."

The requirement expressed earlier, that of rules which deal only with the exceptional part of a translation, leaving the more "normal" parts to be dealt with by general t-rules, is, therefore, achievable in the RC notation. However, as will be seen, problems do arise which limit the potential of the RC notation in this area.

0.5 A Typology of Translation Cases

0.5.1 Introduction

We shall see that all transfer cases divide into two classes for CAT. Firstly, there are the "tame" cases, whose rules abstain from encroaching on eachother, i.e. individual rules whose interaction with other rules leads to the desired outcome. These translational problems can be solved by writing one rule per problem case. Secondly, there are the "wild" cases, which encroach on eachother's rules and on those of the "tame" cases, i.e. despite being correctly formulated rules, they nevertheless fail to interact correctly with certain other rules. This encroachment engenders special rules for each combination of wild-wild and tame-wild.

A little more precisely, there are a large number of possible divisions into tame and wild, which we shall discuss at length below. It should suffice to note here that whatever division we choose to make proves no problem for the RC notation, but does not help CAT to solve certain combinations of hard translational problems in a compositional fashion, reducing the machine close to the level of a sentence dictionary.

We will show that one instance of two rules failing to interact correctly is in the translation of:

(13) On sait le faire $\Longrightarrow$ How to do it is known.

where the translation necessitates passivisation (translating *on* as *one* in such examples leads to unnatural English sentences), as well as the insertion of *how* after the verb in English. Ignoring the details of such rules for the time being, we can summarize here by saying that it is possible to cope with both complex transfer phenomena in a modular fashion, as desired, by writing two rules which perform the translations:

(14) a. FR: *on* + transitive verb $\Longrightarrow$ EN: passive

b. FR: *savoir* $\Longrightarrow$ EN: *know* + *how*

but when they occur in the same sentence, as in (37), the individual rules impinge on one another, thereby necessitating a new rule to be written for the combination of complex transfer cases.

In such circumstances, one could insist that the *on* case had to be regarded as tame, and so a division could be set up that way. The consequence would be that every complex transfer case involving a transitive verb (such as the *savoir* case) would have to be regarded as wild, because their rules encroach on the *on* rule. If the *savoir* case were regarded as tame, then the cases exemplified in (15) and (17) below, and very many more cases, would become wild. However, there are many other complex transfer rules such as (23), which would be tame, since the *savoir* rule would never encroach on a rule for *lors*.

Perhaps one of the natural views is to regard as tame those cases which need to refer to the lexical unit of the governor of any local tree that they describe. The definition of the tame class seems then to be:

(IV)

> The lexical unit of a governor is "implicated" in a rule if that governor is given by an identifier which is interpreted as the translation of a mentioned lexical unit.
>
> A translator rule is "criterially tame" if the lexical unit of every governor given in the rule either
> (i) is mentioned; or
> (ii) is implicated.
>
> A translator rule is "tame" if either
> (i) it is criterially tame; or
> (ii) it fails to encroach on any criterially tame rule.

(IV)(b) includes (31), since there every governor's lexical unit is explicitly mentioned; it also includes simple transfer rules as tame.

Nevertheless, as stated previously, whatever class one ends up as regarding as wild is unworkable in the CAT notation, because these are the cases where the translator grammar will approximate to a sentence dictionary. One can write a translator grammar for any class of tame cases, and the wild remainder will then be in effect untranslatable.

For the transfer cases we have been able to examine, (IV) appears to apply without producing contradictions to define a subclass which is in fact one of the possible tame classes for CAT. Why is it the case that referring to specific governors all the way down should guarantee a rule's tameness? Wildness occurs whenever two rules put distinct constraints on the same node. Given the procedural interpretation of the CAT notation, the "all-the-way-down" restriction *guarantees* that all information regarding translation flows from governor to dependant. Thus a fortiori nodes do not get conflicting constraints from (say) mother and sister. Nevertheless, the bilingual data and our assumptions concerning it require that information *does* flow back: the point is to ensure that when it does, it is not inconsistent with the information at the higher node. To reiterate, it is the local tree restriction that gives rise to this inconsistency[13].

On such a view, so far as they can be identified by looking at individual words of French,[14] the wild cases are not at all common, a fact which tends to justify regarding this as a natural view. In fact we know of four wild cases in French-English transfer:

(15) On le fait $\Longrightarrow$ It is done

(16) Le gouvernement sait le faire $\Longrightarrow$ The government knows how to do it

(17) Le gouvernement permet que les étudiants le fassent $\Longrightarrow$ The government permits the students to do it

(18) Il vient de le faire $\Longrightarrow$ He has just done it

The specifics of these translation problems are discussed in depth below. However, with this latter example, for instance, it should suffice to state at this juncture that a new rule would have to be written for the combination:

(19) On vient de le faire $\Longrightarrow$ It has just been done

as well as for any combinations of any such rules.

[13]It may be helpful to the reader to think of the local tree restriction as not permitting rules to specify any context, whether this be higher, lower, or width-wise in a tree; all nodes on the LHS of a rule are active–they will all be rewritten.

[14]An important point to stress, of course, is that a phenomenon cannot be wild in one language alone, but only in translation to another named language. For example, the German word *Rückstand* (= *backlog*) appears innocuous in isolation, but the sentence fragment *einen Rückstand haben* has to be translated as *lag* to cover all contexts.

The tame cases defined by (IV) fall into a sort of hierarchy. At the bottom there is simple transfer, where only the lexical units differ. Next up are those where only feature annotations differ, as in:

(20)
moindre $\Longrightarrow$ slightest
```
(gov,{word=moindre}) =>
(gov,{word=slight,degree=superlative})
```
veiller à $\Longrightarrow$ look after
```
(gov,{word=veiller,pformofarg2=a}) =>
     (gov,{word=look,pformofarg2=after})
```
sujet de $\Longrightarrow$ grounds for
```
(gov,{word=sujet,pformofarg1=de}) =>
     (gov,{word=ground,number=plural,pformofarg1=for})
```

Most of these involve the pformofargN attributes, the mortal remains of "strongly-bound" prepositions (see discussion of (**??**) above).

The next most complex cases which are tame on the view proposed here are the "whole-tree" cases where one whole subtree is replaced by another. For example:

(21)
jusqu'ici $\Longrightarrow$ until now
le cas échéant $\Longrightarrow$ if the situation arises

which require the following transfer rules:

(22)

```
(mod,{cat=advp}).[ (gov,{word=jusqu'ici}) ]
=>
(mod,{cat=pp}).[ (gov,{word=until}),
                 (arg1,{cat=pp}).[ (gov,{word=now}) ]

(mod,{advp}).[ (gov,{word=le-cas-echeant}) ]
=>
(mod,{pp}).[ (gov,{word=if}),
             (arg1,{s}).[ (gov,{word=arise}),
                          (arg1,{np}).[ (gov,{word=situation}) ]]]
```

A characteristic of these rules[15] is that they feature no identifiers. That is, there is no unit such that the translation of that unit has to be found and positioned on the RHS.

The next most complex are those which necessarily involve the whole local tree, but do require identifiers. *Lors de* is an example of this:

[15] *Le cas échéant* is a completely fixed locution of French, hence one word in IS.

(23) lors de l'élection $\Longrightarrow$ at the time of the election

Another example is[16] Suppose also, that voice annotations are, in unproblematic cases, dealt with by the following RC rule, *t-voice*, which effectively strips the voice annotation from the source object making it unavailable for any subsequent t-rule operating on the remainder of the source object:

(55)
```
rest! ?(cat=v, a!voice) <=> rest! ?(cat=v, a!voice)
```

As stated earlier, RC t-rules can collaborate in the progressive decomposition of an object and the way in which this is achieved is by the successive application of a set of rules. Importantly, since a t-rule is applicable if the LHS of the rule strictly matches the source object, at any stage in the process more than one t-rule may be applicable for any given object. Thus, *t-on-rc*, *t-gen* and *t-voice* all match and could apply to (51). It can be seen straightaway from these three rules alone that the way in which the t-rules collaborate can directly affect the outcome of the translation process. If, for example, *t-voice* applies to a given

[16] A lexical transfer rule such as:

(24) t_104 = word=trop $\Longrightarrow$ word=too.

will ensure the correct translation of *trop* here. There is consequently no need to state explicitly :

(25) (beaucoup) trop $\Longrightarrow$ (a lot) too much

```
(?).[ $TROP:(gov,{word=trop}),
      $MOD:^mod ]
=>
(?,{cat=ap}).[ (gov,{word=much}),
               mod.[ $TROP,
                     $MOD ]]
```

These might be called "whole-tree shoehorn cases" if a label were desired: *much* and its mother are shoehorned around the whole LHS tree.

