Natural Language Engineering **3** (1): 15–55. Printed in the United Kingdom

© 1997 Cambridge University Press

Enabling technology for multilingual natural language generation: the KPML *development environment*

JOHN A. BATEMAN

Language and Communication Research Group,
Department of English Studies, University of Stirling,
Stirling FK9 4LA, Scotland, UK

(*Received 20 December 1996; revised 12 May 1997*)

Abstract

Natural language generation is now moving away from research prototypes into more practical applications. Generation functionality is also being asked to play a more significant role in established applications such as machine translation. In both cases, multilingual generation techniques have much to offer. However, the take-up of multilingual generation is being restricted by a critical lack both of large-scale linguistic resources suited to the generation task and of appropriate development environments. This paper describes KPML, a multilingual development environment that offers one possible solution to these problems. KPML aims to provide generation projects with standardized, broad-coverage, reusable resources and a basic engine for using such resources for generation. A variety of focused debugging aids ensure efficient maintenance, while supporting multilingual work such as contrastive language development and automatic merging of independently developed resources. KPML is based on a new, generic approach to multilinguality in resource description that extends significantly beyond previous approaches. The system has already been used in a number of large generation projects and is freely available to the generation community.

1 Introduction

It is increasingly regarded as a requirement for Natural Language Processing (NLP) systems that they support applications involving more than one natural language. Multilingual solutions are desired for increasingly multicultural social and economic worlds. Among these applications, the need for *information presentation* in diverse languages is rapidly multiplying, supported by international legislation (such as that of the European Union) and multi-national companies with broad international customer bases. Whereas translation has been one traditional means of producing information in diverse target languages, recent changes in work practices have started focusing on parallel multilingual document production. Instead of writing a source text and employing translators or translation systems to produce multilingual variants, more satisfactory results can often be obtained by technical writers who

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

are competent both in the domain area of a document and in the required target languages (cf. Hartley and Paris, 1997b).

With this change in emphasis, techniques from *Multilingual natural Language Generation* (MLG) become crucially relevant. MLG is a further specialization of natural language generation (cf. Reiter *et al.*, 1995; Bateman (to appear)) focusing on the particular problems of producing texts automatically by machine in various natural languages. The relative merits of (machine) translation and MLG have been discussed by Kittredge *et al.* (1988), Rösner (1992), Kittredge (1992; 1995) and Hartley and Paris (1997a), among others. MLG has been employed with particular success in the bilingual weather report generation system FOG (Bourbeau *et al.*, 1990), and there is now a wide range of attempted or proposed further application areas; these include multilingual 'information brokers' (e.g. Glass *et al.*, 1994; Alexa *et al.*, 1996; and Schütz, 1996), the generation of letters replying to customers' mail (Coch *et al.*, 1995), of medical expressions (as in the EU LRE Project ANTHEM), of technical instructions (UK EPSRC Project J19221, Drafter: Paris and Vander Linden, 1996), of manuals (TECHDOC: Rösner and Stede, 1994), instructions for filing administrative forms (European Union LRE Project 62-009, GIST: Not and Stock, 1994) and of software messages (Spyropoulos and Karkaletsis, 1996), as well as Europe's own planned multilingual weather report system Multimeteo (EU LRE 1612). Generation is also beginning to play a more important role in proposals for Machine Translation architectures. The influential 'Shake and Bake' approach of Whitelock (1992), the minimal transfer approach of Copestake *et al.* (1995) adopted for Verbmobil (Kay *et al.*, 1994), and situation-theory influenced accounts such as that of Hobbs and Kameyama (1990) all place high reliance on their generation components, although the generation components actually employed in MT remain relatively restricted (i.e. compared to generation found in both monolingual and multilingual natural language generation systems).

Despite both the clear role for MLG that these developments indicate, and the need for accompanying engineering-scale work that would enable generation to play this role to greatest effect, current MLG efforts are compromised by a lack both of suitable linguistic resources and of development platforms for creating and maintaining those resources. As argued in Bateman (1997a), analysis-oriented resources and their development environments do not provide support for the kind of resources that are most appropriate to natural language generation – multilingual or monolingual. As an attempt to improve on this situation, we introduce here the KPML multilingual development environment. KPML offers generation projects:

- a growing set of standardized linguistic resources appropriate for real generation,
- a basic tactical generation engine for using such resources,
- a variety of highly focused debugging aids supporting efficient maintenance and development of further such linguistic resources,
- a palette of customization tools, and
- specialized techniques supporting typical multilingual work such as contrastive language development and automatic merging of independently developed resources for distinct languages.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Experience with KPML has already required us to reverse some commonly received views on constructing multilingual generation systems. It has shown that the development of separate tactical generators for distinct languages involves costly overheads that are often unnecessary, that the development of separate linguistic resources for distinct languages is usually an inefficient strategy for achieving generation in those languages, and that the development of small, particular application-restricted subgrammars is potentially a wasteful methodology. These strategies have until now largely determined how work in multilingual generation projects has proceeded.

KPML also provides a broader basis for multilingual representation than that found in other systems. The linguistic descriptions maintained by KPML are inherently multilingual in that they allow free sharing of similar (or congruent) descriptions across languages wherever this is empirically motivated or practically useful, but without needing to postulate any 'more general', common representation. Such resources can in turn support a variety of uses: ranging from strict monolingual generation (possibly, but not necessarily, starting from a shared interlingual semantic representation) to contrastive generation where the differences between language descriptions are explicitly utilized in the generation process. KPML remains neutral on the kinds and degree of multilinguality that are employed in any particular application of the general multilingual resources that it provides.

The KPML development environment is fully documented in Bateman (1997b); in this paper, therefore, our aims are the following:

- to provide an explanatory overview of the system, describing its basic aims and functionalities, and its approach to linguistic representation, multilinguality, and resource development,
- to present a detailed example of the kind of resource development that the system supports, and
- to discuss some early results of using the system and the issues raised.

2 KPML development environment: overview

In this section, we provide a basic overview of the KPML system. We first sketch the aims and functionality of the system as a whole, and introduce the particular form of linguistic resources supported. We then build on this to discuss the definition and role of 'multilinguality' within a system such as KPML. Since the theoretical motivations and consequences for linguistic description of this approach are set out at length in Bateman *et al.* (1997), we restrict the present introduction to just that necessary for the discussion and the presentation of early results in Section 4. Following the account of multilinguality, we summarize more fully the classes of linguistic objects supported by KPML and then set out one supported methodology for efficiently constructing linguistic resources for generation.

2.1 Aims and functionality of the system

KPML builds on the experiences gained with several approaches to producing grammar development environments for systemic functional resources and for natural

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

language generation, as well as on a series of trials in which the system was used for building linguistic resources for independent projects. The result is a convenient platform for the construction and maintenance of multilingual linguistic resources where interaction with the system is predominantly by combined mouse-click and graphical/textual information presentation. User commands are offered for loading definitions of (multilingual and monolingual) linguistic resources, displaying these resources in a variety of ways useful for development and maintenance, performing static integrity checks of the resources loaded, and using the resources for generation. Support is thus provided over the entire life-cycle of linguistic resources.

The report of the Expert Advisory Group for Language Engineering Standards (EAGLES) on development environments for linguistic resources lists several desirable aspects for development platforms (EAGLES, 1996, p. p117). These include selective viewers for parts of larger structures, automatic testing of grammars against test suites, versioning and multilinguality. KPML provides all of these functionalities from the generation perspective. In addition, particular points of emphasis of the system include:

- an integrated view of examples, generation and linguistic resources: resource maintenance is supported by extensive test suites which are interlinked with the resource definitions providing example-based on-line 'documentation';
- the possibility of combining graphical views of the linguistic resources with particular details of the generation process: generation debugging is graphically driven;
- a very high degree of modularity in the linguistic resource definitions;
- very extensive graphical and textual inspection of all aspects of the linguistic resources and their use;
- automatic resource management, including patch facilities for extending linguistic resources;
- provision of 'fast generation' modes;
- provision of structured and annotated 'string' generation to support hyperlinks and other application specific mouse-driven functionalities;
- provision of hybrid template and full generation;
- multilinguality throughout.

Large-scale sets of linguistic resources (grammars, lexicons, semantics-grammar mappings, punctuation rules) for different languages may be built up either from scratch or by automatic inheritance from other language resource sets; these may then be modified and extended as required. The KPML generation engine can be used as a black-box tactical generator accepting input specifications and producing marked-up strings; this core generator can also run as an independent 'generation server' process accepting semantic input from clients.

The intended users of KPML are linguistic resource developers and generation projects where practical generation is an issue. Considerable efforts have been made to make KPML appropriate for the needs of practical generation, with transparent user access not only to the linguistic resources maintained and developed, but also to 'fringe' activities such as defining punctuation rules and final string 'clean-ups' as

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

necessary for the spelling or graphological traditions of particular languages. Work in progress is also investigating the role of KPML as a teaching aid for both natural language generation and functional grammar descriptions.

2.2 Form of linguistic resources

The particular model of linguistic resources adopted by KPML is that of the Penman system (Mann, 1983; Mann and Matthiessen, 1985). This model has exerted a significant influence on generation generally and has been shown appropriate for a wide variety of text generation tasks; it is probably described in most detail in Matthiessen and Bateman (1991). The very large resources for English generation developed within the Penman project throughout the eighties provided extensive experience in issues of re-use, distributed development, and dealing with large-scale linguistic resources for generation. The formal power of the mechanisms provided are very limited and thus lend themselves well to a combination of linguistic coverage and practical utility. All of these experiences have now fed directly into the design of KPML.

Applications typically interact with resources created by KPML by providing semantic input specifications for the sentences (or other syntactic units) that they wish to have generated. Applications may either construct such specifications directly or allow a text planner to mediate. Input specifications are usually couched in the 'Sentence Plan Language' (SPL) (Kasper, 1989) developed for Penman, examples of which appear below. One of the principle aims of SPL relevant for practical generation is that it attempts to reduce the work involved in linking application concepts with a generation component as far as possible. The more straightforward approaches of, for example, Elhadad and Robin's (1996) SURGE component for English generation, Meteer's (1992) use of the MUMBLE component, or the German Verbmobil project (Quantz *et al*., 1994) place far more onus on the application to directly specify how particular domain concepts are to be expressed. This brings short-term flexibility but is less supportive of the re-use of semantics-grammar mappings.

To improve on this, SPL specifications make use of a hierarchy of semantic types, called an *Upper Model*. These types are motivated by their ability to control or constrain broad selections of grammatical features. Using a particular semantic type then allows input specifications to be formulated without resorting to low-level syntactic information. An application interfaces with the generation component by linking its domain terms to Upper Model concepts. Although this can be done in a number of ways, the simplest method is by enforcing direct subsumption. This sacrifices some flexibility in the generation possibilities, but is often sufficient because the mapping between Upper Model types and their grammatical realization is itself quite flexible, remaining uncommitted both to the syntactic category and to the particular syntactic pattern (thematic, transitivity, etc.) generated.[1] Upper Model

[1] There is sometimes a confusion between the terms of the Upper Model and some similarly named grammatical functions in standard systemic grammars (e.g. Actor, Goal, etc.): the latter are clearly much more constrained in their grammatical realization and are not suitable for interfacing with domain knowledge.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

```
(l0 / spatial-locating
    :speechact (a0 / assertion :polarity positive
                               :speaking-time t0)
    :reference-time-id t0
    :event-time (t0 / time)
    :theme d0
    :domain (d0 / object :lex dog
                :identifiability-q notidentifiable)
    :range (p0 / three-d-location :lex park
                :identifiability-q identifiable))
```

Fig. 1. SPL specification for the sentence "A dog is in the park".

types provide a convenient, often language-invariant, anchoring point for semantic specifications such as SPL and allow the grammar developer to concentrate more on grammatical issues; the motivation and use of an Upper Model in NLP is addressed at length in Bateman (1992b).

SPL specifications show similarities to Quasi-Logical Form (Alshawi *et al*., 1991) in terms of their degree of semantic abstraction and to minimal recursion semantics (Copestake *et al*., 1995) in terms of their decommitment from details of syntactic structure. However, as typical in natural language generation, SPL specifications contain more explicit 'semantic' information concerning desired communicative effects and textual context. A simple example of an SPL specification is given in Figure 1. Variables (`l0`, `a0`, etc.) are given semantic types (`spatial-locating`, `assertion`, `object`) and a set of properties and values (e.g. `:theme d0`). It is possible to move information back-and-forth between explicit representation in the SPL (e.g. `:identifiability-q identifiable`) and information inferred from domain or discourse model. Examples tend to contain complete specifications so that they are independent of external processing modules. The example in the figure therefore also selects some particular lexical items (by means of the `:lex` property). The specification freely mixes propositional content, interpersonal/interaction-relevant information (such as the type of speech act, the polarity, the time of speaking), and textual, discourse development information (such as information statuses, e.g. theme and identifiability).

