
Machine Translation **18:** 81–127, 2003.

Hybrid Natural Language Generation from Lexical Conceptual Structures

NIZAR HABASH and BONNIE DORR
Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA
E-mail: {habash,bonnie}@umiacs.umd.edu

DAVID TRAUM
University of Southern California Institute for Creative Technologies
13274 Fiji Way, Marina del Rey, CA 90292, USA
E-mail: traum@ict.usc.edu

Abstract. This paper describes Lexogen, a system for generating natural-language sentences from Lexical Conceptual Structure, an interlingual representation. The system has been developed as part of a Chinese–English Machine Translation (MT) system; however, it is designed to be used for many other MT language pairs and natural language applications. The contributions of this work include: (1) development of a large-scale Hybrid Natural Language Generation system with language-independent components; (2) enhancements to an interlingual representation and associated algorithm for generation from ambiguous input; (3) development of an efficient reusable language-independent linearization module with a grammar description language that can be used with other systems; (4) improvements to an earlier algorithm for hierarchically mapping thematic roles to surface positions; and (5) development of a diagnostic tool for lexicon coverage and correctness and use of the tool for verification of English, Spanish, and Chinese lexicons. An evaluation of Chinese–English translation quality shows comparable performance with a commercial translation system. The generation system can also be extended to other languages and this is demonstrated and evaluated for Spanish.

Key words: Hybrid Natural Language Generation, Multilingual Natural Language Generation, interlingua, Lexical Conceptual Structure

1. Introduction

This paper describes Lexogen, a system for generating natural-language sentences from Lexical Conceptual Structure (LCS), an interlingual representation (Dorr, 1993b). The system has been developed as part of a Chinese–English Machine Translation (MT) system, called ChinMT; however, it has also been successfully used for many other MT language pairs (e.g., Spanish and Arabic (Dorr et al., 1995)) and other natural-language applications (e.g., cross-language information retrieval (Dorr et al., 2000)).

The work presented here focuses on large-scale sentence-level Hybrid (rule-based/statistical) Natural Language Generation (NLG) in the context of interlingual

MT. Most work on NLG systems has been done within a symbolic framework. Recently, there has been more work on statistical NLG (e.g., IBM travel reports (Ratnaparkhi, 2000)) and hybrid NLG (e.g., Nitrogen (Langkilde and Knight, 1998b) and FERGUS (Alshawi et al., 2000)). There is also a sizeable amount of research in NLG on text planning and generation beyond the sentence level (Hovy, 1988; Vander Linden and Scott, 1995; Olsen, 1997; Marcu et al., 2000).

In the context of MT, the interlingual approach is the most committed to multilinguality, as opposed to transfer MT or direct MT (Polguère, 1991). Interlingual MT systems share the advantage of reuse of knowledge (lexicons or algorithms) at the expense of higher complexity. Examples of interlingual MT include PIVOT (Okumura et al., 1991) and Rosetta (1994). Multilingual NLG systems tend to focus on specific domains with non-linguistic inputs such as database entries or human-created conceptual representations instead of interlinguas. Examples of such systems are Météo (Chandioux, 1989), DRAFTER-I (Paris et al., 1995) and TECHDOC (Rösner and Stede, 1994).

The contributions of this work include:

- Development of a large-scale hybrid NLG system with language-independent components.
- Enhancements to an interlingual representation and associated algorithm (Dorr, 1993b) for generation of ambiguous input.
- Development of an efficient reusable language-independent linearization module with a grammar-description language that can be used with other systems. Additionally, the target-language grammar can be reused by developers of MT systems from new input languages, e.g., those studied in recent TIDES-related efforts (Cebuano, Hindi).
- Improvements to an earlier algorithm (Dorr et al., 1998) for hierarchically mapping thematic roles to surface positions.
- Development of a diagnostic tool for lexicon coverage and correctness and use of the tool for verification of English, Spanish, and Chinese lexicons. An evaluation of Chinese–English translation quality shows comparable performance with a commercial translation system. The generation system can also be extended to other languages and this is demonstrated and evaluated for Spanish.