Then there are cases where a governor is translated as a governor plus a complete dependant. These have an established label in the literature, the "Schimmel" cases, from the famous example where the German noun *Schimmel* translates as the English governor *horse* plus a complete AP modifier containing *white*.[17] For example:

(26) revoir $\Longrightarrow$ see again
inscrire $\Longrightarrow$ write down
renverser $\Longrightarrow$ knock over

```
(?,{cat=s}).[ $REVOIR:(gov,{word=revoir}),
              $REST:*(?) ]
=>
(?,{cat=s}).[ $REVOIR,
              $REST,
              (mod,{cat=advp}).[ (gov,{word=again}) ]]
```

It is at this point on the hierarchy of tame cases that the CAT notation loses its naturalness. These cases still appear to be tame by our definition, but they specify the translation of nodes which are not intuitively involved in the complex case. In the rule given above, these are all the arguments and modifiers of *revoir* identified by `$REST` it is unintuitive to mention them. To put it another way, were it not for restriction (43), these constituents would not be mentioned in the rule. This characteristic of involving innocent bystanders is to be observed at the two remaining points on the hierarchy.

Next most complex are pure relation-changing cases, of which *plaire-like* is the standard example. These seem to be vanishingly rare, at least in French-to-English transfer. *Plaire-like* itself of course is naturally bypassed by translating *plaire* as *appeal-to*. The present study of 2000 French words identified the following solitary case:

(27) Cette proposition rappelle la dernière au comité $\Longrightarrow$ This proposal reminds the committee of the last one

```
(?,{cat=s}).[ $RAPPELER:(gov,{word=rappeler}),
              $1:arg1,
              $2:arg2,
              $3:arg3,
              $MODS:*mod ]
=>
(?,{cat=s}).[ $RAPPELER,
              $1,
              $3,
              $2,
              $MODS ]
```

Finally, there are the "shoehorn" cases to which (23), the *lors* case, as well as (16), the *savoir* case, belong. (16) has been sufficiently discussed to date for their notational characteristics to be familiar, although we go into considerably more depth in the following sections. The innocent bystanders in (16) are the subject and modifiers of *savoir*. Since they have many interesting translational aspects, all those encountered in the present study are listed below[18]:

(28) assister à quelquechose $\Longrightarrow$ be present at something
s'averer une bêtise $\Longrightarrow$ turn out to be a mistake
pouvoir faire quelquechose $\Longrightarrow$ be able to do something
valoir beaucoup $\Longrightarrow$ be worth a lot
risquer faire quelquechose $\Longrightarrow$ be likely to do something
John dispose d'un ordinateur $\Longrightarrow$ a computer is available to John
Ce tâche incombe à John $\Longrightarrow$ John is responsible for this task
rapprocher quelquechose de quelquechose $\Longrightarrow$ bring something closer to something
opposer quelqu'un à quelqu'un $\Longrightarrow$ bring someone into opposition with someone

0.5.2 The degree of success of the CAT translator notation

Having adopted such a typology, we will now show in considerable detail how CAT and RC attempt to cope with some of these cases. Initial investigations will centre on four hard cases of transfer, namely (15), (16), (17), and (23).

In CAT, the representations involved in (23) are[19]:

(30)

```
(?,{pp}).[ (gov,{word=lors}),
           (arg1,{np}).[ (gov,{word=election}) ]]

(?,{pp}).[ (gov,{word=at}),
           (arg1,{np}).[ (gov,{word=time}),
           (mod,{pp}).[ (gov,{word=of}),
                        (arg1,{np}),[ (gov,{word=election}) ]]]]
```

Put generally, the problem is to add in an extra NP *time*, as well as to "shoehorn in" a level of PP structure headed by *of*, conditioned by the source word being *lors*. CAT (version (0.4)(c)) does this with this rule *t-lors*:

(31)

object first, *t-on-rc* will be unable to apply because *t-voice* removes part of the object necessary to *t-on-rc*, the voice annotation, and vice versa.

```
(?,{cat=pp}).[ (gov,{word=lors}),
               $1:arg1 ]
=>
(?,{cat=pp}).[ (gov,{word=at}),
               (arg1,{cat=np}).[ (gov,{word=time}),
                                 (mod,{cat=pp}).[ (gov,{word=of}),
                                                  $1 ]]]
```

where in such a rule, the part preceding the arrow is known as the left-hand side or LHS and that following the arrow as the RHS[20].

We will examine the interpretation of these rules by investigating what happens when two complexities occur in the same source sentence. In the first case transfer is successful, and in the second not so. The successful case is:

(32) On le fait lors de l'élection $\Longrightarrow$ It is done at the time of the election

Let us look first at the translation of the simpler sentence (15), where the two representations involved are in essentials:

(33)

```
(?,{s,voice=active}).[ (gov,{word=faire}),
                       (arg1,{np}).[ (gov,{word=on}) ],
                       (arg2,{np}).[ (gov,{word=le}) ]]

(?,{s,voice=passive}).[ (gov,{word=do}),
                        (arg2,{np}).[ (gov,{word=it}) ]]
```

Here is the rule for (15), *t-on*:

(34)

```
(?,{voice=active}).[ $G:gov,
                     (arg1,{cat=np}).[ (gov,{word=on}) ],
                     $2:(arg2,{cat=np}),
                     $MODS:*(?) ]
=>
(?,{voice=passive}).[ $G,
                      $2,
                      $MODS ]
```

That is, an active transitive sentence with *on* as subject is almost always best translated as an agentless passive. This is true, because it is difficult to provide natural-sounding equivalents for *on* in English[21], and the agentless passive makes it possible to avoid the question of equivalents. No claim is made here about the question of how a machine should translate *on* in other circumstances.

Note that the active transitive sentence with *on* as subject is in no way a natural class of French: sentences with *on* as subject are one class, active transitives are another. Given this, one would be justified in including a feature in French IS objects to represent each of these linguistic generalisations, but not the class "active transitive sentence with *on* as subject", any more than "verbs starting with a /b/", or something. Such sentences cannot, therefore, be specially coded in the IS of French (which would undoubtedly simplify their translation) without coming to be at the top of the slippery slope that is tuning.

Returning to our example (32), we assume a translator grammar containing *t-lors*, *t-on*, some simple rule for translating NP's (*t-np*, say), and rules to transfer leaves as necessary. We will ignore as irrelevant for present purposes the translation of determiners, tenses, and other details. The source structure is in essentials:

(35)

```
(?,{s,voice=active}).
          [ (gov,{word=faire}),
            (arg1,{np}).[ (gov,{word=on}) ],
            (arg2,{np}).[ (gov,{word=le}) ],
            (mod,{pp}).[ (gov,{word=lors}),
                         (arg1,{np}).[ (gov,{word=election}) ]]]
```

and the target:

(36)

```
(?,{s,voice=passive}).
   [(gov,{word=do}),
    (arg2,{np}).[(gov,{word=it})],
    (mod,{pp}).[(gov,{word=at}),
                (arg1,{np}).
                     [(gov,{word=time}),
                      (mod,{pp}).[(gov,{word=of}),
                                  (arg1,{np}).
[(gov,{word=election})]]]]]]
```

The first thing that happens in transfer is that the LHS of (34) unifies with (35). As a result, the identifier `$G` is associated with the *faire* node, and so on for the other nodes. `$MODS` is associated with the PP headed by *lors*. The rule is read as an instruction: to find the translation of anything that unifies with this LHS, first find the translations of the things identified as `$G` etc. A leaf rule gives the translation of `$G` as `(gov,{word=do})`, and *t-np* gives appropriate simple translations of the NP's. Then the translator grammar tries to find a translation of the *lors* PP identified by `$MODS`.

The LHS of (31) unifies with this PP. The complement NP is identified as (a different) `$1`, and *t-np* yields a translation of it. New material is added over this NP as specified in the RHS of (31). As each level of hierarchy is built, there is a check that the target grammar permits such a structure. (The translator cannot build anything that is not English, where what is English is independently defined by the English IS grammar.) We now have a translation of (35)'s `$MODS`.

The RHS of (34) can now come into play. We now have translations of `$G`, `$2`, and `$MODS`. We see that the voice changes to passive, with the source ARG1 deleted in translation to produce an agentless passive in English. This is checked against the target grammar. Then finally the three daughters are arranged in order as the daughters of a new node `(?,{cat=s})`, and the result of that is checked. If all the grammar-writers concerned have done their job properly, translation is successful.