Grammars in KPML are given as *system networks* built up, as in Penman, from disjunctions over communicative options (called *grammatical systems*) as defined in systemic-functional linguistics (SFL, e.g. Halliday, 1966, 1978). Since this kind of organization is actually an explicit representation of the alternatives for expression that are available, the generation process itself can remain very simple. For example, KPML's own generation engine uses the system network in order to construct 'augmented' surface strings (described below) by traversing the network repeatedly left-to-right, once for each grammatical constituent to be generated. In each grammatical system, one and only one *grammatical feature* (one of the system's disjuncts) is selected. Each such feature may bring a set of syntactic constraints to bear on the overall syntactic structure being generated. Allowable constraints include statements of linear precedence, immediate dominance, unification of functional constituents,

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

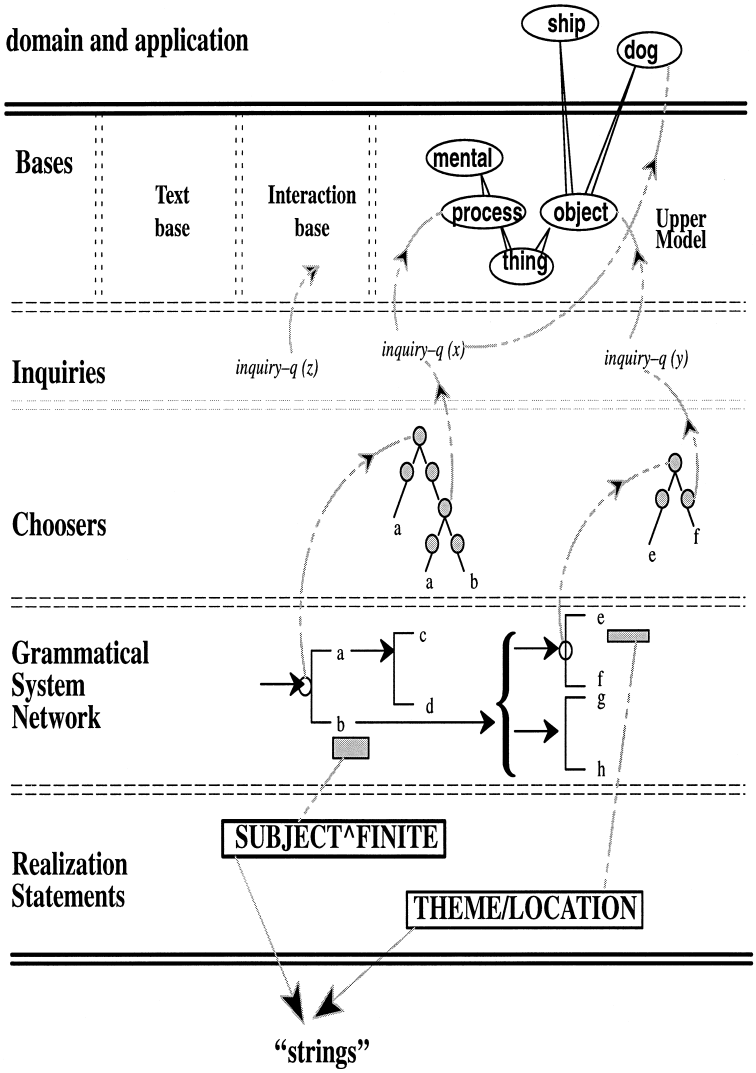


Fig. 2. Penman-style architecture for lexicogrammar, semantics, and their interrelationships.

and further type constraints to hold for subconstituents; an overview of the realization statements and of their place in a systemic description is given in Matthiessen and Bateman (1991, pp. 95-97). Generation is complete when sufficiently fine structures have been constructed to allow the insertion of lexical items (which may have been chosen at any time during the generation process).

The selection of a grammatical feature in a grammatical system is determined by a *chooser* for that system. The chooser makes its selection by traversing a decision tree of semantic *inquiries*. These semantic inquiries either ask questions of the SPL input (via, for example, Lisp functions making use of a small set of knowledge-representation independent interface functions) or obtain pointers

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

to subcomponents of that semantic input (for subsequent questions); there is no backtracking. Using McDonald's (1983) distinction, the architecture is therefore one of *grammar-driven* control; it is the grammar and its traversal that organizes all aspects of the generation process, inspection of the semantic input, construction of structure, etc. The use and interrelationships of these objects during generation is depicted graphically in Figure 2; a brief overview of this kind of generation is given in Bateman (1992a).

Although appearing relatively complex in terms of the distinct types of entities assumed, linguistic descriptions following the Penman-style architecture are in fact so constrained in terms of the interrelationships that are possible that they offer a useful and accessible decomposition of the information necessary for tactical generation. When resources grow to a realistic size (for example, the English and German grammars available each consists of approximately 700 grammatical systems – see Table 1), the main problem in their use is allowing a developer to secure ready access to the components involved during generation. We consider the KPML development environment to have effectively solved this problem by means of selective, dynamic viewing of the linguistic resources linked to their concrete use during any particular cycle of generation; this is illustrated extensively below. Furthermore, as described in the following section, the approach to multilinguality and multilingual descriptions that the general architecture allows is proving itself to be particularly powerful.

2.3 Systemic-functional multilinguality

The concept of multilinguality to which we make appeal throughout this paper is primarily directed at the *linguistic resources* that are developed rather than to particular applications. There are various positions that can be taken on the role and nature of multilinguality in an application, ranging from duplication of information in various languages (as supported by some web browsers for example), through on-the-fly 'translation' of various degrees of complexity, to strict localization of language-dependent information in readily exchangeable modules. Our discussion is deliberately restricted to a consideration of the representation of linguistic resources for generation. Focusing on generation avoids, as Kay (1996) notes in his overview of 'multilinguality', the very difficult and unsolved translation problem of requiring an interpretation of a text before an equivalent in some different language can be guaranteed. In general, this interpretation is given as the input to the generation process.

We are also not concerned here with any particular position on multilinguality taken *within* any specific MLG system. The KPML approach to representing multilingual linguistic resources aims to support as wide a range of such positions as possible, starting from a theoretically very broad notion of multilinguality *per se*. Thus, it is of little import whether multilinguality in the final generation component is achieved by employing fully independent sentence generation modules that use distinct grammatical resources to create distinct sentences from a shared semantic representation (a semantic interlingual approach to generation) or, alternatively, by maintaining distinct text planning and sentence generation components so that the

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

sentences of the texts generated in distinct languages might not even share their semantic specifications. The effectiveness and relevance of the tools and methodologies supported by KPML remains unaffected.

There are, however, several customization steps that may need to be carried out when moving from a general linguistic resource to final target application and these can naturally influence the resources delivered. For example, if it is decided (on engineering grounds) that it will be *required* that 'translationally equivalent' sentences are to be generated in distinct languages from a single semantic specification, then the mapping from semantic specification to string may need to be complicated in order to obtain acceptable results. This is because it is not in general true that a single semantic specification will naturally result in appropriate translation equivalents (cf., for example, Kay (1996) and Teich *et al.* (1996)). Forcing this to be the case means that the mapping from semantics to string must provide particular application-tuned linguistic realizations that may not be motivated in the general case. Practically, in terms of the KPML architecture for generation, this moves some of the work of producing SPL specifications away from the application (or text planner) and into the mapping from SPL specifications to strings. And, although the appropriateness of adopting application-specific mappings depends entirely on the flexibility of generation required in the target application, the detailed support that KPML provides for developing semantics-grammar mappings in general (essentially captured by the chooser and inquiry interface) can in any case be used to good effect.

The resources targetted by KPML are generally intended to be as use-neutral as possible (apart from their orientation to generation) and so do not make the assumption that translation equivalence is expressable at the level of shallow, linguistic-oriented semantics expressed by SPL specifications. The inappropriateness of such an assumption for the general case is also argued by results obtained in MLG projects such as GIST and TECHDOC. The 'equivalence level' adopted in both these projects lies prior to the generation of semantic specifications, which can then diverge for different languages. We will see the use of this in example sets below where multilingual conditionalization is also used at the level of semantic specifications. The general model is therefore that the resources developed serve as starting points for further application-specific customization of various kinds. Extensive and appropriate multilinguality is aimed for in these resources so as to rapidly provide substantial bases for varied applications in the natural languages required.

The idea of constructing such multilingual linguistic resources – generation grammars included – is certainly intuitively appealing for a number of reasons. Most straightforwardly, if resources for several languages can be combined so that commonalities are shared rather than re-represented then the description as a whole may become more economical regardless of subsequent use. This should then also reduce development times since results for one language can be 'inherited' by descriptions of other languages when there is overlap. Such resources might then be expected to support contrastive linguistic studies more effectively. To these benefits, Cahill and Gazdar (1995) also add the possibility of increased robustness in NLP systems since when information is missing for one language (such as a lexeme, or a

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

morphological pattern for a lexeme), it may be possible to deduce sufficient information via known commonalities with other languages; this approach was previously investigated within an MT system by Hajičová and Kirschner (1987).

There are also, however, a number of reasons to doubt the practical efficacy of such multilingual resources – the most direct being the actual degree to which languages allow common representations. If language descriptions quickly diverge so that a theoretically multilingual representation in fact becomes by and large non-multilingual then not much has been gained. There is also a justifiable concern that attempting multilingual descriptions may skew individual language descriptions inappropriately, i.e. that 'inheriting' a description from one language to another may in fact produce an unnatural description. Moreover, even if appropriate, a common language description may involve overheads for resource management and development since it must be ensured that a possible change for one language does not percolate to affect other language descriptions adversely.

As experiences with multilingual systems grow, it appears that the first of these doubts can now be discounted. Languages often do allow sufficient commonalities in their grammatical descriptions to bring significant economies. This has been shown in, for example, Alshawi *et al.* (1992) and Rayner *et al.* (1996) extensions of the Core Language Engine to obtain Swedish, French and Spanish from English, in the NEC Pivot MT system to obtain Korean from Japanese (Lee *et al.*, 1991) and French and Spanish from English (Okumura *et al.*, 1991), in TECHDOC (Rösner and Stede, 1994) to obtain German from English, in Jacob and Maier's (1988) adaption of the English generator MUMBLE to German, and several others. Significant reductions in development time are reported – a result also supported by our own experiences with KPML described below.

One remaining problem with these previous approaches, however, is their response to the second area of doubt. The typical development cycle is that resources are taken from one language to provide a starting point for description: this starting point is then adapted independently of its origins to produce a distinct *monolingual* description for a target language. Such 'code scavenging' approaches solve the problem of possibly deleterious interactions across language descriptions at the cost of restricting multilinguality to the initial stage of resource development. Since there is no theoretical relationship between derived grammars, subsequent development and extension proceed as for monolingual grammars and no economy of representation is achieved. This problem has recently been recognized and partially addressed by Rayner *et al.* (1997), who describe a technique for preserving contact between derived representations for distinct languages. This technique relies, however, on the languages handled being 'closely related' since the derived representations are related back to a generalizing common description.

A less ephemeral and restricted multilinguality can be achieved by simultaneously satisfying the competing goals of *integrity* and *integration* of linguistic resources (Matthiessen *et al.*, 1991; Bateman *et al.*, 1997):

- **Integration** of different languages is required so that commonality is separated from particularity and re-used; in other words, the resources should maximize

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

the *factoring* out of generalizations across the languages of the system and the particulars of individual languages in the linguistic resources.

- **Integrity** of each individual language is required so that it can be used, maintained and developed separately; that is, resources should support *views* both from the standpoint of their multilinguality and from the standpoint of the individual languages.

Providing for both of these properties simultaneously has determined the approach to multilinguality implemented in KPML. Linguistic descriptions are made multilingual by allowing language conditionalization for all the types of linguistic 'objects' used in the Penman-style architecture and illustrated in Figure 2. Language-particular linguistic specifications are achieved by defining 'partitions' in the resources that hold for a subset of the languages covered. A partition is a conditionalization of the information maintained according to one or more languages; the information that is valid only for a particular language is the collection of specifications given within all the partitions that mention that language.

Multilingual conditionalization holds not only for grammatical systems and realization statements, but also for the mapping from semantics to grammar defined by choosers and inquiries and for semantic specifications. *All* of this information is subject to the principles of multilingual representation set out at length in Bateman *et al.* (1997) – that is, there is no theory-enforced commonality at any level where conditionalization is not possible or is assumed to be unnecessary; this is crucial for allowing KPML to support both 'interlingual' approaches to MLG and more sophisticated models involving semantic variation.

This kind of multilingual organization brings significant gains for grammars organized for generation. Since the grammatical organization adopts the dependency relationships between paradigmatic alternations as backbone (yielding the system network), it is usually influenced strongly by the semantic distinctions that the grammar is capable of realizing. This means that the system network organization can often be held common across languages even when the grammatical distinctions present are expressed structurally cross-linguistically in quite different ways. A simple example is shown in Figure 3; this shows a fragment of the system network for the multilingual English, Chinese and Japanese clause grammar discussed in Bateman *et al.* (1991). It contains both common grammatical systems and grammatical system partitions valid only for English, Chinese and/or Japanese.

Even in this simple example there are clear commonalities – or, better, 'congruences' – among the three languages even though they belong to three structural-typologically distinct language groups. This is shown in Figure 3 by the fact that the grammatical systems of Chinese, English, Japanese (and many other languages) all share the options 'declarative' and 'interrogative'; this information can therefore be factored out as common. These options can nevertheless be realized (i.e. expressed structurally) in very different ways in these languages because the realization statements capturing the differences are theoretically separate and can therefore also be factored out as being particular to each language. This preserves the system network organization as congruent and shared, thereby simplifying both the task

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

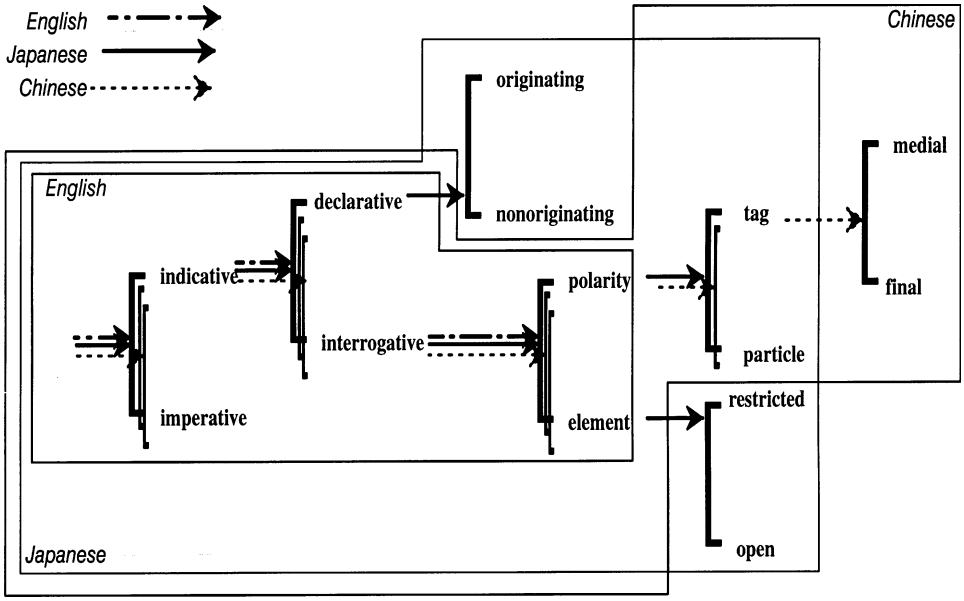


Fig. 3. Multilingual system network with conditionalized partitions.

of maintaining a working mapping from semantics to strings and that of keeping parallel grammars 'in step' as descriptions change. The factorization of realization statements is illustrated further in Figure 4, which shows the structural (syntagmatic) consequences of two paradigmatic features ('declarative' and 'polarity') drawn from the multilingual system network of Figure 3.