The next two sections provide overviews of hybrid NLG and LCS-based MT, respectively. The LCS interlingual representation is discussed in more detail in Section 4. Section 5 discusses the Lexogen generation system. Section 6 describes how the Lexogen system can be used to generate output in other languages by adding appropriate target-language resources. A comparison with other sentence generation systems is presented in Section 7. An evaluation of several different aspects of the performance of our system is given in Section 8.

2. Hybrid Natural Language Generation

Most NLG systems follow a symbolic approach that depends on manually creating a system with rules reflecting an understanding of the relationship between input and output. The most basic symbolic approach to NLG, and by far the most common, is template filling. Templates are an excellent choice for very domain-specific interfaces such as ATM machines or junk mail labels; however, they are extremely limited for large-scale domain-independent (or even domain-specific) applications (Reiter, 1995). As for the more realistic systems concerned with larger domains and robust behavior, two major approaches to symbolic NLG are Systemic Grammar and Functional Unification Grammar (FUG) (Jurafsky and Martin, 2000). Within Systemic Grammar – implemented in the paradigm of Systemic Functional Linguistics (Halliday, 1985) – a generation grammar is constructed as a system network of realization statements (grammar rules) that specify mappings from input features to syntactic positions and surface forms. One example of this approach is the Penman generation system (Penman, 1989; Mann and Matthiessen, 1985). FUG uses unification of input representations, called Functional Descriptions, and an FUG as its main process for generation decision making (Elhadad and Robin, 1992; Elhadad et al., 1997). The original functional unification formalism was put forward by Kay (1979).

In contrast to the symbolic approaches, statistical natural language processing is concerned with the creation of computer programs that can perform language-processing tasks by virtue of information gathered from (typically large) corpora. Usually this information is in the form of statistics, but it may be distilled into other forms, such as dictionary entries, or various kinds of rules (Charniak, 2000). In the context of statistical NLG, the object is to create a mathematical model in which the process of mapping meaning representations to fluent natural language is statistically enabled. This approach was used in IBM air travel reports (Ratnaparkhi, 2000), which was the first system to learn mappings from semantics to surface realization automatically by searching for parameters and learning them automatically without using a hand-crafted grammar. What made this possible was a limited domain with simple semantic representations (simple sets of attribute–value pairs) and an available semantically annotated corpus. This system is at best a proof of concept for the use of statistical techniques for semantic-to-surface mapping. It works well for domains with similar complexity to air travel (for which template-based systems already seem adequate). However, the techniques used in it are useless for large-scale domain-independent NLG applications especially with the lack of large-scale semantically annotated corpora.

Hybrid NLG systems combine symbolic and statistical techniques to maximize the advantages and minimize the disadvantages of these different approaches. Large-scale systems need large amounts of lexical, grammatical, and conceptual knowledge. While statistical approaches are quite successful in collecting lexical information such as n-grams, they lack the ability to extract conceptual knowledge

automatically. Moreover, most available knowledge bases have many gaps that compromise their robustness in handling input with missing pieces of knowledge (Knight and Hatzivassiloglou, 1995). Therefore, they require restrictive defaults or large sets of rules that must be created manually and are restricted by domain. Also defaults can often be wrong and disfluent. Hybrid systems try limiting the space of ambiguity resulting from linguistically blind statistical hypotheses using linguistically motivated, albeit overgenerating, rules. Statistical knowledge is then used to rank and make the final choice. Examples of hybrid NLG systems include Nitrogen (Langkilde and Knight, 1998a,b,c)[1] and FERGUS (Alshawi et al., 2000). On the symbolic–statistical continuum, Lexogen is in the middle: a hybrid approach that borrows from the benefits of the approaches at either end of the continuum. Section 7 compares Lexogen to both Nitrogen and FERGUS.

3. LCS-based MT

One of the major challenges in natural language processing is the ability to make use of existing resources. In this project we have reused existing resources, where feasible, while also creating new reusable resources for multilingual generation. Existing resources include the LCS representation and English lexicon, as well as aspects of the Nitrogen generation system (Langkilde and Knight, 1998b) including its "Abstract Meaning Representation" (AMR) and lattice representations, the morphological generator, and statistical lattice extractor. We have additionally built a language-independent decomposition module, which requires a target-language lexicon in order to be reconfigured for a particular target language, a language-independent realizer, which requires a target-language grammar, and extensions to the representation languages. The modular design of the Lexogen system (see Figures 1 and 6) allows for easy reuse of particular resources, being able to swap in and out different modules that can perform the same function, or use components for tasks other than Chinese–English translation. In particular, we have used three different linearization components, and the same generation system has been used for broad, shallow generation for information retrieval as well as translation.