This is an informal outline of the interpretation of these two rules. The important point about this interpretation is this: at the fringe[22] of the part of (35) matched by the LHS of (34), other rules are looked for. The node of (35) identified by `$MODS` in (34), the *lors* PP, is part of this fringe, and for that node rule (31) is found. For other parts of the same fringe there is a leaf rule and *t-np*. At the fringe of the part of (35) matched by (31), there is one node identified `$1`, and rule *t-np* is invoked again for that node. In general, a search for more rules is started at fringe nodes.

So much for our successful example. Now let us look at the translation of the following sentence:

(37) On sait le faire $\Longrightarrow$ How to do it is known[23]

The rule *t-savoir* for the simpler sentence (16) is:

(38)

```
(?,{s}).[ $SAVOIR:(gov,{word=savoir}),
          $1:arg1,
          (arg2,{s,verbform=infin}).[ $2:*(?) ],
          $MODS:*mod ]
=>
(?,{s}).[ $SAVOIR,
          $1,
          (arg2,{s,verbform=infin}).[ $2,
                                      (mod,{advp}).[
                                           (gov,{word=how}) ]]
          $MODS]
```

This rule disregards the arg2 constituent containing *le faire* as a unit. Instead, it translates first *faire* and then *le* as separate nodes, both identified by `$2`, and then adds *how* as a sister to the translations of these nodes.

Thus (38) is different from (23). In the latter, a governor plus one of its arguments translates as a governor with some structure shoehorned beside and over the translation of that argument. Here a governor (*savoir*) translates as a governor (*know*) plus a modifier of its argument: more of a gluing on than a shoehorning in.[24]

Before we proceed further, one amendment needs to be made to rule *t-on*. As noted earlier, as it stands currently this rule tells us how to translate *on* only in those cases where the ARG2 is an NP. Given that the ARG2 in (37) is a sentential object, the rule needs to be generalised as follows, *t-on-general*:

(39)

```
(?,{voice=active}).[ $G:gov,
                     (arg1,{cat=np}).[ (gov,{word=on}) ],
                     $2:arg2,
                     $MODS:*(?) ]
=>
(?,{voice=passive}).[ $G,
                      $2,
                      $MODS ]
```

The source IS for *On sait le faire* is:

(40)

```
(?,{s}).[ (gov,{word=savoir}),
          (arg1,{np}).[(gov,{word=on})],
          (arg2,{s,verbform=infin}).[(gov,{word=faire}),
                                     (arg2,{np}).[(gov,{word=le})]]]
```

Let us assume first of all a translator grammar containing leaf rules, *t-on-general*, and *t-savoir*. The LHS of *t-savoir* will match (40), assigning `$SAVOIR` to the verb governor and `$1` to the *on* NP. Here, at the fringe of the part of *t-savoir* matched by (40), as expounded above in connection with example (32), it begins the search for further rules. It finds a leaf rule to translate `$SAVOIR`, with *le faire* translated as outlined previously, with *how* added as a sister to the translations of these nodes. But what about `$1`, the *on* NP?

It certainly will not find the desired equivalent. The desired equivalent is as given by (39), a voice switch with non-translation of the arg1, and the LHS of (39) does not match the substructure now identified as `$1`. That substructure is `{cat=np}` with a noun governor, and the LHS of (39) matches things that are `{cat=s}` with a verb governor.

The LHS of (39) will also match (40). `$G` will be assigned to the *savoir* leaf, and `$2` to the infinitival clause. After the application of leaf rules and *t-np*, the RHS of (39) will try to assemble the following structure:

(41)

```
(?,{voice=passive}).[(gov,{word=know}),
                     (arg2,{s}).[(gov,{word=do}),
                                 (arg2,{np}).[(gov,{word=it})]]]
```

where the synthesis grammar will produce (after subsequent application of other rules at other levels) *It is known to do it*, which is not a translation of the source sentence.

Either way, translation fails. The reason is that *t-savoir* and *t-on-general* needlessly encroach on each other's territory. *t-savoir* tells the machine how to translate the arg1 of *savoir*, by assigning it the identifier `$1` which says "look for a rule that translates this". This is so even though there is no need for it: the arg1 of *savoir* is not part of the transfer problem surrounding *savoir* and should be left transparent to all other rules. Similarly, *t-on-general* tells the machine how to translate the governor and arg2 of an *on* sentence, even though they are not part of the complexity involved there and should be accessible to a rule like *t-savoir*.

A similar example is (18), where the rule might be:

(42)

```
(?,{s}).[ (gov,{word=venir}),
          (arg1,{s}).[ $2:*(?) ]]
=>
(?,{s,aspect=perfective}).[ $2,
                            (mod,{cat=advp}).[ (gov,{word=just}) ]]
```

(We assume that the surface subject of *venir-de* becomes a dependant in the lower clause in French IS, because *venir-de* does not selectionally restrict its subject while the lower verb generally will.) This mentions the lower governor, as one of the sequence of nodes identified by `$2`, purely as an innocent bystander, and this too encroaches on tame rules and other wild rules.

The obvious reaction, then, is to try to draft rules of this type in such a way that they only specify the parts of the translation that they need to, perhaps like this:

(38)′

```
$SAVOIR:(gov,{lu=savoir}),
$2:arg2
=>
$SAVOIR,
(arg2,{s}).[ $2,
             (mod,{advp}).[(gov,{word=how})]]
```

(39)′

```
(?,{voice=active}).[ (arg1,{cat=np}).[ (gov,{word=on}) ]]
=>
(?,{voice=passive})
```

These mention exactly the parts involved: the governor and arg2 of *savoir* and the voice feature and arg1 of an *on* sentence (mentioning the arg1 only to throw it away by its non-appearance on the RHS). However, this takes us well beyond the realm of the CAT translator notation. (38)′ is not even within the notation proper, since it features two trees (albeit of depth 0) on the LHS rather than one. (39)′ does not have the interpretation desired, since the LHS will only match those (non-existent) representation parts which have a `voice=active` node with exactly one daughter, an arg1 of category NP headed by *on*.

In other words, the CAT translator notation makes a kind of "local tree" restriction. Suppose we define a local tree as a node with all of its daughters, thus restricting the notion to local trees of depth 1. Then the CAT restriction is:

(43) If a rule mentions any two nodes in a local tree, it must mention the whole local tree.

This means that if a rule mentions two nodes in a local tree (i.e. a root and a daughter), the object subtree must have a root and a daughter matching that rule, and the subtree must have no other daughters. In other words, either side of a t-rule describes width-wise, but not depth-wise, a complete subtree. This combines with the already mentioned restriction that further rules are only called at the fringe of the given rule to produce the problem under discussion of "encroachment". Under (43), *t-savoir*, for instance, is forced to mention the arg1 of *savoir*, and a further rule can only be called at that arg1 node. So when the arg1 is *on*, *t-on-general* cannot be called, because it does not "start" at the fringe of *t-savoir*[25].

The only option open to a CAT grammar writer is to add to the translator grammar a specific rule for *on sait le faire*, thus:

(44)
```
(?,{s,voice=active}).[ $SAVOIR:(gov,{word=savoir}),
                       (arg1,{np}).[ (gov,{word=on}) ],
                       (arg2,{s,voice=infin}).[$2:*(?)],
                       $MODS:*mod]
=>
(?,{s,voice=passive}).
      [ $SAVOIR,
        (arg2,{s,voice=infin}).
           [$2,
            (mod,{advp}).[(gov,{word=how})]],
        $MODS]
```

(44) combines the operations of *t-savoir* and *t-on-general* in a single rule (but of course only applies where the two cases are combined).

Similar problems will occur with most combinations of the constructions exemplified at the beginning of this section in (15), (16), and (17)[26]. The rule for (17) is:

(45)
```
(?,{s}).[ $PERMETTRE:(gov,{word=permettre}),
          $1:arg1,
          (arg2,{s,verbform=finite}).[ $2-G:gov,
                                       $2-1:arg1,
                                       $2:*(?) ],
          $MODS:*(?) ]
=>
(?,{s}).[ $PERMETTRE,
          $1,
          $2-1,
          (arg2,{s,verbform=finite}).[ $2-G:gov,
                                       $2:*(?) ],
          $MODS ]
```

The main assumption here is that *permit* in English is a three-place "object-control" (i.e. Equi) verb. We take it that this assumption is licensed by data of the following kind[27]:

(46) I permitted John to be examined by the doctor $\neq$ I permitted the doctor to examine John

Thus in the English side of (17), *the students* is an argument of *permit* not of *do*. This is reflected in the LHS of the rule where the arg1 of the lower clause, `$2-1`, is lifted one level up the hierarchy.[28]

Now consider a combination of (15) and (17), for example:

(47)

On permet que les étudiants le fassent $\Longrightarrow$ The students are permitted to do it

where it should not be beyond the reader to see that the rules *t-on-general* and *t-permettre* encroach on eachother. Again, a special rule for the combinations will have to be written.