Note that it is important that the 'meta-information' concerning the languages for which the language partitions are valid, although an intrinsic aspect of the representation, is maintained separately to the details of the linguistic description itself. Multilingual representations such as the rules of, for example, Dik (1992) or Cahill and Gazdar (1995) invoke additional parameters or rule-naming conventions for the languages for which a rule is applicable. This complicates the process of adopting more or less contrastive views of the linguistic resources and hinders integration. It can even, as in some cases in Dik's (1992) representation, encourage unfortunate compromises in the integrity of individual language descriptions (cf. Bateman 1995 and Section 4).[2]

It should now be straightforward to see how the demand for *integrity* is met in the KPML architecture. The overall multilingual resources provide a source from which language-specific views can be built. The monolingual view contains any common, language-neutral resources and those parts of the resources that are within partitions specific to the language in question: it can simply be 'lifted out' from the multilingual resources. At the same time, however, there is a multilingual view where

[2] In fact, the KPML approach to multilinguality is more reminiscent of the strict separation allowed by the 'tlink' mechanism of the EU Acquilex project (cf. Copestake,1994). This relationship is followed up in more detail in Bateman *et al.* (1997).

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

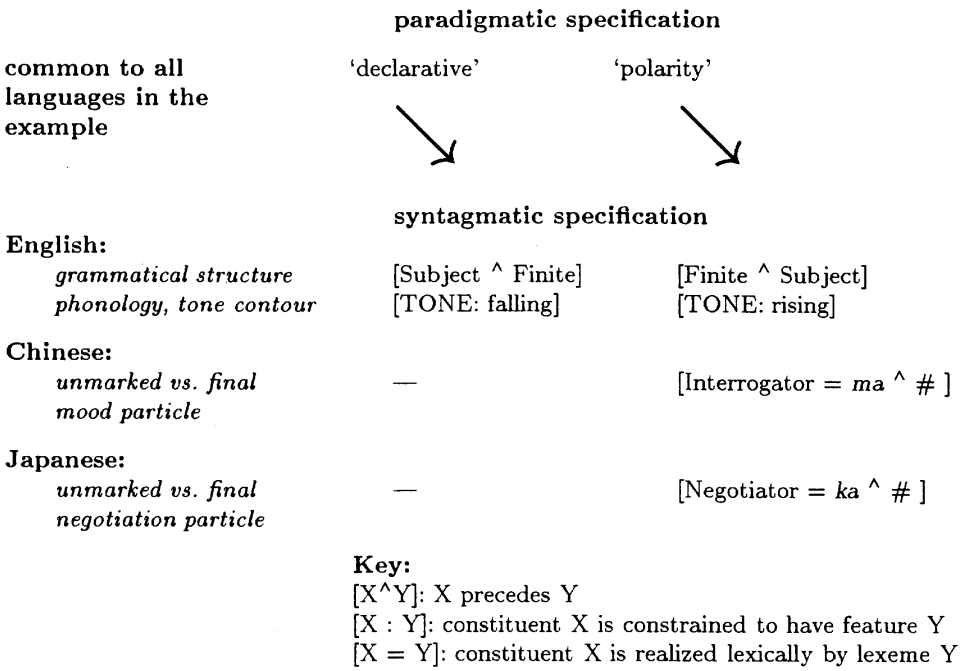


Fig. 4. Intra-stratal multilinguality: common paradigm, different syntagms (taken from Bateman *et al.*, 1997).

congruence across (some set of) languages is represented simply by congruence in the specifications, without the need for special mappings. Resource *integration* is therefore also a natural property. As a consequence, the approach can support the full spectrum of multilingual methodologies: ranging from independent development for individual languages through to 'transfer comparison' (cf. Teich, 1995), where one language is taken as the starting point for the description of another. Contrastive linguistic work is encouraged and this supports convergence and re-use of resources. KPML therefore improves on the code scavenging methodology introduced above in three ways: first, by providing detailed software support for such development; second, by maintaining the relatedness of source and target so that integration is preserved; and third, by extending the languages among which similarities may be recognized beyond those related by structural typology.

2.4 Extensions to the input specifications

KPML provides a number of extensions to the SPL input specification as inherited from the Penman system. These improve the utility of the specifications both for multilingual work and for practical generation.

The first extension follows straightforwardly from the multilingual conditionalization provided, i.e. it is possible to provide input specifications that are partially conditionalized according to language. This means that it is possible to maintain contrastive example sets where 'translational' equivalents are maintained as single ex-

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

```
((p0 / :english directed-action & :lex destroy
                                & :actor e0 & :actee b0
      :japanese nondirected-action & :lex collapse
                                & :actor b0 & :cause e0
      :tense past)
 (e0 / object :lex earthquake)
 (b0 / object :lex building))
```

Fig. 5. Example of language conditionalized SPL-semantic specification.

amples even though the semantic specifications required for them are non-identical. This covers standard examples from the Machine Translation literature where complex or semantic transfer might be required. An example is the contrast exhibited in a translation pair such as:

Japanese:	*jishin-de*	*tatemono-ga*	*kowareta*
	earthquake	building	collapse
	Cause	Actor/Medium	Process: past
English:	*The earthquake*	*destroyed*	*the buildings*
	Actor	Process: past	Goal/Medium

Whereas it would be possible to use the KPML tools to construct linguistic resources for English and Japanese where these contrasting sentences would be generated from a common semantic representation (as might be done in interlingual MT), we are not required to do so. In fact, these two sentences are most naturally generated using different SPL specifications – Nagao *et al.* (1988) suggest that this type of systematic contrast can be explained by postulating a process ("BE") orientation for Japanese as opposed to an action ("DO") orientation for English and thus represents a genuine 'semanticization' difference between the two languages. KPML therefore provides the further option of preserving the 'shallowness' of the semantic specifications employed, thereby maintaining simpler semantics-grammar mappings. This then relies on a prior process of text planning (presumably involving some notion of perspectivization) to select the semantics appropriately.

While developing corresponding grammatical resources, then, both of these example sentences could be generated from a conditionalized SPL specification of the form given in Figure 5. In this figure, language conditionalization is indicated by a language specifier (e.g. `:japanese`) which applies to all items following that are linked by an '`&`'; unconditionalized elements apply to all languages considered. The specification of the figure therefore states that whereas for English we are concerned with a semantic type `directed-action` which is a configuration consisting of an `:actor` and an `:actee`, for Japanese we are instead concerned with a `nondirected-action` which is a configuration of an `:actor` and a `:cause`.[3] There

[3] For ease of exposition, we have again specified lexical items in Figure 5 directly and have used the 'SPL macro' `:tense`. Details (particularly information statuses for selecting determiners and particles) have also been ommitted. See Teich *et al.* (1996) for further examples of conditionalized SPL specifications.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

are, of course, various other ways of conditionalizing the information presented, and questions concerning both more systematic cross-language and intra-language relations (e.g. is 'destroy' simply a further directed-action way of expressing 'collapse': "make to collapse"?) are immediately raised. Whatever particular treatments are selected, however, KPML provides general means for ensuring that the intended grammatical realizations follow from the desired semantic specifications while investigating the treatments possible.

The second extension allows for the straightforward generation of sentences involving hyperlinks for use in constructing hypertexts. This is supported by a simple addition to the SPL notation. Any semantic element in the input specification may receive the additional keyword `:annotate`. The value of this keyword is then associated with the surface constituent that realizes the semantic element annotated. With KPML, the actual result of generation is a lean-structured string that preserves the syntactic structure to a greater or lesser extent as determined by the user. The annotation is passed on unchanged to this lean structure. One simple use, therefore, is to provide WWW URLs as annotations and to pass the generated structure through a postprocessor that wraps appropriate HTML around the annotated constituents. Similar postprocessing is used for preparing the mouse-sensitive string presentations used in the KPML interface itself; these can be straightforwardly adapted for more application-based presentations and incorporation in graphical user interfaces.

The final extension provides for mixed full and template based generation. Although generating texts using templates is most usually considered as a practical shortcut rather than as text generation proper, this view is incorrect. It is not realistic to always generate from scratch all parts of a required text. Any real text will usually contain parts that have been written previously, or are fixed externally (by style, convention, or legislation). A full text generation account must naturally include such possibilities. In terms of linguistic theory we can state that textual *instantiation* is as an integral a part of the account as any other. This opens up the door to a proper theoretically based inclusion of notions of text history (i.e. text chunks that have been generated before: either by human text producers or automatically). KPML extends SPL further to include template specifications by defining the new pseudo-semantic type `template` and the SPL keyword `:pattern`. Any such component of an SPL specification creates the structure given in the pattern without traversal of the system network defined by the loaded linguistic resources. The template pattern may include slots where further full generation (or further template patterns) is required.

Since template generation has several clear disadvantages – especially when used for multilingual generation – it will probably be common that the template facility is used together with the language conditionalization facility. Figure 6 is an example that combines multilingual conditionalization, annotations and templates in order to generate phrases of the form "Host X is unreachable" and the corresponding German "Host X ist nicht erreichbar". The expression for X is in both cases generated normally (i.e. as the loaded linguistic resources would generally generate a component with semantic type `host`) and will be associated with information for presentation as a hyperlink.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

```
(e / template
     :pattern :english ("Host " n " is unreachable")
              :german  ("Host " n " ist nicht erreichbar")
     :actor (n / host ...
               :annotate "http://www...."))
```

Fig. 6. Example multilingual template generation specification.

2.5 Linguistic objects and operations

User interaction with KPML is essentially object-oriented: analogously to, for example, Smalltalk or the Advanced Language Engineering Platform (ALEP) (Simpkins *et al*., 1993), particular objects permit particular operations. For example, a grammatical system allows operations such as inspecting, loading and modifying. Objects reflecting a wide range of linguistic granularity are maintained; there are both large-scale objects (such as a resource set, a grammar, and a functional region of a grammar) and very fine-scaled objects (such as the individual elements of realization statements used within grammatical systems). Operations can be parameterized in various ways: according to linguistic object, according to language, according to grammatical region.

The principal operations supported by KPML are as follows:

Loading**:** the constitutive objects of a designated class (i.e. all objects for a designated linguistic 'resource', or the systems, choosers, inquiries of a designated 'functional region') are read into the system and become part of the loaded linguistic resources; for most classes of object it is not necessary for a user to know the files of origin for the object definitions affected.

Writing**:** the constitutive objects of a designated class are written out to the file system; again it is not generally necessary for a user to know which files are created by this operation. The Loading/Writing operations are complementary, creating the required directory structures on writing and assuming these structures on loading.

Graphing**:** the designated object is shown in a mouse-sensitive graphical form, showing the interrelationships among its constitutive sub-objects. This is most commonly used for subnetworks drawn from the system network definition as a whole (consisting of collections of grammatical systems), choosers and generated syntactic structures. Significantly for debugging and maintenance, most graphing operations also allow *the actual options instantiated during generation* to be highlighted in the graphical display of any graphed object; this is the most common starting point for maintenance and debugging.

Inspecting**:** the designated object definition is shown in a mouse-sensitive textual form in the KPML inspector window.

Modifying**:** the designated object definition is brought into a GNU Emacs editor buffer where it may be modified; on return from the editor buffer the modified definition is automatically added to the loaded definitions active.

Editing**:** the designated object must be brought by the user from its definition file into an editor and altered.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

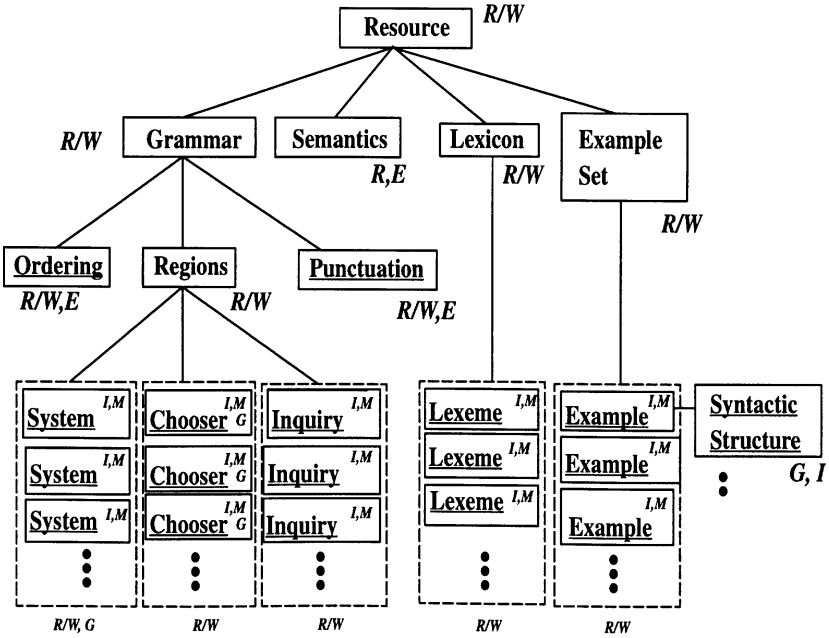


Fig. 7. Linguistic objects and operations supported by KPML.