Reusing resources in large-scale applications can often be difficult, because of the barriers created by large differences in syntax, semantics, and ontologies of these resources. A case in point is the wide range of "interlingual representations" used in MT and cross-language processing. Such representations are becoming increasingly prevalent, yet views vary widely as to what these should be composed of, ranging from purely conceptual knowledge representations that have little to do with the structure of language, to very syntactic representations that maintain most of the idiosyncrasies of the source languages. The Lexogen generation system makes use of resources associated with two different (kinds of) interlingua structures: LCS and AMR. The two representations serve different but complementary roles in the translation/generation process. The deeper lexical-semantic expressiveness of LCS is essential for language-independent lexical selection that transcends

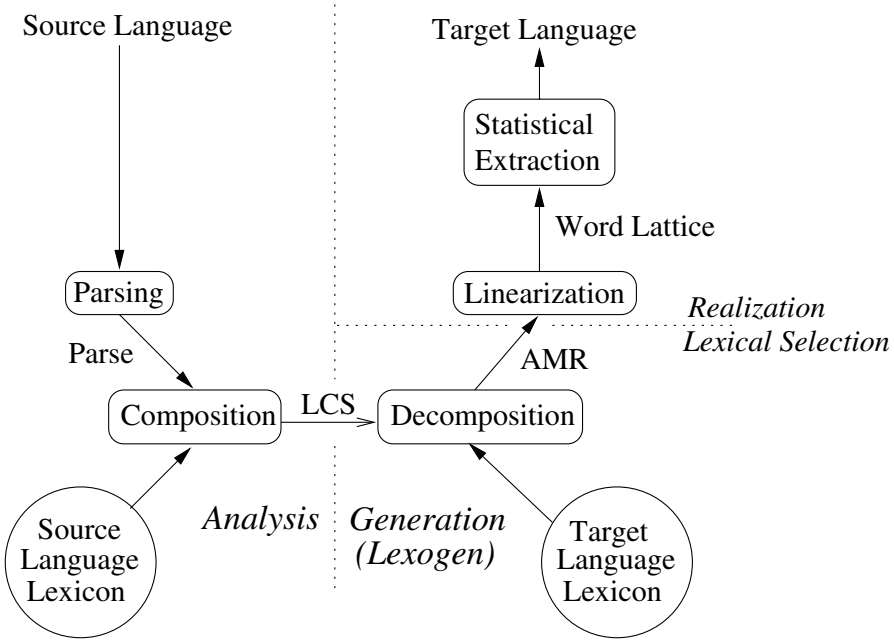

Figure 1. LCS-based MT.

translation divergences (Dorr, 1993a). The shallower yet mixed semantic-syntactic nature of AMRs makes them easier to use directly for target-language realization.

The use of two representations in generation mirrors the use of two representations on the analysis side of LCS-based MT, in which a parsing output is passed to a semantic-composition module; the target-language AMR is analogous to the source-language parse tree (see Figure 1). The Composition module takes the source-language parse tree and creates a deeper semantic representation (the LCS) using a source-language lexicon. During generation, the Decomposition module in Lexogen performs a reverse step that uses a target-language lexicon to create the hierarchical word and feature structure, a "parse-like" AMR. The linearization module flattens an AMR into a sequence of words. Because of the ambiguity inherent in all of the modules involved from the parser to the lexicons, multiple sequences are created, represented compactly in a lattice structure. Lexogen uses the statistical extraction module of the generation system Nitrogen (Langkilde and Knight, 1998b,c) to select among alternative outputs, using n-gram probabilities of target-language word sequences.

The next section discusses the LCS representation in deeper detail.