So, the interpretation of the notation has to include an element of ordering[31].

Similarly, the *savoir* rule encroaches on the *permettre* rule in:

(48) Le gouvernement sait permettre que les étudiants le fassent ⟹ The government knows how to permit the students to do it

and vice versa in:

(49) Le gouvernement permet que les étudiants sachent le faire ⟹ The government permits the students to know how to do it

Readers are invited to determine for themselves the "encroachments" that would occur in:

(50) On sait permettre que les étudiants le fassent ⟹ How to permit the students to do it is known

We will provide this clue, that the net result is that a specific rule will have to be written for the combination of constructions here exemplified.

This section has attempted to exemplify in detail the claim made briefly in Arnold et al. (1988, section 2) that "strict compositionality" in translator notations, the type of restriction that we have worded as (43), the adoption of which guarantees termination, leads to a new rule being added for each combination of problem cases. The CAT translator notation in some cases reduces the machine close to the level of a sentence dictionary. And this has to be the main verdict on this notation—it simply fails to deal with the hardest cases. Considered as a theory of transfer, it is falsified by a small amount of the data.

0.5.3 The degree of success of the RC translator notation

If we first approach the problem posed in (15) (slightly modified here), i.e. *On le fait maintenant* → *It is done now*, the (simplified) source and target objects in RC would be (51) and (52).[29]

(51)
```
faire(voice=active).[arg1=on,
                     arg2=le,
                     mod=maintenant]
```

(52)
```
do(voice=passive).[arg1='EMPTY',
                   arg2=it,
                   mod=now]
```

Since the RC notation allows a t-rule to operate on an object without affecting all the daughters of the governing verb, let us assume the grammar contains the following rule *t-on-rc*:

(53)
```
rest! ?(cat=v,!voice=active).[!arg1=on,
                               arg2]
<=>
rest! ?(cat=v,!voice=passive).[!arg1='EMPTY',
                               arg2]
```

The interpretation of (53) as a piece of RC notation is as follows. `!arg1=on` means, quite simply, "extract and discard the whole of the arg1 of a verb when it is headed by *on*" (Note that in this instance the *only* structure in the arg1 is the lexical unit *on*. However, in principle, extraction involves an entire branch unless some part of it is "saved" by being separately bound). The branch headed by *on* ceases to exist on the application of (53). Similarly, `!voice=active` means "extract and discard the feature annotation voice=active". The remainder of the object, that which is left after these two parts have been extracted is bound to the variable `rest` (which includes the arg2 node which is mentioned here purely for context).

Given that it is true that, for the vast majority of verbs, translation is an unproblematic task of linking the source verb and frame with the target verb and frame, let us assume, for the moment, a "general" t-rule for unproblematic cases in which the whole local tree is specified. The interpretation of *t-gen* is virtually identical to that of a similar rule in CAT[30]

[31] Critics of the RC approach might be alarmed at the 'burden' of rule-ordering being put

This is not necessary in CAT and EF, for in these formalisms rules are restricted to dealing with a whole local tree, so cooperation of rules is not an issue, given that only one rule applies anyway. We will concentrate for the time being on the interpretation which involves a non-deterministic application of t-rules, that is:

(V) For a given object, all possible combinations of applicable rules will be tried.

We can now return to rules *t-on-rc* and *t-gen* applying to (51). Suppose *t-gen* applies first. The arg1, arg2 and mod are extracted, and called "a", "b" and "m". T-rules operating on these parts provide a translation for "b", *it*, for "m", *now* and for "a" also. "a", *on* will be translated as *one* for the sake of examples like:

(56) On mange bien en France $\Longrightarrow$ One eats well in France

where the passive translation is not applicable. After *t-gen* has applied, the mother in (51) is the only part of the original object to be bound to `rest` and will eventually be found to be *do*. *t-on-rc* cannot now apply because the arg1 and arg2 nodes have been extracted. On the RHS, the mother *do*, the arg1 *on*, arg2 *it* and mod *now* will be assembled, resulting in the unwanted translation *One does it now*.

Now, following (V), another derivational path is followed, in which *t-on-rc* applies first. The daughter headed by *on* and the annotation `voice=active` are extracted. The mother and all remaining daughters are bound to `rest`. The `rest` of *t-on-rc* would be:

(57)
```
faire(cat=v).[arg2=le,
              mod=maintenant]
```

(57) would now be matched by *t-gen* which will apply as previously described

back on the grammar writer, which in a practical enterprise such as MT may be significant. The ELU system of Estival et al. (1990) allows a certain amount of rule-ordering in transfer, but this is compiled into the system (using their notion of 'rule specificity', based on subsumption), and which cannot be altered by the grammar writer. The main reason why this is not permitted is because it would entail having to "check at compile time for coherence and computability, and this checking would be computationally very expensive, indeed in some cases intractable" (op cit, p.139). Nevertheless, even in CAT or EF - or even in any NLP application written in a (so-called) 'declarative' language (as is the case with ELU, for instance) - one orders rules to a certain extent, so as to ensure the order of application of grammar rules is as optimal as possible, allowing for 'normal' input. Again, it should suffice to quote from Arnold & Sadler (1989, pp. 24-25), where they state their motivation for requiring explicit rule-ordering as threefold:

> "First, experience suggests that users often have some clear ideas about there being an optimal order in which rules should be applied ... and it seems unreasonable to prevent the linguist from stating this.
> Second, the order of application of mutually compatible rules sometimes has linguistic significance in terms of scope ... and in order to ensure that only the grammatical result is produced ... (certain) rules must apply before (others).
> Finally, of course, applying rules non-deterministically in all possible orders is not computationally very efficient."

with the exception of the arg1 slot, which is no longer present. The mother, found to be *do*, the arg2, *it* and the mod, *now* will be assembled on the RHS and and empty arg1 slot and the annotation `voice=passive` will be added. The result is the desired object (52), from which the correct translation will be synthesised.

Let us now see what happens with (16), repeated here:

(16) Le gouvernement sait le faire $\Longrightarrow$ The government knows how to do it

Here, the source object would be (58), the target object (59).

(58)
```
savoir.[arg1=government,
        arg2=faire(finite=no).[arg1='EMPTY',
                               arg2=le ]]
```

(59)
```
know.[arg1=government,
      arg2=do(finite=no).[arg1='EMPTY',
                          arg2=it,
                          mod=how]]
```

Recall that the problematic part of the object is the governing verb and its arg2. If we deal *only* with the problematic part of the translation, then we would need a t-rule something like (60), *t-savoir-rc*, again, exactly what is advocated in discussion of (43) above.

(60)
```
!savoir.[a!arg2= ?(cat=v,finite=no)]
<=>
!know.[a!arg2= ?(cat=v,finite=no).[!mod=how]]
```

When going from French to English, this rule binds the arg2 of *savoir* to "a" when it is sentential and non-finite and extracts and discards *savoir* itself. On the RHS, the governor *know* is inserted, the translation of "a" is positioned in the arg2 slot and *how* is inserted as a modifier in the lower clause. From English to French, the reverse is true.

However, *t-savoir-rc* deals *only* with the problematic part of the object. The arg1 and any potential main verb modifiers are still untreated. The only other rule currently available to deal with these 'normal' parts is *t-gen* above. Yet, if we look at the effects of interaction between these two rules, we see that they simply cannot cooperate in the translation of the object.

If *t-savoir-rc* operates on the object (58) first, then not only is the governor, *savoir* extracted and discarded, but, vitally, so is the arg1 and any modifiers. Recall that "!" indicates extraction not only of a node, but also all substructure not separately bound to indices. In *t-savoir-rc* only the arg2 node is bound and thereby *saved*. Once *t-savoir-rc* has applied, no structure is left to act as input to *t-gen*. Clearly, in order to achieve translation of this object, *t-savoir-rc* must apply *after* the arg1 and any modifiers have been dealt with.

However, if *t-gen* were to operate on the object first, the effect would be to remove *all* daughter nodes from the structure, leaving only the governing verb attached to `rest`. Included in the daughter nodes extracted would be the arg2

node, which is essential input to *t-savoir-rc*. Thus, once *t-gen* has operated, the input for *t-savoir-rc* has been removed, rendering *t-savoir-rc* inapplicable.