The objects supported and their hierarchical constituent relationships, as well as a summary view of the operations relevant, are set out in Figure 7. The types of objects generally follow closely those of the Penman-style of linguistic resource description introduced above. There are also some additional objects that represent the *results* of generation: for example, the generated string and its underlying constituent structure. These may be inspected, graphed, and, via inclusion in an example set, written and loaded. They cannot, however, be directly modified since they are only meaningful with respect to a particular generation history. Finally, some meta-information concerning a resource set is also maintained – including, for example, declarations of the fonts to be used for a given language allowing development in non-Latin writing systems and provision for the use of external morphology components.

In addition to the constituency relations shown in Figure 7, there are also relations between objects. These cross-links follow directly from the use of the linguistic objects during generation. As we have seen (cf. Section 2.2 and Figure 2), a grammatical system typically has a chooser which in turn consists of several inquiries, each of which has an *inquiry implementation* that is responsible for querying knowledge sources, classifying concepts, etc. The existence of such object cross-links supports the use of *information chains* by which a resource developer may obtain information concerning aspects both of the resource definitions and of their deployment during

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

generation. This makes it straightforward, for example, to start from a particular generated example and to follow an information chain from a substring to its grammatical constituent to the path of grammatical features through the system network responsible for that constituent to a particular grammatical system to a particular chooser to a particular inquiry which caused a particular inquiry implementation to be triggered with particular actual parameters: each transition is achieved by a single mouse-click. Navigating all components of the resource architecture as set out in Figure 2 is thus rendered straightforward regardless of the size of the resources maintained; we show this process in action in Section 3.

2.6 Working with KPML*: resource maintenance and development*

One of the main tasks that an environment such as KPML has to support is the ongoing verification that the resources defined do what they are supposed to do: That is, that a correctly formed semantic specification will lead to an appropriate linguistic realization of that specification in the desired languages. The principle means adopted to achieve this is by supporting extensive test suites, or *example sets*, for any resource set released. A test suite is constructed by generating from a wide range of semantic specifications, attempting to cover as many components of the grammar as is possible. Developing such test suites relies heavily on the generation functionality of KPML and the extensive resource debugging aids provided.[4] Extensions to a linguistic resource should then bring their own additions to the standard example sets as well as maintaining the input-output behaviour of the existing examples. KPML provides tools specifically aimed at facilitating this type of consistency check. The existence of example sets also provides a kind of on-line documentation for any resource set for which an example set exists. KPML allows examples to be selected according to the grammatical features they deploy and can display the generated example strings with the constituents containing selected features highlighted. This provides a fast and convenient method of ascertaining concrete examples of particular areas of an existing linguistic resource. The highlighted constituents can also serve as the starting point for information chains. Developing resources for a new application domain typically proceeds by constructing a set of examples (the 'target example set') which span the grammatical phenomena necessary for the sublanguage to be generated. This may involve extensions to existing resources, which necessitates cycles of rerunning already existing example sets.

KPML provides explicit support for a continuum of approaches to creating a linguistic generation resource for languages not previously covered. This appears to match well with the kinds of different approaches different developers prefer as well as with the approaches best suited for differing available development times: that is, the approach best suited to producing a generation resource is different dependent

[4] This builds on the maintenance experience with the Penman project: the first example set for the Nigel grammar of English was constructed by Lynn Poulton at ISI in 1988 and consisted of over 100 example sentences that together spanned all of the 1200 or so features used in the grammar.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

on whether one month is available or six. The trade-offs against time are the normal ones of breath of coverage and quality of the linguistic motivation. Both suffer when the time available is reduced, but an appropriate development methodology can in both cases minimize the damage.

One effective development cycle for resource development is the following, which is the KPML equivalent of the 'code scavenging' approaches mentioned above. The starting position is generally that some set of resources exist for one or more languages and that it is desired that resources for a further language (or languages) are to be added with minimal effort, and maximal re-use, of those existing resources. Subsequently:

1 A 'base case' is created for the new language(s), maintaining the integrity of the existing resources. This can be done either for individual functional regions from existing resource sets, or for broad sweeps across the language, e.g. we can theorize that Dutch transitivity is congruent with that of English whereas Dutch agreement is congruent with that of German (see Degand 1996 for such an experiment), or we can hypothesize that the organization of German overall is highly congruent with that of English (see Rösner and Stede 1994 for a system built on this assumption, and Hauser 1995 for a similar example involving Japanese and English), etc. Here both the joint development history of the multilingual resources and the very detailed resource debugging functionality that KPML provides play a crucial role. The former reduces the amount of reconciliation that has to be done; the latter makes that reconciliation a feasible task.
2 The new resource set is then progressively refined to cover the desired language (or, more usually, the desired register, or sublanguage, of the desired language), working towards covering the target example set. Initially, this involves replacement of structural constraints and closed-class lexical items, but can equally include semantic alterations. This latter is crucial since, as we have suggested above, it is still often not possible (or even desirable) to maintain semantic specifications across languages. Refinement of this kind is a highly constrained and guided process, far removed from code scavenging. Although the new resource set is completely separate from any of the original language resources and so can be changed freely without compromising those resources in any way, its common origin assists contrastive work and subsequent automatic re-merging. The result of this step is a system capable of generating the desired language at least approximately.
3 The debugged new resource set may be re-merged with the general multilingual resource set if required. This adds the new language to the complete generation functionality of the system. Alternatively, the resource set can be left as a new monolingual resource, or be combined with any subset of the original languages covered for use contrastively (as attempted in more advanced/experimental generation work where the *contrasts* are explicitly used to trigger generation strategies: (cf. Zeng, 1996)). From this point on the new resources may be used for generation.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

4 During extended use, it is typically the case that further refinement, additions, and corrections for all languages maintained will be required. It may also occur that as time, personnel resources and interest allow, individual regions of the new resource set can receive more intensive linguistic description *internal* to the language being described. This provides a means by which a linguistic resource can move away from a description that is essentially based on a different language and towards one that may be more motivated from within the new language itself. The methodology progressively replaces the approximate linguistic resources with monolingually motivated linguistic views and, finally, this monolingual view can feed back into the general multilingual resources – perhaps suggesting alterations of some of the treatments found there. Crucially, throughout all of this refinement, generation remains possible in the new language as set out in step (3). The methodology as a whole therefore combines fast production of generation resources for target languages, while maintaining (and supporting) the subsequent linguistically motivated refinement of these resources.

Stages (2) and (4) can call for varying amounts of contrastive work and the results can be highly synergetic. For example, to take an actual case, in constructing an account for verb-bound prepositions for French, it might be noted that a similar task is carried out for the separable verb prefixes for Dutch; accordingly, importing the relevant 'region' of the Dutch resources into the resources for French is an operation supported by KPML. The resulting treatment is, however, not particularly French-specific and so suggests a similar importation for English (which formerly lacked a description of verbally bound prepositions). This is again supported by the KPML resource maintenance operations, as is the consistency and integrity checking of the individual language resources overall when extended by the imports.

All of the operations required in the development cycle just described are directly available in KPML, most often by simple combinations of mouse-clicks. Several methodologies of the kind illustrated in this section are supported by KPML and particular developers tend currently to favour differing working practices – there is clearly much here still to be learnt.

3 Developing resources with KPML: an example

In this section, we sketch very briefly some of the development, debugging and maintenance techniques supported by KPML, illustrating them by means of an example. We will consider the simple case of establishing a grammatical resource capable of generating some German prepositional phrases by means of transfer comparison from English. The example is overly artificial but has the advantage that its details can be understood without further descriptions of the linguistic resources in use. It also nevertheless shows exactly the kind of operations that would be used with more realistic resource development – especially with the additional assumption that the developer might not know how the resources concerned are organized prior to attempting the task. Consider, then, English spatial prepositional phrases such as

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

in the park, *opposite the house*, *on the roof*. The example task is to create German resources that produce corresponding phrases in German, taking as a starting point equivalent semantic specifications. As noted above, this latter assumption can be relaxed if it appears empirically motivated to do so.

This particular task is by and large quite straightforward, since the syntactic structures are congruent between English and German. They therefore allow a direct construction of German resources as set out in step (1) of the methodology given in the preceding section, i.e. we load our general grammar of English and request KPML to write out a version of that grammar marked for use as a German grammar rather than as an English grammar. Following this operation, an input specification – for example, the SPL used as a first example in Figure 1 above – will generate the same result regardless of whether we request the generation language to be English or German. Prior to the step of creating a 'German' set of resources, requesting any other language would have produced a null string (plus some warnings) corresponding to the fact that no appropriate grammar resource had been defined. A replacement of lexical items (or, even simpler, a conditionalization of the 'spelling' of lexical items) would then already produce a first approximation of the desired German sentence.

One exception, however, to the simple syntactic congruence between English and German arises for the equivalent to the English *opposite the house*; this must be rendered as the German *dem Haus gegenüber*, i.e. the order of preposition and its object must be reversed. We will sketch how KPML allows us to easily locate the exact point where this statement of noncongruency between German and English should be made.

First, we need to find which component of the resources is responsible for the structure in English (or, equivalently at this stage, in our newly created image of English that is being modified to become German). The simplest way to do this assuming that one does not even know where approximately to look (which quickly ceases to be the case when working with the resources of course, allowing more direct inspection methods to be adopted) is to find similar examples in the provided example sets. One might look for occurrences of "opposite" in the examples, or for similar prepositions, etc. Having found these, one can opt to display the corresponding example strings in the KPML development window: such strings already have a mouse-sensitive constituent structure that was formed when creating the example and this can be displayed in a form such as:

```
(((The ) (church )) (is ) ((opposite )((the )(house ))).)
```

Clicking on the relevant constituent brings up a menu of options for inspecting the constituent in more detail, as well as the decisions and resources responsible for its creation. These include the grammatical structure involved and the traversal path through the grammar network (or *selection expression*) that created that structure.

These are shown in the two windows on the lefthand side of Figure 8: the fragment of the constituent structure of concern here is shown in the upper left as a straightforward functionally annotated structure (cf. Halliday, 1985), while the selection expression responsible for this structure is shown below this. The order of

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

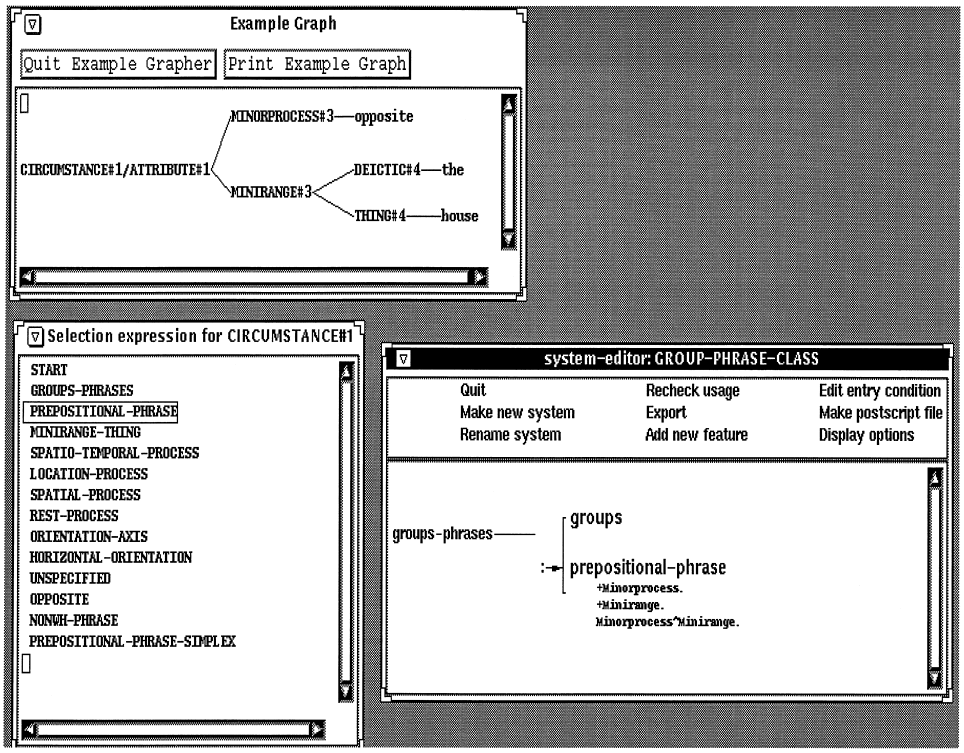


Fig. 8. Example selected constituent structure, traversal path responsible for that constituent, and a selected grammatical system.

features in the selection expression reflects the partial ordering of features defined by the system network. We now need to find the particular features on this traversal path that were responsible for the aspects of the phrase that are noncongruent in German, i.e. the ordering and the lexical selection 'opposite'. There are various ways to do this: one is to ask directly concerning the ordering and lexical constraint information of the nodes in the syntax tree (by selecting from a menu obtained by clicking on the node required). Doing this for the prepositional phrase node (i.e. the one functionally labelled Circumstance/Attribute at the root of the syntactic fragment shown), informs us that the ordering is introduced by a constraint associated with the feature 'prepositional-phrase' and the lexical selection is achieved by a lexicalization constraint attached to the feature 'opposite'. The grammatical systems involved can then be inspected by clicking on their features in the selection expression display. Thus, selecting the feature 'prepositional-phrase' brings up the grammatical system shown in the lower right of the figure. Grammatical systems are shown in the standard systemic notation, with the 'input conditions' (i.e. the less specific types) on the lefthand side and the 'output features' (i.e. the defined subtypes) on the right. The structural constraints attached to any feature are also shown: so here we can confirm that the feature 'prepositional-phrase' indeed imposes the ordering constraint that

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

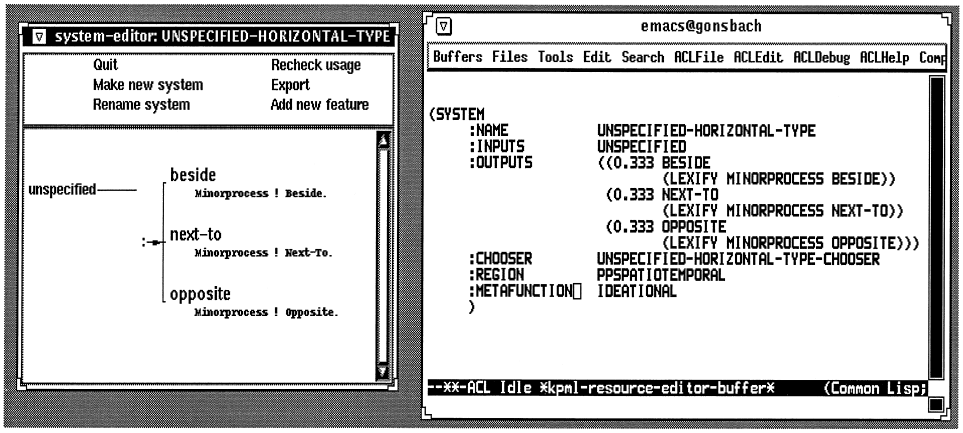


Fig. 9. Two editable forms of display for the grammatical system defining the feature 'opposite'.

the preposition (the `Minorprocess`) precedes its object (the `Minirange`) by virtue of the linear precedence statement: `Minorprocess^Minirange`.

It now remains simply to alter these constraints for German. This can either be done contrastively by adding into the general multilingual definitions constraints that are conditionalized only for German, or monolingually by changing the German grammar created automatically on the basis of the English grammar. This can also be done directly from the selection expression list. For example, turning our attention to the other feature involved in the noncongruence, 'opposite', we can edit the grammatical system responsible either by changing the displayed system or by selecting a further menu option that brings up the textual definition of the grammatical system in an associated Emacs buffer. These two possibilities are shown side-by-side in Figure 9: the grammatical system is shown graphically on the left and in its textual form on the right. In both cases, it is the lexical realization of the preposition constituent (`Minorprocess`) that is to be changed: perhaps simply by altering the lexical item employed (e.g. to be *gegenüber* rather than *opposite*). The lexical item can also be created either by adapting the corresponding English item, or by referring to existing lexical resources for German that might be available.

The problem of the noncongruent ordering is more complicated since it is not possible in German to decide which ordering is appropriate as soon as it has been decided that a prepositional phrase is to be generated: both orders are possible depending on the particular prepositions (and their readings) selected. Probably the two most straightforward options are: (i) to make the pre-nominal preposition ordering a default for German and only to give other orderings explicitly; and (ii) to divide German prepositions into two classes and allow the ordering to be determined by the class selected. In the first case, the strict ordering constraint is simply removed from the 'prepositional-phrase' feature and an explicit ordering (`Minirange^Minorprocess`) is added to the feature 'opposite'. In the second case, if a semantic motivation for the ordering can be found, then this can be appealed to

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

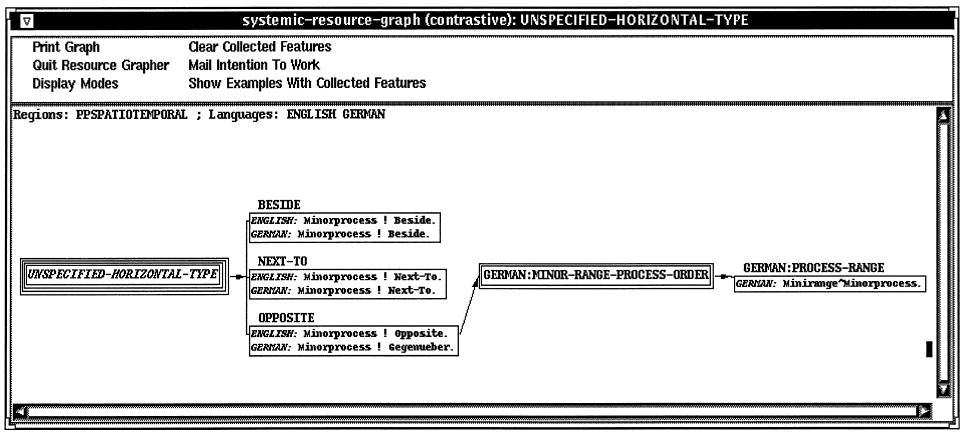


Fig. 10. Contrastive bilingual system network fragment showing the divergence of the English and German resources.

as a further grammatical system dependent on 'prepositional-phrase', otherwise the distinction can be encoded grammatically – by placing the required ordering constraint at a position in the system network dependent on the particular preposition that has been selected – or lexically – by introducing a lexically-dependent grammatical alternation and having the lexical entries for German prepositions indicate which ordering they require. Examples of all these strategies can be found in the example sets standardly supplied with the existing KPML resources; the particular places to examine are located using exactly the same strategies as described here for locating the required prepositional phrase information, i.e. by following information chains from example strings that appear to exhibit similar phenomena to that being investigated.

If we take the grammatical, non-semantic motivated encoding, then the alterations just proposed for German can be represented as shown in the system network graph shown in Figure 10. This is a *contrastive* graph that KPML produces when resources for different languages are to be compared. Grammatical systems and their features are shown conditionalized if they are not defined for both languages being displayed. Shared grammatical systems receive multiple heavy boxing to distinguish them from non-shared systems.[5] The system network fragment (again produced by a menu option directly available for each feature presented in the selection expression) presents a treatment that is sufficient for the 'opposite' case, although not particularly attractive for the non-opposite case.

Rerunning the semantic specification associated with the original example "The church is opposite the house" with German as the specified generation language then produces the constituency marked-up string:

```
(((The ) (church )) (is ) (((the )(house ))(gegenüber )).)
```

[5] This is the monochrome representation; on colour displays the differences are presented rather more perspicuously by colour-coding.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

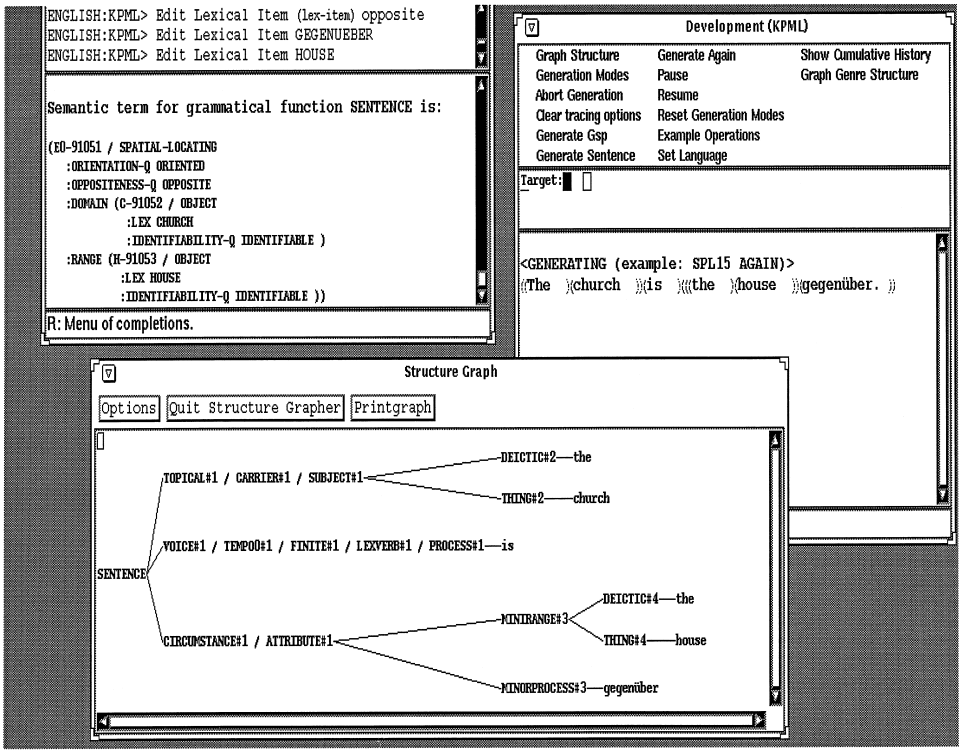


Fig. 11. Semantic specification, and corresponding generated string and structure with the modified resources.

Generating with English as the requested language naturally produces the original English sentence unchanged.

The semantic specification, the generated string, and the complete grammatical structure for the 'proto-German' sentence is shown in Figure 11 (left, right, and bottom, respectively). The other features necessary for producing an appropriate German sentence, such as altering the remaining lexical items, adding case control to the definite determiners and perhaps allowing a more specific main verb (e.g. "liegen", "stehen"), may be performed in exactly the same manner: by locating the features responsible for particular properties of a constituent and altering or adding resource distinctions as required.

Since the inspection of the resources is organized concretely around those distinctions that are involved in a specific phenomenon, the size of the resources overall plays less of a role. This accounts for one of the significant improvements in resource management that KPML provides. Each cycle of change/addition to the resources described here is a matter of seconds, with single mouse-clicks guiding the location of the points of incongruence. This produces a workable generation component for particular phenomena very quickly: the breadth and accuracy of the linguistic descriptions developed can then be re-examined and replaced as time allows. Given that such development automatically maintains the necessary mappings between a

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

standardized semantic representation as input and the generated surface strings, it can be seen that developing separate tactical generators for individual languages is potentially quite wasteful: the need to coordinate the input representations required by each component is naturally subsumed by the KPML development methodology.

Debugging resources proceeds in exactly the same way as shown in the above example. An erroneous resource specification can be considered as an 'incongruence' with respect to a desirable resource definition. This can then be pursued by following information chains until the point of divergence between desired and produced is located and then by altering/adding as appropriate. Again, the absolute size of the resources involved is less significant.

It has only been possible here to present a small fraction of the development tools KPML makes available. For example, it is possible to ask directly where some lexical entry is used in the grammatical resources, to view the traversal of the grammar network dynamically as it occurs, and to go directly to a graphical representation of some selected region of the grammar concerned with a particular area of semantics (e.g. the region 'PPSPATIOTEMPORAL' for the example used in this section; cf., also, Table 1). This latter option particularly enables more global changes to be made for a meaningful segment of the resources as a whole. The same techniques also apply equally when changes are to be made, for example, in the mappings from semantics to grammar. If it is decided that it is there that languages diverge, or even in the input semantics, then the appropriate changes are supported as shown here for grammar. Full details of the possibilities are given in the documentation (Bateman, 1997b).

4 Early results of using KPML

In this section, we present some preliminary results of using KPML for resource development and generation, briefly discussing some of the issues that are raised.

4.1 Two previous approaches to multilinguality: a contrast

Space precludes a thorough review of previous and current approaches to multilingual generation and the representation of multilingual resources – although an almost exhaustive overview is given by Bateman *et al.* (1997). We limit ourselves for current purposes to a brief comparison with just two significant approaches proposed for achieving multilinguality: the combined multilingual lattices of Kameyama (1988) and Kameyama *et al.* (1991), and of Cahill and Gazdar (1995). Both approaches are clearly related to our account in that they adopt unified subsumption lattices containing information from more than one language.

In Kameyama *et al.* (1991), all linguistic resources are placed within a single subsumption hierarchy, allowing generalizations across linguistic information both within and across languages. Integrity of individual languages must then be achieved by labelling all sorts in the hierarchy additionally by the language(s) for which they hold. An example of a shared grammar from Kameyama (1988) is shown in

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

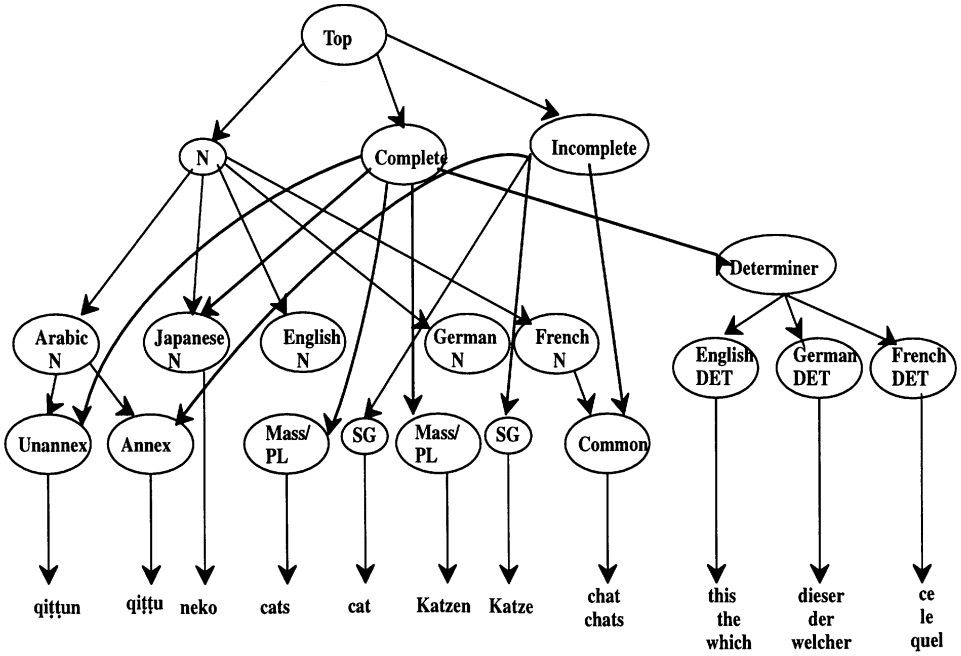


Fig. 12. Shared grammar from Kameyama (1988, p. 195).

Figure 12. The lattice defines inheritance over structural templates, which are then merged to give a grammar. This is, however, problematic in a number of ways.

First, since the grammar of an individual language may be 'constructed' by inheritance from categories that are defined as applicable for other languages, development work in those other languages may have unfortunate side-effects. This requires effective orchestration of development efforts and accordingly creates a management overhead. In short, integration has been achieved, but integrity is threatened. The interpretation mechanism for multilingual systemic networks differs substantially. Grammars are not created by inheritance: the multilingual congruence of a systemic network is an *organizational* feature of the resources, not an *operational* property of those resources. Thus, shared features are re-used across languages, but any language-specific changes to a specification do not have automatic consequences for languages not involved in the change. This is crucial for serious distributed development work.