Given that a representative grammar will also contain lexical rules for *savoir*, *know* and *how*, to account for data such as:

(61) a. John doesn't know how he did it $\Longrightarrow$ John ne sait pas comment il l'a fait

b. He knows Mary $\Longrightarrow$ Il connait Mary

the application of *t-gen* first will result in an incorrect, compositional translation of the object.

Two options are open to the grammar writer: to formulate *t-gen* in such a way that the unproblematic parts of an object are treated differently, or to include the unproblematic parts in *t-savoir-rc*. If the latter course were to be adopted, the rule dealing with *savoir*$\Longleftrightarrow$*know how* would be formulated thus:

(62)
```
!savoir.[a!arg1,
         b!arg2(finite=no),
         m!*mod]
<=>
!know.[a!arg1,
       b!arg2(finite=no).[!mod=how],
       m!mod]
```

However, formulating the problem case in this manner is rather self-defeating. In the first place, such a solution clearly fails to separate the problematic parts from the unproblematic parts. Furthermore, it is immediately obvious that a rule such as (62) would not combine successfully with the rule dealing with agentless passives above, *t-on-rc*, and a further rule would be necessary to deal with the combination of hard cases that occurs in examples such as *On sait le faire*.

As mentioned in Arnold et al (1988), as an alternative to the "frame-to-frame" approach adopted in *t-gen*, the "linking rule" strategy can be adopted.

This involves using separate t-rules to deal with the governor and each of the slots for the dependants. Since *t-on-rc* and *t-savoir-rc* are already themselves examples of this strategy, it seems perfectly reasonable to suppose that other dependents of the governor would be treated in a similar fashion. Thus, instead of regular translation of unproblematic arguments/modifiers being executed by one rule such as *t-gen*, several complementary t-rules would co-operate to achieve the same effect:

(63) `rest! ?(cat=v).[a!arg1] <=> rest! ?(cat=v).[a!arg1]`

(64) `rest! ?(cat=v).[b!arg2] <=> rest! ?(cat=v).[b!arg2]`

(65) `rest! ?(cat=v).[c!arg3] <=> rest! ?(cat=v).[c!arg3]`

(66) `rest! ?(cat=v).[c!arg4] <=> rest! ?(cat=v).[c!arg4]`

(67) `rest! ?(cat=v).[m!mod] <=> rest! ?(cat=v).[m!mod]`

(63) states that the translation of the arg1 of a given verb will become the arg1 of the translation of that verb; (64) states that the arg2 becomes the arg2 and

so on. The rules would operate on an object consecutively and, effectively, such an approach constitutes a reiterative decomposition of the source object.

Applying this strategy to the problem in hand, that of how to deal with the arg1 and any modifiers of savoir, these would be treated by (63) and (67) respectively, with the original rule, *t-savoir-rc* treating the head and the arg2.

This solution may seem to be ideal. The problematic part of the translation has been dealt with in a separate rule which specifies only the problematic parts. The rest of the translation is dealt with in a "natural" way by the linking rules. The same principle should apply with any piece of complex transfer.

Returning to the discussion of the problematic verbs, with one exception, interaction between them does not present any problems for the RC notation.

The appropriate t-rules for (23), (16) and (17) are :

(68) (23) lors de l'élection $\Longrightarrow$ at the time of the election

```
!lors.[a!arg1(pform=de)]
<=>
!at.[!time.[!mod=of.[a!arg1]]]
```

(69) (16) Le gouvernement sait le faire $\Longrightarrow$ The government knows how to do it

```
!savoir.[a!arg2]
<=>
!know.[!mod=how.[a!arg2]]
```

(70) (17) Le gouvernement permet que les étudiants le fassent $\Longrightarrow$ The government permits the students to do it

```
rest!permettre.[b!arg2= ?(cat=v).[a!arg1= ?(cat=n)]]
<=>
rest!permit.[a!arg2,
             b!arg3= ?(cat=v).[!arg1='EMPTY']]
```

In (71) three hard cases are combined:

(71) On sait permettre que les étudiants le fassent

The object would be:

(72)
```
savoir(voice=active).
   [arg1=on,
    arg2=permettre.[arg1='EMPTY',
                    arg2=faire.[arg1=etudiants,
                                arg2=le]]]
```

If we assume an optimal ordering of the relevant t-rules, *t-on-rc* would operate first, stripping out the feature `voice=active` on the mother node and the arg1 of *savoir*. The rest of the object would be:

(73) `savoir.[arg2=permettre.[arg1='EMPTY',`
`arg2=faire.[arg1=etudiants,`
`arg2=le]]]`

t-savoir-rc (=(60)) would now apply and would remove the governer *savoir*, binding the arg2 to "a", leaving:

(74) `[arg2=permettre.[arg1='EMPTY',`
`arg2=faire.[arg1=etudiants,`
`arg2=le]]]`

t-perm, (70) could now apply, binding *faire* to "b" and its arg1, *étudiants* to "a" ("raising" it). The unproblematic arg1 of (74) would be extracted and bound by (63), with the arg2 of (74) treated similarly and bound by (64). The translations of these parts will be assembled on the RHS, resulting in successful transfer.

With one exception, the combination of the other complex rules yields equally successful results. The reason for this is that only the lexical units specific to each problematic case are mentioned in each rule. No call for translation of any other parts not relevant to the problematic case is made, leaving these free and intact to act as input to subsequent t-rules. Thus, the reason that these rules collaborate so successfully is that none of them encroach on another's territory. That is, they all deal with separate parts of the object and do not directly affect the parts needed by subsequent rules.

However, as has been mentioned, there is one exception to this success story. Interaction between *t-on-rc* and *t-perm* is impossible when *on* is the arg1 of the lower clause. The reason for this is that both rules specify and extract the same piece of structure. An example sentence might be *Le gouvernement permet qu'on le fasse*, which needs to be translated as *The government permits it to be done*[32]. Given this, the source object would be:

(76) `permettre.[arg1=gouvernement,`
`arg2=faire(voice=active).[arg1=on,`
`arg2=le]]`

t-on-rc obviously cannot apply until all structure above the governing verb of the lower clause *faire* has been stripped from the object, since the LHS of *t-on-rc*

[32] As mentioned in Section 2.1, our working assumption is that *permit* is an Equi verb. Given this, the active to passive conversion of its complement is unjustified, and it is appropriate that it fails. Nevertheless, it is difficult to see how the two-place predicate *permettre* could legitimately translate as the three-place predicate *permit* under any circumstances. It would appear, however, that the translation required above can only be licensed if *permit* is a *raising* verb. Notwithstanding sentences like (46), there are some cases such as:

(75) The law permits cyclists to ride on the pavement = The law permits pavements to be ridden on by cyclists

where there does not seem to be a significant difference in meaning. It would appear, therefore, that this verb can be ambiguous between equi and raising readings. As we illustrate here, RC cannot handle this rule interaction when *permit* has its raising reading; if it could, our argument that RC is an adequately expressive notation would be strengthened considerably. However, as just argued, the reason why RC fails on this example is not just a problem with its notation, but more fundamentally a genuine conflict of semantic information.

starts at that lower governor. However, the rule which will do this, *t-perm*, will *also* extract the arg1 from the lower clause, and link it to the arg2 position in the higher clause. Since *t-perm* effectively removes the *on* node from the object the context needed for *t-on-rc* to apply ceases to exist, rendering it inapplicable.

Unless one is prepared to rewrite *t-perm* in such a way that the arg1 is not affected, a solution which would inevitably involve more problems, the only solution is to write a special t-rule which combines *t-on-rc* and *t-perm*. This is unfortunately the only recourse currently available whenever two t-rules affect the same piece of structure.

0.5.4 Rule Interaction in RC

Having shown that, in theory at least, it is possible within the RC notation to separate the problematic part of a translation from the unproblematic parts, let us return to the question of interaction between "special" rules such as *t-savoir-rc* and the other "general" rules proposed, namely, (63) – (67).

Before we investigate this question, let us first examine the different relations that can exist between pairs of t-rules (cf Arnold & Sadler (1990)).