Another problem is the nature of the distinctions maintained: these are essentially structural and do not give priority to *functionally motivated* disjunctions as required for a generation grammar; this makes its organization to a certain extent arbitrary with respect to semantics. Since this syntagmatic level of representation represents the most cross-linguistically variable component of a grammar (cf. Bateman *et al.*, 1997), the range of congruences that will be found across languages is reduced. Kameyama's style of lattice therefore does little to bring out commonalities useful for grammar development when working multilingually.

Finally, the presentational form of a combined subsumption lattice is clearly

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

problematic for large-scale resources. Even this small example grammar is hardly perspicuous. As Kameyama (p. 202) herself notes:

> "There are two potential problems in an effort to develop a shared grammar as described here. One is the need for serious cooperation among the developers. A small change in shared templates can always affect language-specific templates that someone else is working on. The other problem is the sheer complexity of the inheritance lattice."

We believe KPML to have largely solved both problems by avoiding formal inheritance across languages and by providing task-oriented, focused access to the linguistic resources as illustrated in the previous section. The components of our multilingual resources responsible for nominal groups are very much larger than the fragment reported in Kameyama (consisting of the CLASSIFICATION, COUNTNUMBER, DEICTIC, DETERMINATION, NG-COMPLEXITY, NOUNTYPE, ORDINALITY, POST-DEICTICITY, PROC-THINGTYPE, PRONOUN, QUALIFICATION, QUANTIFICATION and SELECTION regions of Table 1) and were developed by different grammar writers at different times and locations.

More recently, Cahill and Gazdar (1995) have proposed a further development building on the direction started by Kameyama and employing the powerful representational facilities of DATR (Evans and Gazdar, 1996). Cahill and Gazdar examine the problem of capturing similarities and differences in 'related languages'. They show that such similarities and differences exist across various distinct linguistic strata and that a multilingual lexicon should capture this. To do this they provide a single set of DATR statements that combine to define a set of hierarchically related lexicons. For a multilingual lexicon of n languages, $n + 1$ individual lexicons are created where each lexicon maintains the linguistic structure typical of a full lexicon (in the currently popular all-inclusive sense, i.e. including phonological, morphological, syntactic and semantic information). The extra lexicon additional to the n monolingual lexicons is a 'common lexicon' that maintains the default information that is held in common across the monolingual lexicons. Inheritance is permitted from the default common lexicon to individual monolingual lexicons, but not between monolingual lexicons.

A concrete example from the number systems of English, Dutch and German is given by Cahill and Gazdar (1996); we extract two brief and simplified examples from their description in order to clarify the approach. The general scheme is that the DATR statements of the multilingual lexicon serve to 'transduce' an input specification (consisting of a sequence of specifiers for the numbers of thousands, hundreds, tens and units) into an output phonological specification. One difference between English on the one hand and German and Dutch on the other (captured by Cahill and Gazdar in their morphological description) is that whereas English would require "one hundred", German and Dutch are content with simply "Hundert"/"Honderd". This is captured by specifying in the multilingual, common component of the lexicon DATR rules that state that an input specification of a single hundred (`<c 1>`) can be rendered as the phonological realization for 100 (called `Phon100`), whereas other quantities of hundreds require a specifier (re-expressed as units: `i`):

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

```
Morphology:
 <c>  == <i> "Phon100:<L phn root>"
 <c 1> == "Phon100:<L phn root>".
```

However, for the English-specific morphological component, the latter general rule needs to be overriden by the more specific:

```
Morphology/E:
 <c 1> == <i 1> Morphology.
```

The second example is drawn from Cahill and Gazdar's phonological description. Here they present a common, shared phonological form, for example of the phonological item `Phon006`, as consisting of the sequence [z, E, k, s]. Each element of the sequence is named, i.e. onset, peak, body, and tail respectively. Phonological representations for individual languages can then be 'produced' by overriding these defaults: Dutch, for example, selects a null body ([z, E, , s]), whereas English selects an onset of 's' and a peak of 'I' ([s, I, k, s]).

Although this approach is in some ways similar to that of multilingual system networks, the main differences are in the reliance on an explicit common ground for inheritance and in the notion of 'related language'. Presumably because of experiences in a structural linguistic tradition, the possibility of significant resource sharing is only investigated for languages classified as similar according to traditional structural typology. Under this assumption, the adoption of a single common lexicon is understandable. From the communicative-functional perspective represented in multilingual system networks, however, there is no expectation that similarities will be restricted to lines of structural typology. This was one message behind our choice of three typologically unrelated languages in the example of Section 2.3. It is, furthermore, unlikely that for arbitrary sets of languages, a single common resource set will serve as the best model; this is the situation that leads to extreme 'atomization' in Kameyama's account. There are few grounds for a model that only includes one 'common' resource set from which the individual monolingual accounts are seen as deviations or instantiations.

In short, we need to move away from the hardcoded common-noncommon relationship to a flexibility to move freely along the dimension of language variety to support more or less mono/multilingual views. In the multilingual system network approach, it is straightforward to represent the fact that even within traditional sets of 'typologically-related' languages (e.g. English, German, French, Dutch), some languages will be more closely related than others. Moreover, the closeness of the relationship varies depending on what part of the linguistic system one is examining (cf. Table 1). It is therefore unnatural to require that there be one 'common' default set of definitions. Our approach rejects this requirement and presents all variants on an equal footing. This avoids the problem noted for interlingua in Machine Translation (cf., for example, Hutchins and Somers, 1992, p. 120) of having to select a 'neutral' form: the selection of 'zeks' above as the neutral form might be argued diachronically, but this is not an ideal situation when engineering large-scale resources for languages that might not be so closely or clearly related. Finally, note

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

that if a multilingual system network only contains partitions that are relevant to either all languages or to one language, and never to other language combinations, then such a network collapses to the kind of multilingual lexicon structure proposed by Cahill and Gazdar. We have, however, found no such resources in our own multilingual work.

It is probably straightforward to extend the DATR implementation in order to include not a single common lexicon, but an entire hierarchy of relatively more or less common lexicons. However, this would also change the interpretation of multilinguality appealed to in those resources. In fact, they would become more like multilingual system networks as described in this paper. This points to an interesting line of development in which a DATR-representation could provide a further useful implementation of the kinds of generation resources developed with KPML. We predict, however, that the level of abstraction offered by the multilingual system network and, indirectly, the kind of views provided by KPML are, as far as a developer is concerned, more suitable for larger-scale multilingual development work. What is important is the level of abstraction presented to the user/developer: the implementational basis may vary considerably as long as the views KPML offers are preserved. This is an important aspect of using the system: the views offered allow a user to manipulate and debug linguistic resources without needing to consider the representation of those resources, whatever that might be. This naturally raises the question as to what extent the level of abstraction provided by KPML can be usefully applied to other approaches: this will no doubt form an area of future inquiry, particularly given the translatability of systemic-style grammars into alternative representational schemes mentioned below. To the extent that an approach supports the kind of abstractions upon which KPML's resource development/maintenance methodologies rely, then useful transfer can be expected.

4.2 Resource development time

Although it is clear that more extensive evaluations and comparisons of development times with and without KPML need to be undertaken, preliminary results obtained for the task of adding generation capabilities in a new target language are very encouraging. *Regardless* of language-family distance it appears that 1–3 person-months is an adequate time for a non-expert to bring up a working generation component covering simple applications and capable of mapping extended SPL-style semantic specifications to surface strings or for producing a substantial core grammar that can serve as a basis for extensive development. This time includes training in the use of KPML. The linguistic components developed naturally inherit the normal features of KPML resources, including support of hypertext generation and the provision of generation servers. Documented work includes the development by transfer comparison of Dutch resources by Degand (1996) and of Japanese resources by Hauser (1995). Similar results were obtained in the construction of French resources within the Drafter project Hartley and Paris (1997b).

These experiences support those of the CLE group where, building on the earlier experiments mentioned above for developing Swedish grammars from English, there

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

have now been developments for French and Spanish (cf. Rayner *et al.*, 1996) and for producing Danish resources from Swedish (Rayner *et al.*, 1997). Here there has been a marked 'learning-curve' improving the speed with which new resources can be developed. Although precise comparisons are difficult due to the different natures of the grammars involved and the different purposes for which they are developed (the CLE grammars are intended to provide analysis and generation for single sentences in dialogues; the grammars discussed here concentrate more on generation of extended texts on the basis of specifications produced by state of the art text planners), it appears that the general multilingual resource organization motivated by Bateman *et al.* (1997) and the corresponding development tools provided by KPML cover a wider range of grammatical phenomena in a shorter time and are not restricted to structurally similar languages such as English/Swedish, or French/Spanish. A more detailed comparison of the two approaches leading to a possible combination of their respective strengths is, however, clearly called for.

Establishing the complete development time for a generation capability must also take into consideration the size of the lexicon that an application requires. The Penman-style approach to lexicalization that KPML adopts is deliberately simple: the main body of work during generation is done by the detailed *grammar* rather than a lexicon. Lexicon entries contain little more than morphological information and by and large simply plug into appropriate places in structure. Input specifications can either call for such lexemes directly, or allow associations to be specified between semantic entities and lexeme sets. Lexicon entries may also be created dynamically as required on the basis of spellings presented in the input specifications; such entries take on default morphological properties. This strategy can be used for interfacing straightforwardly between semantic specifications for generation and application-specific thesauri that may be available: for example, the texts generated in one of our test scenarios (the Dictionary of Art experiment: Alexa *et al.*, 1996) freely avail themselves of the many thousands of terms defined in the Getty *Arts and Architecture Thesaurus* (AAT).

While most existing proposals for lexicon architectures are not specifically beneficial for generation, it will also generally be worthwhile interfacing generation resources with such linguistic databases as they become available. In the terms of recent EAGLES working group recommendations, for use in generation it is necessary to provide a 'structuring layer' whereby lexical information may be accessed during the generation process, i.e. a bridge between generic lexicon resources and practical generation-capable resources, such as system networks, could be beneficially defined. Current work in progress includes establishing relationships between the standard multilingual semantic model adopted by KPML resources – the Generalized Upper Model (Bateman *et al.*, 1995) – and the Princeton WordNet.[6]; this will be extended for EuroWordNet (EU LRE project) as fragments become available. Interfaces with other lexicons as they become available are also obviously desirable.

[6] This is analogous to the 'Middle Model' and 'Sensus' experiments reported by Knight and Luk 1994.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

4.3 Resource sharing

The expectation that there are indeed substantial gains to be made by sharing congruent linguistic descriptions has also been borne out. An overview of the current state of development of KPML resources for Greek, English, German, Dutch, French, and Japanese is shown in Table 1.[7]

The table is organized according to the grammatical *regions* that make up a grammar, i.e. a portion of the grammar responsible for the realization of some particular area of semantics. Each row shows the region concerned, the number of grammatical systems in that region for each language, the total for these individual languages (Σ), the total number of systems in the *multilingual* representation of the region (*ml*-Σ), and the percentage of the individual total that the multilingual total represents. For example, the region CULMINATION has six grammatical systems in the Greek grammar, six grammatical systems in the English grammar, none in the German grammar, seven in the Dutch grammar, six in the French grammar, and six in the Japanese grammar. Individually, that is, if these grammars were represented separately, that would be a total of 31 grammatical systems that need representation. However, in the multilingual representation, there are only seven systems and these are re-used for most of the language varieties considered. This represents 22.58% of the individual total. The final totals show that for all six languages the multilingual representation contains only 32.4% of the number of objects (grammatical systems) required if the resources were to be maintained individually.

Sharing on a broader scale is shown by the very few regions that are applicable only to a single language – and these are generally due to particular quirks of development rather than theoretical differences. This indicates that languages are more or less congruent in their general resource organization.

These figures represent a snapshot of the current development, not a claim concerning the final or 'absolute' shareability of resources across the languages considered. They are therefore to be read as indications of a line of development rather than as definitive statements. Some regions have not been merged at all since they have been developed separately without reference to multilingual aspects (e.g. the TRANSITIVITY region of German which could be beneficially reconciled with the relational and nonrelational transitivity regions – REL-TRANS and NONREL-TRANS – of the other languages); some regions have been inherited by languages automatically by our development tools even though they should probably be weeded out or

[7] Some notes on this table are in order. First, we can obtain a broad degree of comparison between the size of these grammars and other grammars making use of type lattices by adopting the type-lattice encoding of systemic grammars developed by Henschel 1995. Grammatical systems introduce 'on average' two grammatical features each and these features can be coded as types. Each grammar therefore involves around 1200 types in a definitional language such as that for CUF or TDL; due to the high degree of multiple inheritance that systemic grammars typically employ, however, this figure expands to several hundreds of thousands (at least) for encodings such as that of ALE. Second, as typical for generation, the grammar is organized around *functional semantic* regions rather than around syntactic constructions. A more detailed comparison would need to locate the syntactic constructions usually appealed to in analysis-oriented work. However, as the type figures indicate, these grammars are, by any current standards, 'large grammars'.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Table 1. *Grammatical resources statistics by functional regions (extract)*

REGION	*Gre*	*Eng*	*Ger*	*Dut*	*Fre*	*Jap*	Σ	*ml-Σ*	*ml-size*
ADJ-COMP	27	27	8	37	26	26	151	40	26.49%
ADJ-GROUP	1	1	1	1	1	2	7	2	28.57%
ADVERBIAL	8	8	6	10	8	8	48	10	20.83%
ADVERBIALS	0	0	2	0	0	0	2	2	100.00%
ATTITUDE	1	1	0	1	1	1	5	1	20.00%
CIRCUMSTANTIAL	28	26	16	26	26	29	151	39	25.83%
CLASSIFICATION	4	4	4	4	4	4	24	4	16.67%
CLAUSECOMPLEX	34	34	38	38	34	39	217	50	23.04%
CONJUNCTION	21	21	21	21	21	21	126	21	16.67%
COUNTNUMBER	6	6	6	6	6	6	36	7	19.44%
CULMINATION	6	6	0	7	6	6	31	7	22.58%
DEICTIC	5	0	4	0	1	0	10	7	70.00%
DEPENDENCY	31	31	4	33	31	30	160	37	23.12%
DETERMINATION	32	36	47	75	47	35	272	114	41.91%
DIATHESIS	0	0	6	0	0	0	6	6	100.00%
DIATHESIS-GATES	0	0	66	0	0	0	66	66	100.00%
ELABORATION	10	10	0	10	10	10	50	10	20.00%
ELLIPSIS	5	5	4	6	5	5	30	9	30.00%
EPITHET	18	5	18	6	5	5	57	20	35.09%
MODALITY-FORMS	0	0	0	0	0	9	9	9	100.00%
MOOD	44	44	7	42	44	48	229	60	26.20%
NG-COMPLEXITY	12	13	18	13	14	21	91	30	32.97%
NONREL-TRANS	56	55	1	54	55	55	276	56	20.29%
NOUNTYPE	20	16	30	17	16	17	116	32	27.59%
ORDINALITY	3	3	3	3	3	3	18	3	16.67%
PHRASAL-MOOD	1	1	0	1	1	1	5	1	20.00%
POLARITY	8	8	1	5	8	8	38	8	21.05%
POST-DEICTICITY	8	7	0	7	7	7	36	9	25.00%
PPOTHER	26	26	44	25	26	26	173	63	36.42%
PPSPATIOTEMPORAL	30	32	33	33	33	36	197	54	27.41%
PROC-THINGTYPE	7	7	5	7	7	7	40	11	27.50%
PRONOUN	60	34	70	36	33	35	268	128	47.76%
QUALIFICATION	10	11	17	11	11	18	78	25	32.05%
QUANTIFICATION	18	18	19	18	18	18	109	19	17.43%
QUANTITY-GROUP	26	26	16	26	26	26	146	26	17.81%
RANKING	8	8	5	7	8	10	46	13	28.26%
REL-TRANS	42	42	0	46	42	47	219	51	23.29%
SELECTION	9	8	5	9	9	9	49	9	18.37%
TAG	0	21	0	21	21	21	84	21	25.00%
TENSE	14	15	12	14	15	17	87	21	24.14%
THEME	24	24	45	27	24	24	168	73	43.45%
TRANSITIVITY	0	0	39	0	0	0	39	39	100.00%
VOICE	17	17	1	18	17	17	87	19	21.84%
WORD-FORMS	29	29	0	0	29	52	139	80	57.55%
⋮			⋮				⋮	⋮	⋮
Totals	747	733	679	823	746	809	4537	1470	32.40%

considerably reduced at a subsequent stage in development (e.g. the TAG regions of Dutch, Japanese and French); and some regions have simply had more attention in one language than in others (e.g. the THEME region in German). Nevertheless, since there will be both further increases in congruent, and hence shareable, descriptions as well as further divisions, it appears clear that a considerable saving in linguistic

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

description is possible within the multilingual framework adopted, as well as the support of a wide range of interesting contrastive statements.

Kameyama describes the goal of multilingual description as follows: "It should reduce the size of a multilingual rule base, and facilitate the addition of new languages"(1988, p. 194). This has certainly been achieved with KPML.

4.4 Subgrammars

The favoured methodology for developing resources in KPML for use in applications has come to contrast interestingly with several previous application-oriented approaches. For example, the approaches to multilinguality adopted in the Meaning-Text-Model based systems FOG, LFS, etc. (cf. Bourbeau *et al*., 1990; Polguère, 1990; and Iordanskaja *et al*., 1992), adopt from the outset the target of generating for a specific sublanguage. This has meant that new applications have required the creation of new generation components. Although it is sometimes suggested (e.g. Busemann, 1996) that it is quicker to develop simple application-specific resources than to adapt broader coverage, general grammatical resources to the needs of particular applications, such specialized resource sets become increasingly unmanageable as the needs of an application grow. The extensibility and re-usability of the specialized resources are therefore limited.

KPML favours instead working from general resources – adding to these and then pruning as required for a particular application. This ensures that extensibility is built in. With sufficiently broad coverage of *realistic* natural language phenomena, i.e. the phenomena that actually occur in texts, this results in little overhead in development time when providing a generation capability. Most overheads arise when the application demands and the existing linguistic resources are ill-matched: and this can occur most readily when the original motivations for linguistic resource development were not text-driven.[8] Even in the extreme case that, due for example to time pressures, linguistic resources are developed for some language that only function for a particular application scenario, KPML continues to provide full support for extending and maintaining these resources in the face of subsequent changes or extensions in that, or another, application's requirements. There is then no qualitative difference between the (temporarily) restricted linguistic resources and broad-coverage resources.

A further current line of development is therefore the automation of the process of extracting specific application resource sets from a general resource set given a characterization of the target requirements (as provided, most straightforwardly, by a target example set). This can usefully be seen as another of the customization steps often necessary when moving from general resources to resources for particular applications. Some initial results concerning tools for automatic subgrammar extraction from KPML-style resources are given in Henschel and Bateman (1997).

[8] An approach now also accepted by analysis-oriented projects such as LS-GRAM (cf. EAGLES p. 106).

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

4.5 Implementation basis

KPML may be usefully considered as consisting of two main modules: the core generation and maintenance facilities, and the graphical user interface. It is straightforward to configure the system for use without the user-interface to provide black-box tactical generation; this functionality is also available in server mode, i.e. to provide a generic server that accepts semantic specifications and produces augmented string structures for its clients.

Currently, the generation core of KPML is built on top of the Penman text generation system (Mann and Matthiessen, 1985; Penman Project, 1989).[9] The resource specification language and its interpreter for Penman-style systemic-functional linguistic descriptions has been extended to permit straightforward conditionalization according to language (Zeng, 1992). The multilingual conditionalization inserts a further layer of operations between the original processes for generation and the resource definitions: the traversal algorithm (cf. Matthiessen and Bateman, 1991, p. 100) is unchanged. The notational extension is fully general and applies to all types of linguistic entities supported in the Penman-style approach, i.e. grammatical systems, constraints on structure (realization statements), choosers, inquiries, lexical entries, punctuation rules, logical forms, and examples in test suites. This generation core is written in ANSI standard Common Lisp; the main implementation configuration supported is Allegro Common Lisp (ACL 4.3), running primarily on Unix workstations.[10]

Generation is relatively fast. Short paragraphs can be produced from semantic specifications in a few seconds; generation is also considerably faster than spoken language and so is appropriate if speech output is to be produced. Nevertheless, it is clear that further improvements are desirable. Possible future directions here include the reimplementation of the basic generation engine streamlined so as to remove support for the extensive tracing and debugging information maintained in KPML. The simplicity of the traversal algorithm currently employed renders such a direct re-implementation straightforward. An alternative strategy would be the adoption of an intermediate re-implementation in a formalism that has itself already been reworked for faster generation, e.g. the suggested C implementation of Elhadad and Robin's (1996) Functional Unification Formalism.

This latter strategy is well supported by our maintenance of a formally and computationally specifiable relationship to state of the art computational formalisms (such as typed feature formalisms) at all points of our linguistic representation. This connection is built on ongoing work where automated mappings from system networks to a variety of typed feature formalisms (e.g. ALE, CUF, TDL) have been established (cf. Henschel, 1995). While typed feature systems will not provide the performance required for practical text generation in the near future, the formalisms involved will continue to represent the standard in computational linguistics. The

[9] Although numerous code changes have also been made to the underlying Penman system. It is not, for example, advisable to simply load KPML 'on top of' a Penman installation.

[10] Core generation has also been successfully tested under Linux and under Windows on a PC-version ported by Mick O'Donnell.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

automatic migration of linguistic information (particularly at this time lexicons) from formally expressive but computationally too expensive representations to restricted notations of demonstrated value for practical generation (such as system networks) is therefore a continuing concern.

The graphical user interface of KPML is provided as an additional layer to the system. It is currently implemented using the Allegro version of the Common Lisp Interface Manager (CLIM 2.1). As well as the graphical displays for linguistic resources supported here, further extensions include language-specific font selection for the information presented, seamless interaction with GNU Emacs (and Mule for use of further writing systems such as Japanese, Chinese, etc.), and graphical editing of system networks. Other implementations of the most useful functionalities for resource development and maintenance are being investigated to achieve improved responsiveness and wider portability. Add-ons for Emacs and Web-based browsers are being considered.

5 Conclusions

It is now accepted that large-scale linguistic resources should be established that maximise their re-usability across different applications. Unfortunately there has been insufficient support for resources of this kind that are appropriate for natural language generation. In this paper we have described KPML, a system that provides for the development of such resources multilingually. KPML has aimed:

- to offer generation projects large-scale, general linguistic resources which are well tested and verified in their coverage, possess standardized input and output specifications, and are appropriate for practical generation;
- to offer generation projects a basic engine for using such resources for generation;
- to encourage the development of similarly structured resources for languages where they do not already exist,
- to provide optimal user-support for undertaking such development and refining general resources to specific needs;
- to minimise the overhead (and cost) of providing texts in multiple languages;
- to encourage contrastive functional linguistic work; and
- to raise awareness and acceptance of text generation as a useful endeavor.

KPML enables detailed investigations of the feasibility of achieving sharing and re-use of linguistic information across different languages and has already provided basic generation and resource development technology in a number of large generation projects; these projects include to date GIST (Not and Stock, 1994), Drafter (Paris and Vander Linden, 1996), KOMET (Bateman and Teich, 1995), TECHDOC Rösner and Stede, 1991, and HealthDoc (DiMarco *et al*., 1995).

A common position taken in multilingual generation is that multilinguality can best be positioned externally to the language generation components and that individual languages are then the responsibility of distinct modules (e.g. Rösner, 1992, p. 307). Here we have argued that our view of multilinguality significantly

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

increases the ease of development and maintenance of language resources while nevertheless preserving a high degree of resource 'sharing' within the languages covered. We have also outlined a supported methodology for constructing generation capabilities for new languages and/or tasks and its results and have given some concrete examples of the degree of resource sharing that KPML thereby provides.

The language resources currently provided as compatible with KPML were indicated in Table 1. These resources are being developed further as necessary and illustrate a variety of resource construction methodologies. Whereas English and German resources have been developed largely monolingually, Dutch and Japanese were developed by explicit transfer comparison, i.e. by the 4-stage methodology given in Section 2.6 above); the Dutch resources are now well into step (4) of the methodology where particular regions are being analysed in close detail purely language-internally. Greek and French are being developed according to a hybrid methodology which we can label casual transfer comparison: here resources are motivated language-internally, but with a view to treatments already present in other resources in order to avoid duplication of effort. Further languages are in preparation and a resource-developers user-group is being set up.

Further information about the current status and availability of KPML and KPML-compatible resources can be obtained from the KPML WWW home page at '`http://www.darmstadt.gmd.de/publish/komet/kpml.html`'.

Acknowledgements

The KPML development environment (β-versions 0.1 to 0.9 and release 1.0) was developed while the author was at the German National Research Center for Information Technology's institute for Integrated Publication and Information Systems (IPSI) in Darmstadt, Germany. Particular thanks are due to Licheng Zeng (University of Sydney) for the initial implementations of the multilingual extensions to Penman adopted in KPML and to Christian Matthiessen (Macquarie University). Additional thanks for various aspects of the development of KPML and its documentation are due to Markus Fischer, Cécile Paris, Keith Vander Linden, Tony Hartley, John Wilkinson, Brigitte Grote, Fabio Rinaldi, Elke Teich, Melina Alexa, Renate Henschel, Liesbeth Degand, and Bernhard Hauser. The final version of this paper was also considerably improved by the helpful comments of three anonymous reviewers. Finally, the financial support of DAAD/British Council cooperation project (DAAD/ARC-313) and the former German Ministry for Research and Technology (BMFT: Project 'INTEGRA') is gratefully acknowledged. The development of the Penman system on which KPML builds was supported by U.S. National Science Foundation Grant IST-8408726, and U.S. Federal Contract numbers MDA903-81-C-0335, MDA903-87-C-0641, F49620-84-C-0100, and F49620-87-C-0005.

References

Alexa, M., Bateman, J. A., Henschel, R. and Teich, E. (1996) Knowledge-based production of synthetic multimodal documents. *ERCIM News* (26): 18–20, July.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Alshawi, H., Carter, D. and Rayner, M. (1991) Translation by quasi-logical form. *Proceedings of the 29th. Annual Meeting of the Association for Computational Linguistics* Berkeley, CA.

Alshawi, H., Carter, D., Gambäck, B. and Rayner, M. (1992) Swedish-English QLF translation. In Hiyan Alshawi, editor, *The Core Language Engine*, pp. 277–319. MIT Press.

Bateman, J. A. and Teich, E. (1995) Selective information presentation in an integrated publication system: an application of genre-driven text generation. *Information Processing and Management: Special Issue on Summarizing Text*, **31**(5): 753–768, September.

Bateman, J. A., Matthiessen, C. M. I. M., Nanri, K. and Zeng, L. (1991) The re-use of linguistic resources across languages in multilingual generation components. *Proceedings of the 1991 International Joint Conference on Artificial Intelligence, Sydney, Australia*, volume 2, pp. 966–971. Morgan Kaufmann.

Bateman, J. A., Henschel, R. and Rinaldi, F. (1995) Generalized upper model 2.0: documentation. *Technical report*, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany.

Bateman, J. A., Matthiessen, C. M. I. M. and Zeng, L. (1997) A general architecture of multilingual resources for natural language processing. *Technical report*, University of Stirling, Stirling and Macquarie University, Sydney.

Bateman, J. A. (1992a) Grammar, systemic. In: S. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Second Edition*, pp. 583–592. John Wiley & Sons.