(77) a. the LHS of two t-rules are disjoint if they do not both match any one object (e.g. *t-savoir-rc* and *t-lors-rc* are disjoint).

b. two t-rules are in competition with each other if they could both apply to an object (e.g. *t-on-rc* and *t-gen* are in competition with each other).

c. (i) a t-rule A is incompatible with, or "bleeds" B if A is in competition with B and the application of A extracts part of an object necessary for B to match the successor object, thus preventing the subsequent application of B to the reduced object. For instance, *t-gen* is incompatible with *t-on-rc* and *t-savoir-rc* and *t-arg2* are mutually incompatible.
(ii) a t-rule A is compatible with B if A is in competition with B and the application of A does not prevent the subsequent application of B to the reduced object. For instance, *t-on-rc* is compatible with *t-gen*[33]. *t-arg1* and *t-arg2* are mutually compatible.

We assumed in (V) a non-deterministic application of t-rules. That is (in the strict sense of the word), that all possible derivational paths should be followed. A closer examination of this reveals that may be an unsatisfactory arrangement for several reasons. In the first place if we look at the five mutually compatible t-rules relating to the unproblematic transfer of arguments and modifiers, (63), (64), (65), (66) and (67), we see that if (V) is adhered to, then translation of a four-argument verb with modifiers can be achieved via 120 (i.e. 5 factorial) derivational paths, resulting in 120 identical target objects. If the translational requirement is to have a target object for every possible translation of the source object, multiple target objects are only valid if they reflect ambiguity in the

[33]Note that the process need not be transitive, given that applying these rules the other way round causes them to be incompatible.

source object. So, identical objects must be rejected at some stage. If this stage is during parsing[34], then such rules must somehow be seen by the parser as belonging to a batch or set of rules and a restriction built in ensuring that any member of such a set fires only once during a particular stage of the translation, thus preventing backtracking which tries all possible permutations.

As stated previously, since in CAT and EF rules are restricted to dealing with a whole local tree, more than one rule never cooperates on a given local tree. The RC notation is forced, therefore, to address problems which cannot arise in the simpler notations.

Secondly, there is the problem which occurs when two t-rules are mutually incompatible but only one derivational path yields a correct object. If non-deterministic rule application is adhered to, both a correct *and* an incorrect object will be produced. An example of such a problem is the interaction between *t-on-rc* and *t-arg1*: if there exists in the grammar a lexical rule translating *on* into *one* then two objects will be produced, one desired agentless passive English object (via *t-on-rc*) and one undesired active English object with 'one' in the arg1 slot (via *t-arg1*).

In an attempt to avoid these problems, the RC notation makes use of an elementary rule "inhibition" mechanism. This allows the grammar writer to make arbitrary statements about which rule(s) in a set of competitive rules must be tried before other rules in the same set. This mechanism effectively prevents the parser from following certain derivational paths.

(78) Rule 1 $\prec$ Rule 2

In (78) "$\prec$" indicates that Rule 2 can only apply if:

(79) a. Rule 1 does not apply

b. If Rule 1 applies, its application does not bleed Rule 2

The effect of such a mechanism is two-fold. Some rules (typically the "default" rules) can be "inhibited" by others (typically the "special" rules) in that they can only apply iff other rules do not, thus avoiding the cases where translation via the default rules leads to incorrect objects.

As an example, the specific rule *t-on-rc* could be made to "inhibit" the general rule *t-arg1* thus:

(80) *t-on-rc* $\prec$ *t-arg1*

What this means is that *t-arg1* can only ever apply to an object iff *t-on-rc* cannot, which is exactly what is wanted if a compositional translation is not required.

When a compositional translation *is* possible, the relevant t-rules would simply be marked as having no precedence relation (indicated by separating rules with a comma) with respect to each other resulting in as many derivational paths as the object is ambiguous. In such cases, it may be deemed useful to

[34]It should be clear by now that grammars at each level "parse" incoming objects to check for acceptability. Given this, the reader should not confuse this usage of the term "parsing" with its normal usage, i.e. of analysing source input strings.

incorporate some kind of preference mechanism in the operation which will put the derived objects into a prescribed order.

Similarly, where two (or more) rules are in competition with each other and are mutually compatible (eg *t-arg1*, *t-arg2* etc), only one combination of the rules is permissible. Identical objects created because more than one derivational path can be followed are thus eliminated. For example:

(81) *t-arg1* $\prec$ *t-arg2* $\prec$ *t-arg3* $\prec$ *t-arg4* $\prec$ *t-mod*

This simply means that given any relevant object, *t-arg1* will fire first, then the remainder of the object will be offered to *t-arg2*, *t-arg2* will fire if applicable, and so on. The precise precedence among this set of rules is arbitrary at this stage. They are all mutually compatible, and, if one of the rules isn't applicable (say *t-arg4* with a three-place predicate, or if there are no modifiers present in the object) the parser simply bypasses that rule and moves to the next one in the group. The important thing is that no other combination is tried.

Returning to the problematic case involving the rule *t-savoir-rc*, recall that the compositional translation of (58) is wrong. Since *t-savoir-rc* and *t-arg2* are mutually incompatible, inhibiting *t-arg2* by *t-savoir-rc* would mean that *t-arg2* would only operate iff *t-savoir-rc* could not.

Note, however, that if *t-savoir-rc* is incompatible with *t-arg2*, it is *also* incompatible with the two other rules needed to treat the object, namely *t-arg1* and *t-mod*, because all three rules affect the governing verb. Therefore, to achieve correct translation a statement of the following type would be needed:

(82) *t-arg1* $\prec$ *t-mod* $\prec$ *t-savoir-rc* $\prec$ *t-arg2* [=(64)] $\prec$ *t-arg3* [=(65)] $\prec$ *t-arg4* [=(66)]

This would produce a correct translation of the object because the arg1 and any modifiers would be handled before *t-savoir-rc* fired. Also, non-problematic cases would still be treated in the way described above because *t-savoir-rc* would not match the object and would be bypassed, leaving *t-arg2* free to fire. The placing of *t-arg3* and *t-arg4* within the new statement is arbitrary since they are not applicable to the *savoir* special case anyway.

The problematic daughter dealt with in *t-savoir-rc* is the arg2. However, other problematic cases involve different daughters and it is here that difficulties with the rule inhibition mechanism arise. The following is a problematic case where a verb plus modifer translates into verb:

(83) Jean est entré dans la pièce en courant $\Longrightarrow$ Jean ran into the room.

where the relevant objects are:

(84)
```
entrer.[arg1=jean,
        arg2=dans.[arg1=piece],
        mod=en.[arg1=courant]]

run.[arg1=jean,
     arg2=into.[arg1=room]]
```

and the special t-rule would look like this:

(85)
```
!entrer.[!mod=en.[!courant],
        m!*mod]
<=>
!run.[m!*mod]
```

For such a rule to collaborate successfully with the general rules, it would clearly need to inhibit *t-mod*, but not *t-arg1* and *t-arg2*, leading to the statement:

(86) *t-arg1* $\prec$ *t-arg2* $\prec$ *t-arg3* $\prec$ *t-arg4* $\prec$ *t-entrer* [=(85)] $\prec$ *t-mod*

and there's the rub. According to (86), to accommodate *t-savoir-rc* the relation between *t-mod* and *t-arg2* must be:

(87) *t-mod* $\prec$ *t-arg2*

whereas to accommodate (85) the reverse is true, i.e

(88) *t-arg2* $\prec$ *t-mod*

a fact which simply cannot be expressed in one statement.

The only solution available in the current notation is for any special rule to contain within it additional rules to handle those parts of the translation which would normally be handled by rules which the special rule inhibits. Thus, assuming a statement of the following kind:

(89) *t-arg1* $\prec$ *t-savoir-rc* $\prec$ *t-arg2* $\prec$ *t-arg3* $\prec$ *t-arg4* $\prec$ *t-entrer* $\prec$ *t-mod*

the special rule, *t-savoir-rc* will have to be revised to include any modifiers which would normally have been dealt with by *t-mod* but which no longer can be, since *t-savoir-rc* will inhibit *t-mod* within this statement, thus:

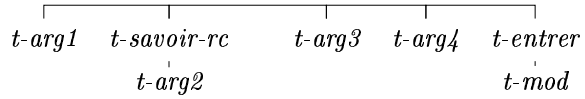
t-savoir-rc'

```
!savoir.[a!arg2= ?(cat=v,finite=no),
         m!*mods]
<=>
!know.[a!arg2= ?(cat=v,finite=no).[!mod=how],
       m!*mods]
```

Thus, the system itself can cause the linguistically unmotivated inclusion of innocent bystanders in t-rules in certain cases.

Ideally, what is required is for all the general "default" rules to apply non-deterministically, but for the "specific" rules to continue to inhibit their corresponding "default" rules as described above. Rather than the rule writer being forced to come up with seemingly arbitrary statements such as (81) or (82), we instead advocate the refinement of such information so that less arbitrary information is added until a contradiction is reached. The following diagram shows the situation regarding choice of rule application for the phenomena discussed above, namely that whilst args one, two and three can always be treated by the default rules, what we would like is for *t-savoir-rc* to affect appropriate objects before the default rule *t-arg2*, which in so doing will prevent this latter from firing, and similarly, the optimum treatment for modifiers is where rule *t-entrer* acts on appropriate objects before *t-mod*:

(90)



which might be expressed, were it possible to do so, in the current notation thus:

(91) *t-arg1*, [*t-savoir-rc* $\prec$ *t-arg2*], *t-arg3*, *t-arg4*, [*t-entrer* $\prec$ *t-mod*]

According to this statement, special rules like *t-savoir-rc* and *t-entrer* will inhibit their corresponding general t-rule, thus preventing compositional translation as necessary, but also that each subgroup of rules having an internal precedence would have no precedence relation with any other subgroup.

Unfortunately, this brings us back to the original problem with regard to the general t-rules since they are, once again, not inhibited by each other, leaving the problem of overgeneration of identical objects unsolved. Still more unfortunately, it would no longer be feasible to suggest some sort of mechanism to ensure that rules within a specified batch only fired once, since this is only applicable when the end result of multiple derivational paths is multiple identical objects. The approach described above would certainly produce the 120 identical objects previously mentioned. It would also produce more than one solution when a problem case is encountered.

Let us suppose that the object in question is (58), the *savoir* case. According to the inhibition statement, the relevant rules could apply as follows:

(92) a. *t-arg1 t-mod t-savoir-rc*

b. *t-mod t-arg1 t-savoir-rc*

c. *t-savoir-rc*

d. *t-mod t-savoir-rc*

e. *t-arg1 t-savoir-rc*

(92)(a) and (92)(b) would both produce correct, albeit identical objects. (92)(c) and (92)(d) would cause the translation to fail because in both cases, *t-savoir-rc* fires before *t-arg1* has the opportunity to do so, and the translators would attempt to build a target object without an arg1. Recall that a target object must be legal according to the target grammars and the lack of an obligatory argument in the target object would render the object invalid and hence rejected.

Order (92)(e), however, presents a further difficulty, not easily overcome. The two obligatory arguments in the target object will be translated by *t-arg1* and *t-savoir-rc*. However, since *t-savoir-rc* is incompatible with *t-mod*, once *t-savoir-rc* has fired, *t-mod* cannot and any modifiers in the source object will not be translated. Importantly, the incomplete translation will be a valid target object according to the target grammars, since modifiers are necessarily optional, and the object will survive.

One might suggest that the solution to this difficulty is to introduce some sort of mechanism for checking that all parts of the source object have been dealt with. The problem is that because extraction on the LHS of a t-rule automatically extracts all substructure of a marked node unless part of it is "saved" by being bound, *whether or not that substructure is mentioned in the t-rule*, the modifiers have indeed been "dealt with" in that they have been extracted from the object by *t-savoir-rc* at the same time as the governing verb.

Any solution which imposes restrictions on the effects of the extract/discard function which cannot also be imposed on the extract/bind function is untenable. So, to suggest, as one might want to, that extraction and throwing away of lexical items can only take place if they are all explicitly mentioned whilst maintaining that a whole branch can be bound just by binding the top node, is not possible.

0.6 Concluding Remarks

We have classified here a number of cases of dificult translation which any reputable MT system would have to attempt to cope with. We have shown that the CAT and EF translator notations encounter certain problems, centrally problems created by the local-tree restriction in their semantics.

It is hard to avoid the conclusion that MT must move on from local-tree systems for bilingual mappings. The central question for translator notations, it seems to us, is the question of the facts: what pieces of a source representation are allowed to correspond to what pieces of a target representation in occuring

translation equivalents? The local tree answer is that source local trees correspond to target local trees, and the data examined here shows that this is not so, if it shows nothing else. Considered as a theory of translation data, the local-tree approach is constrained and clear but too weak.

The RC notation is an attempt to put right the failings of CAT. CAT does not work because of the local tree restriction, which however, ensures computability. Observers may worry about the computational cost of introducing the notion of Relaxed Compositionality - is it fair, for instance, to call it an unconstrained tree-to-tree transducer, with the associated comment that it is no surprise that it allows one to express almost anything? This is a little in the eye of the beholder: it is easily demonstrated that it is not "unconstrained". There are no "path" variables allowed, so one cannot say, for instance, "take something from somewhere arbitrarily deep in the tree and move it to the front". As pointed out already, however, the main thing that distinguishes RC from having true "transformational" power is that the latter relies on one rule feeding another (i.e. with the complete tree) and then another and another, for as many rules as one likes. The interesting constraint here is the "one shot" idea, and the restriction to a fixed number of levels. Beyond these two ideas, however, of course it is unconstrained. To put it more neutrally, it is very powerful, but as we have shown, sadly not powerful enough.

Our intuition is that an answer to the central question just posed is currently only available in terms of specific examples, hence our approach here. The *savoir* type of example shows that a governor plus a dependant of a dependant of that governor can be implicated in a bilingual mapping. [CHANGE: The *wollen* type shows that a governor plus one aspect (more concretely, one feature) of one of its dependants can be so implicated]. And so on. What the limits are on this variation is not fully clear, but those are the limits which translator notations must attempt to span.

In the rest of the literature, this aspect of rule co-operation is usually not expounded about a notation. It is thus usually impossible to judge whether a notation has solved these problems or not. However, it can at least be said of most other notations that no claim has been advanced that they can handle the kind of translation examples that RC can handle.

If we make the reasonably safe assumption that the perfect MT notation has yet to be invented, then we feel that when it comes to coverage at any rate, any other system will need to ensure that it can treat *at least* the combination of exceptional translation cases that we have shown RC is capable of handling. This applies to all systems, whether they have their roots in semantic-based translation, or KBMT, or shake-and-bake, or whatever, but in particular to those formalisms less constrained than those of Eurotra, for we have shown what range of phenomena can be handled by a relatively weak formalism, using a low-level linguistic representation and a one-shot transfer strategy, for a large number of languages. Given this, if the choice of a more powerful system is to be justified, then we feel that it should achieve comparable coverage to that

presented here, and more besides.

As mentioned at the outset of this paper, it is a genuine option to hope that the questions posed here will go away. But until they do, it is an equally genuine option to attempt to solve them. We hope that the detailed translation analyses offered here will provide one of the starting-points for future attempted answers.

References

Allegranza, V., P. Bennett, J. Durand, F. Van Eynde, L. Humphreys, P. Schmidt, E. Steiner (1991) "Linguistics for Machine Translation: The Eurotra Linguistic Specifications", in Copeland et al. (1991a), pp. 15–123.

Alshawi, H. (1996) "Head Automata and Bilingual Tiling: Translation with Minimal Representations", in *34th Conference of the Association for Computational Linguistics*, Santa Cruz, California.

Appelo, L. (1993) *Categorial Divergences in a Compositional Translation System*, PhD thesis, University of Utrecht.

Appelo, L., C. Fellinger, & J. Landsbergen (1987) "Subgrammars, Rule Classes and Control in the Rosetta Translation System", in *Third Conference of the European Chapter of the Association for Computational Linguistics*, Bonn, pp. 118–133.

Arnold, D., S. Krauwer, M. Rosner, L. des Tombe & G. B. Varile (1986) "The ⟨C,A⟩,T Framework in EUROTRA: A Theoretically Committed Notation for MT", in COLING: *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, pp. 297–303.

Arnold, D., S. Krauwer, L. des Tombe & L. Sadler (1988), "'Relaxed' Compositionality in Machine Translation", in *Second nternational Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon University, Pittsburgh, pp. 61–81.

Arnold, D., & L. Sadler (1987), "(Non)–Compositionality and Translation", in J. Peckham (ed.) (1987), pp. 23–55.

Arnold, D., & L. Sadler (1989) "Mimo: Theoretical Aspects of the System, Working Papers in Language Processing 6, Department of Language and Linguistics, University of Essex, Colchester.

Arnold, D., & L. Sadler (1990) "The Theoretical Basis of MiMo", *Machine Translation*, **5**:195–222.

Arnold, D. & L. des Tombe (1987) "Basic Theory and Methodology in EUROTRA", in S Nirenburg, ed., *Machine Translation: Theoretical and Methodological Issues*, Cambridge University Press, Cambridge, pp. 114–135.