Bateman, J. A. (1992b) The theoretical status of ontologies in natural language processing. In: S. Preuß and B. Schmitz, editors, *Text Representation and Domain Modelling – ideas from linguistics and AI*, pp. 50–99. (KIT-Report 97, Technische Universität Berlin, May.) (Papers from KIT-FAST Workshop, Technical University Berlin, October 9th–11th 1991). Also available from the Computation and Language E-print archive: cmp-lg/9704010.

Bateman, J. A. (1995) Rezension von: Dik, Simon C.: Functional Grammar in Prolog: An integrated implementation for English, French, and Dutch: Berlin, New York: Mouton de Gruyter, 1992. *Zeitschrift für Sprachwissenschaft* **14**(1): 91–100. (Review in English.)

Bateman, J. A. (1997a) Some apparently disjoint aims and requirements for grammar development environments: the case of natural language generation. *Proceedings of ACL/EACL97 Workshop: "ENVGRAM: Computational Environments for grammar development and linguistic engineering"*. Association for Computational Linguistics.

Bateman, J. A. (1997b) *KPML Development Environment: multilingual linguistic resource development and sentence generation.* (Release 1.1). GMD-Studie Number 304. German National Center for Information Technology (GMD), Sankt Augustin, Germany.

Bateman, J. A. (To appear) Automatic discourse generation. In: A. Kent, editor, *Encyclopedia of Library and Information Science*. New York: Marcel Dekker.

Bourbeau, L., Carcagno, D., Goldberg, E., Kittredge, R. and Polguère, A. (1990) Bilingual generation of weather forecasts in an operations environment. In: H. Kargren, editor, *Proceedings of the 13th. International Conference on Computational Linguistics*, pp. 318–320. Helsinki, Finland.

Busemann, S. (1996) Best-first surface realization. *Proceedings of the 8th. International Workshop on Natural Language Generation (INLG '96)*, pp. 101–110, Herstmonceux, UK.

Cahill, L. J. and Gazdar, G. (1995) Multilingual lexicons for related languages. *Proceedings of the 2nd DTI Language Engineering Conference*, pp. 169–176, London. Department of Trade and Industry.

Cahill, L. J. and Gazdar, G. (1996) A lexical analysis of numerical expressions in three related languages. *Proceedings of the AISB workshop on multilinguality in the lexicon.* AISB.

Coch, J., David, R. and Magnoler, J. (1995) Quality test for a mail generation system. *Proceedings of Linguistic Engineering '95*, Montpellier, France.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S. and Sag, I. (1995) Translation using Minimal Recursion Semantics. *Proceedings of the 6th. International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, Leuven, Belgium. July.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Copestake, A. (1994) Constraints, tlinks and MT. *Acquilex-II Working Paper 16*, ESPRIT BRA-7315 ACQUILEX-II.

Degand, L. (1996) A Dutch component for a multilingual systemic text generation system. In: G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: an artificial intelligence perspective: Lecture Notes in Artificial Intelligence 1036*, pp. 350–367. Berlin: Springer-Verlag. (Selected Papers from the *4th. European Workshop on Natural Language Generation*, Pisa, Italy, 28–30 April 1993.)

Dik, S. C. (1992) *Functional Grammar in Prolog: an integrated implementation for English, French, and Dutch.* Berlin: Mouton de Gruyter.

DiMarco, C., Hirst, G., Wanner, L. and Wilkinson, J. (1995) Healthdoc: Customizing patient information and health education by medical condition and personal characteristics. In: A. Cawsey, editor, *Proceedings of the Workshop on Patient Education.* Glasgow.

EAGLES. (1996) Formalisms working group final report. *Expert advisory group on language engineering standards document*. September.

Elhadad, M. and Robin, J. (1996) A reusable comprehensive syntactic realization component. In: *Demonstrations and Posters of the 1996 International Workshop on Natural Language Generation (INLG '96)*, pp. 1–4. Herstmonceux, UK.

Evans, R. and Gazdar, G. (1996) DATR: A language for lexical knowledge representation. *Computational Linguistics* **22**.

Glass, J., Polifroni, J. and Seneff, S. (1994) Multilingual language generation across multiple domains. *Proceedings of the 1994 International Conference on Spoken Language Processing*, Yokohama, Japan.

Hajičová, E. and Kirschner, Z. (1987) Fail-soft ("emergency") measures in a production-oriented MT system. *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen. Association for Computational Linguistics.

Halliday, M. A. K. (1966) Some notes on 'deep' grammar. *Journal of Linguistics* **2**(1): 57–67.

Halliday, M. A. K. (1978) *Language as social semiotic.* London: Edward Arnold.

Halliday, M. A. K. (1985) *An Introduction to Functional Grammar.* London: Edward Arnold.

Hartley, A. and Paris, C. L. (1997a) Automatic text generation for software development and use. In: H. Somers, editor, *Terminology, translation and LSP: studies in language engineering in honor of J.C. Sager*, pp. 221–242. Amsterdam: Benjamins.

Hartley, A. and Paris, C. L. (1997b) Multilingual document production: from support for translating to support for authoring. *Machine Translation* **12**(1–2): 109–129.

Hauser, B. (1995) Multilinguale Textgenerierung am Beispiel des Japanischen. (Diplomarbeit).

Henschel, R. and Bateman, J. A. (1997) Application-driven automatic subgrammar extraction. *Proceedings of ACL/EACL97 Workshop: "ENVGRAM: Computational Environments for grammar development and linguistic engineering"*. Association for Computational Linguistics.

Henschel, R. (1995) Traversing the Labyrinth of Feature Logics for a Declarative Implementation of Large Scale Systemic Grammars. In: S. Manandhar, editor, *Proceedings of the CLNLP 95.*

Hobbs, J. R. and Kameyama, M. (1990) Translation by abduction. *13th. International Conference on Computational Linguistics (COLING-90)*, volume III, pp. 155–161. Helsinki, Finland.

Hutchins, W. J. and Somers, H. L. (1992) *An introduction to Machine Translation.* London: Academic Press.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B. and Polguère, A. (1992) Generation of extended bilingual statistical reports. *Proceedings of COLING-92*, pp. 1019–1023.

Jacob, D. and Maier, E. (1988) Die Übertragung des MUMBLE-Generators für die Generierung von Deutsch. In: H. Trost, editor, *4. Österreichische Artificial-Intelligence-Tagung Proceedings*, number 176 in Informatik-Fachberichte. Berlin: Springer-Verlag.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Kameyama, M., Ochitani, R. and Peters, S. (1991) Resolving translation mismatches with information flow. *Annual Meeting of the Association of Computational Linguistics*, pp. 193–200, Berkeley, CA. Association of Computational Linguistics.

Kameyama, M. (1988) Atomization in grammar sharing. *Proceedings of the 26th. Annual Meeting of the Association for Computational Linguistics*, pp. 194–203. Association for Computational Linguistics.

Kasper, R. T. (1989) A flexible interface for linking applications to PENMAN's sentence generator. *Proceedings of the DARPA Workshop on Speech and Natural Language*. (Available from USC/Information Sciences Institute, Marina del Rey, CA.)

Kay, M., Gawron, J. M. and Norvig, P. (1994) *Verbmobil: a translation system for face-to-face dialog*. CSLI, Stanford, CA.

Kay, M. (1996) Multilinguality: overview. In: R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen and V. Zue, editors, *Survey of State of the Art in Human Language Technology*, chapter 8. Kluwer Academic.

Kittredge, R., Iordanskaja, L. and Polguère, A. (1988) Multilingual Text Generation and the Meaning-Text Theory. *Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Carnegie Mellon University, Pittsburgh, PA.

Kittredge, R. (1992) Guest editor's note: On the relevance of text generation research in machine translation. *Machine Translation* **7**(4): 1–4.

Kittredge, R., editor (1995) *Proceedings of the IJCAI 95 workshop on Multilingual Text Generation*. Montréal, Québec: AAAI.

Knight, K. and Luk, S. K. (1994) Building a large-scale knowledge base for machine translation. *Proceedings of AAAI-94*, Seattle, WA. American Association of Artificial Intelligence.

Lee, J.-H., Okumura, A., Muraki, K. and Kim, G.-C. (1991) An English-Korean machine translation system: korean synthesis under the environment of Japanese generation system. *Proceedings of 1991 Japan-Australia Joint Symposium on Natural Language Processing*, pp. 219–224. Fukuoka, Japan.

Mann, W. C. and Matthiessen, C. M. I. M. (1985) Demonstration of the Nigel text generation computer program. In: J. D. Benson and W. S. Greaves, editors, *Systemic Perspectives on Discourse, Volume 1*, pp. 50–83. Norwood, NJ: Ablex.

Mann, W. C. (1983) An overview of the PENMAN text generation system. *Proceedings of the National Conference on Artificial Intelligence*, pp. 261–265. AAAI, August. (Also appears as USC/Information Sciences Institute, RR-83-114.)

Matthiessen, C. M. I. M. and Bateman, J. A. (1991) *Text generation and systemic-functional linguistics: experiences from English and Japanese*. London: Frances Pinter/New York: St. Martin's Press.

Matthiessen, C. M. I. M., Nanri, K. and Zeng, L. (1991) Multilingual resources in text generation: ideational focus. *Proceedings of the 2nd Japan-Australia Joint Symposium on Natural Language Processing*, Kyushu, Japan.

McDonald, D. D. (1983) Description directed control: its implications for natural language generation. *Computers and Mathematics* **9**(1): 111–129. (Reprinted in B. J. Grosz *et al.*, editors, *Readings in Natural Language Processing*. Morgan Kaufman. 1986, pp. 519–538.)

Meteer, M. W. (1992) *Expressibility and the Problem of Efficient Text Planning*. London: Pinter.

Nagao, M., Tsujii, J. and Nakamura, J. (1988) The Japanese government project for machine translation. In: J. Slocum, editor, *Machine Translation Systems*, pp. 141–186. Cambridge: Cambridge University Press.

Not, E. and Stock, O. (1994) Automatic generation of instructions for citizens in a multilingual community. *Proceedings of the European Language Engineering Convention*, Paris, France.

Okumura, A., Muraki, K. and Akamine, S. (1991) Multi-lingual sentence generation from the PIVOT interlingua. *Proceedings of the MT Summit '91*, pp. 67–71.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press

Paris, C. L. and Vander Linden, K. (1996) DRAFTER: an interactive support tool for writing multilingual instructions. *IEEE Computer*.

Penman Project. (1989) PENMAN documentation: the Primer, the User Guide, the Reference Manual, and the Nigel manual. *Technical report*, USC/Information Sciences Institute, Marina del Rey, California.

Quantz, J., Gehrke, M., Küssner, U. and Schmitz, B. (1994) The VERBMOBIL domain model version 1.0. *Verbmobil Report 29*, University of the Saarland, Saarbrücken, Germany.

Rayner, M., Carter, D. and Bouillon, P. (1996) Adapting the core language engine to French and Spanish. *Proceedings of NLP-IA-96*, Moncton, New Brunswick.

Rayner, M., Carter, D., Bretan, I., Eklund, R., Wirén, M., Hansen, S. L., Kirschmeier-Andersen, S., Philp, C., Sorensen, F. and Thomsen, H. E. (1997) Recycling lingware in a multilingual MT system. *Proceedings of ACL/EACL97 Workshop: "From research to commercial applications: making NLP technology work in practice"*. Association for Computational Linguistics.

Reiter, E., Mellish, C. and Levine, J. (1995) Automatic generation of technical documentation. *Applied Artificial Intelligence* **9**.

Rösner, D. and Stede, M. (1991) Customizing RST for the automatic production of technical manuals. *Technical Report FAW-TR-91028*, Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW).

Rösner, D. and Stede, M. (1994) Generating multilingual documents from a knowledge base: the TECHDOC project. *Proceedings of the 15th. International Conference on Computational Linguistics (*COLING 94*)*, volume I, pp. 339–346. Kyoto, Japan.

Rösner, D. (1992) Remarks on multilinguality and generation. In: R. Dale, E. Hovy, D. Rösner and O. Stock, editors, *Aspects of automated natural language generation: Lecture Notes in Artificial Intelligence 587*, pp. 306–308. Berlin: Springer-Verlag.

Schütz, J. (1996) Intelligent web-based information services. In: C. D. Spyropoulos, editor, *Proceedings of the MULSAIC'96 Workshop at ECAI'96*. Budapest, Hungary.

Simpkins, N. K., Cruickshank, G. and P. E International. (1993) ALEP-0 Virtual Machine extensions. *Technical report*, CEC.

Spyropoulos, C. D. and Karkaletsis, E. A. (1996) On-line generation of messages: a knowledge-based approach. In: C. D. Spyropoulos, editor, *Proceedings of the MULSAIC'96 Workshop at ECAI'96*. Budapest, Hungary.

Teich, E., Degand, L. and Bateman, J. A. (1996) Multilingual textuality: Experiences from multilingual text generation. In: G. Adorni and M. Zock, editors, *Trends in Natural Language Generation: an artificial intelligence perspective: Lecture Notes in Artificial Intelligence 1036*, pp. 331–349. Berlin: Springer-Verlag. (Selected Papers from the *4th. European Workshop on Natural Language Generation*, Pisa, Italy, 28–30 April 1993.)

Teich, E. (1995) Towards a methodology for the construction of multilingual resources for multilingual generation. In: R. Kittredge, editor, Proceedings of the IJCAI '95 Workshop on Multilingual Text Generation, pp. 136–148. Montréal, Québec. AAAI.

Whitelock, P. (1992) Shake-and-bake translation. *Proceedings of COLING-92*, volume II, pp. 784–791.

Zeng, L. (1992) ML-Penman: implementation notes. *Technical report*, GMD/IPSI and University of Sydney.

Zeng, L. (1996) *Planning text in an integrated multilingual meaning space: theory and implementation. PhD thesis*, Macquarie University, Sydney, Australia.

https://doi.org/10.1017/S1351324997001514 Published online by Cambridge University Press