Balari, S., L. Damas, N. Moreira, & G. B. Varile (1990) "CLG(n): Constraint Logic Grammars", in COLING: *13th International Conference on Computational Linguistics, Helsinki, vol. 3, pp. 7–12.*

Beaven, J. (1992) "Shake–and–Bake Machine Translation", in COLING: Proceedings of the 15th International Conference on Computational Linguistics, *Nantes, France, pp. 603–609.*

Bech, A., & A. Nygaard (1988) "The E–Framework: A Formalism for Natural Language Processing", in COLING: Proceedings of the 12th International Conference on Computational Linguistics, *Budapest, pp. 36–39.*
Bech, A. (1991) "Description of the E–framework", in Copeland et al. (1991a), pp. 7–40.
Bresnan, J. (1982, ed.) The Mental Representation of Grammatical Relations, *MIT Press, Cambridge, Mass.*
Copeland, C., J. Durand, S. Krauwer, B. Maegaard (eds) (1991a) The Eurotra Linguistic Specifications, *CEC, Luxembourg.*
Crookston, I. (1988) "Linguistic Unmotivation in EUROTRA", EUROTRA-Essex Internal Memorandum 11, Department of Language & Linguistics, University of Essex, Colchester.
Crookston, I. (1990) "The E-Framework: Emerging Problems", in COLING: 13th International Conference on Computational Linguistics, Helsinki, vol. 2, pp. 66–71.
Estival, D., A. Ballim, G. Russell, S. Warwick (1990) "A Syntax and Semantics for Feature–Structure Transfer" in *Third Conference on Theoretical and Methodological Issues in MT*, University of Texas, Austin, pp. 131–143.
Guilbaud, J. P. (1987) "Principles and Results of a German to French MT System at Grenoble University (GETA)", in King (1987), pp. 278–318.
Haddock, N. J., E.Klein & G. Morrill (eds) (1987) *Working Papers in Cognitive Science, Volume 1: Categorial Grammar, Unification Grammar and Parsing*, University of Edinburgh.
Hutchins, W. J. (1986) *Machine Translation, Past, Present and Future*, Ellis Horwood, Chichester.
Isabelle, P. (1987) "Machine Translation at the TAUM Group", in King (1987), pp. 247–277.
Isabelle, P., M. Dymetman, & E. Macklovitch (1988) "CRITTER: A Translation System for Agricultural Market Reports", in COLING: *12th International Conference on Computational Linguistics, pp. 261–266.*
Johnson, R. (1987) "Translation", in Whitelock et al. (1987), pp. 257–285.
Kaplan, R. M. & J. Bresnan (1982) "Lexical-Functional Grammar: A Formal System for Grammatical Representation", in Bresnan (1982), pp. 173–281
Kaplan, R. M., K. Netter, J. Wedekind & A. Zaenen (1989) "Translation by Structural Correspondences", in Fourth Conference of the European Chapter of the Association for Computational Linguistics, *Manchester, pp. 272–281.*
Kaplan, R. M., & J. Wedekind (1993) "Restriction and Correspondence-based Translation", in Sixth Conference of the European Chapter of the Association for Computational Linguistics, *Utrecht, pp. 193–202.*
King, M. (1987 ed.) Machine Translation Today, *Edinburgh University Press, Edinburgh.*
Krauwer, S., & L. des Tombe (1984) "Transfer in a Multilingual MT System", in COLING: 10th International Conference on Computational Linguistics, Stanford, pp. 464–467.
Kudo, I., & H. Nomura (1986) "Lexical–Functional Transfer: A Transfer Framework in a Machine Translation System Based on LFG", in COLING: *11th International Conference on Computational Linguistics, Bonn, pp. 112–114.*

Landsbergen, J. (1987a) "Montague Grammar and Machine Translation", in Whitelock et al. (1987), pp. 113–148.
Landsbergen, J. (1987b) "Isomorphic Grammars in the ROSETTA Translation System", in King (1987), pp. 351–372.
Leermakers, R., & J. Rous (1986) "The Translation Method of ROSETTA", in Computers and Translation *1, pp. 169–183.*
Maas, H–D. (1987) "The MT System SUSY" in King (1987), pp. 209–246.
Nagao, M. & J. Tsujii (1986) "The Transfer Phase of the Mu Machine Translation System", in COLING: 11th International Conference on Computational Linguistics, *pp. 97–103.*
Odijk, J. (1989) "The Organisation of the ROSETTA Grammars", in Fourth Conference of the European Chapter of the Association for Computational Linguistics, *Manchester, pp. 80–86.*
Peckham, J. (1987, ed.) Recent Developments and Applications of Natural Language Understanding, *Kogan Page, London.*
Radford, A. (1988) Transformational Syntax: A First Course, *Cambridge University Press, Cambridge.*
Rosetta, M. T. (1994) Compositional Translation, *Kluwer, Dordrecht.*
Rupp, C. J., M. A. Rosner & R. L. Johnson (1994, eds.), Constraints, Language and Computation, *Academic Press, London.*
Sadler, L., I. Crookston & A. Way (1989) "Co-description, projection, and 'difficult' translation", Working Papers in Language Processing 8, Department of Language and Linguistics, University of Essex, Colchester.
Sadler, L., I. Crookston, D. Arnold & A. Way (1990) "LFG and Translation", in Third Conference on Theoretical and Methodological Issues in MT, *University of Texas, Austin, pp. 121–130.*
Sanfilippo, A., T. Briscoe & A. Copestake, (1992) "Translation Equivalence and Lexicalization in the ACQUILEX LKB", in Fourth Conference on Theoretical and Methodological Issues in MT, *Montreal, Canada, pp. 1–11.*
Schenk, A. (1986) "Idioms in the ROSETTA Machine Translation System", in COLING: 11th International Conference on Computational Linguistics, Bonn, pp. 319–324.
Schmidt, P. (1988a), "A Syntactic Description of a Fragment of German in the EUROTRA Framework", in Steiner et al. (1988), pp. 11–39.
Schmidt, P. (1988b), "Transfer Strategies in EUROTRA–D", in Steiner et al (1988), pp. 161–179.
Schubert, K. (1987) *Metataxis: Contrastive Dependency Syntax for Machine Translation*, Foris, Dordrecht.
Sharp, R. (1988), "CAT-2–Implementing a Formalism for Multi-Lingual MT", in *Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon University, Pittsburgh, pp. 76–87.
Sharp, R. (1991), "CAT-2–An Experimental Eurotra Alternative", in *Machine Translation*, **6**:3, pp. 215–228.
Shieber, S. M. (1986) *An Introduction to Unification-Based Approaches to Grammar*, CSLI, Stanford.
Shieber, S. M., G. van Noord, R. C. Moore & F. C. N. Pereira (1989) "A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms", in *27th Conference of the Association for Computational Linguistics*, Vancouver, pp. 7–17.

Shieber, S. M., G. van Noord, R. C. Moore & F. C. N. Pereira (1990) "Semantic-Head-Driven Generation", in *Computational Linguistics*, **16**:1, pp. 30–42.
Slocum, J. (1985) "A Survey of Machine Translation: Its History, Current Status and Future Prospects" in *Computational Linguistics* **11**:1–17. Also in idem (ed.) (1988) *Machine Translation Systems*, Cambridge University Press, Cambridge, pp. 3–47.
Steiner, E., P. Schmidt, & C. Zelinsky–Wibbelt (1988 eds) *From Syntax to Semantics: Insights from Machine Translation*, Pinter, London.
Van Eynde, F. (1993) "Machine Translation and Linguistic Motivation", in F. Van Eynde (1993, ed.) *Linguistic Issues in Machine Translation*, Pinter, London.
van Noord, G., J. Dorrepaal, P. van der Eijk, M. Florenza, H. Ruessink, & L. des Tombe (1991) "An Overview of MiMo2", in *Machine Translation*, **6**:210–214.
Whitelock, P. (1992) "Shake-and-Bake Translation", in COLING: *15th International Conference on Computational Linguistics*, Nantes, France, pp. 784–791.
Whitelock, P. (1994) "Shake-and-Bake Translation", in Rupp et al. (1994), pp. 339–359.
Whitelock, P., M. M. Wood, H. L. Somers, R. Johnson, & P. Bennett (1987 eds) *Linguistic Theory and Computer Applications* Academic Press, London.
Zajac, R. (1990), "A Relational Approach to Translation", in *Third Conference on Theoretical and Methodological Issues in MT*, University of Texas, Austin, pp. 235–254.
Zeevat, H., E. Klein & J. Calder (1987) "Unificational Categorial Grammar", in Haddock et al. (eds) (1987), pp. 195–222.