4. Lexical Conceptual Structure

Linguistic knowledge in the lexicon covers a wide range of information types, such as verbal subcategorization for events (e.g., that a transitive verb such as *hit* occurs

with an object noun phrase), feature information (e.g., that the direct object of a verb such as *frighten* is animate), thematic information (e.g., that *John* is the agent in *John hit the ball*), and lexical-semantic information (e.g., that spatial verbs such as *throw* are conceptually distinct from verbs of possession such as *give*). By modularizing the lexicon, we treat each information type separately, thus allowing us to vary the degree of dependence on each level. For example, the thematic level may be used during lexical selection for determining the main verb of the target-language sentence (*hit* vs. *slap*), whereas the feature level may be used during syntactic realization for determining the type of direct object associated with the selected verb (*ball* vs. *idea*).

The most intricate component of lexical knowledge is the lexical-semantic information, which we encode in the form of LCS as formulated by Dorr (1993b, 1994) based on work by Jackendoff (1983, 1990, 1996). LCS is a compositional abstraction with language-independent properties that transcend structural idiosyncrasies. This representation has been used as the interlingua of several projects such as UNITRAN (Dorr, 1993a) and MILT (Dorr, 1997a).

Formally, an LCS is a directed graph with a root. Each node is associated with certain information, including a "type", a "primitive" and a "field". The type of an LCS node is one of *Event, State, Path, Manner, Property* or *Thing*. There are two general classes of primitives: "closed class" (also called "structural primitives", e.g., `cause`, `go`, `be`, `to`) and "open class" primitives (also called "constants", e.g., `john+`, `reduce+ed`, `jog+ingly`). Suffixes such as `+`, `+ed`, `+ingly` are markers of open class primitives, signaling also the type of the primitive (*Thing*, *Property*, *Event*, etc.). We distinguish between the structural primitive `go`, denoting generalized movement, and the constant `go+ingly`: the first appears in many lexical entries but the second appears only in specific lexical entries such as the one for the English verb *go*. Fields specify the domain of a particular primitive, e.g., `Locational`, `Possessional`, and `Identificational`.[2] Structurally, an LCS node has zero or more LCS children. There are three ways a child node relates to its parent: as a subject (maximally one), as an argument, or as a modifier.[3]

An LCS captures the semantics of a lexical item through a combination of semantic structure (specified by the shape of the graph and its structural primitives and fields) and semantic content (specified through constants). The semantic structure of a verb is something the verb shares with a *semantic verb class* whereas the content is specific to the verb itself. For example, all the verbs in the semantic class of "Run" verbs (*run*, *jog*, *walk*, *zigzag*, *jump*, *roll*, etc.) have the same semantic structure but vary in their semantic content.

It is important to point out that LCS is not a deep knowledge representation. Rather, it is a representation that normalizes structural idiosyncrasies of different languages to provide a consistent methodology for handling translation divergences. Resolving pragmatic and stylistic questions is addressed by other researchers in the field (Hirst, 1987; Hovy, 1988; Olsen, 1997).

Semantic verb classes were initially borrowed from the classification in *English Verb Classes and Alternations* (Levin, 1993). Our LCS Verb Database (LVD) extends Levin's classification by refining the class divisions[4] and defining the underlying meaning components of each class in the LCS representation. LVD also provides a relation between Levin's classes and both thematic role information and hand-tagged WordNet synset numbers. The first public release of the LVD is now available for research purposes (Dorr, 2001).

Consider the sentence (1a). This can be fully represented (except for features such as tense, telicity, etc.) as shown in (1b) glossed as (1c).

(1) a. John jogged to school.

b.
```
(event go loc
   (thing john+)
   (path to loc
      (thing john+)
      (position at loc (thing john+) (thing school+)))
   (manner jog+ingly))
```

c. 'John moved (*location*) to the school in a jogging manner'

Figure 2 shows the lexicon entry for one sense of the English verb *jog* and the preposition *to*.[5] These entries include several pieces of information such as the root form of the word, introduced by the field `:DEF_WORD`, and the word's semantic verb class, introduced by the field `:CLASS`.

The field `:THETA_ROLES` specifies the set of thematic roles appearing in the "Root LCS" (RLCS) entry. Theta roles preceded by an underscore ("_") are obligatory, whereas roles proceeded by a comma (",") are optional. Parentheses indicate that the corresponding phrases must necessarily be headed by a preposition. Sometimes the specific preposition is provided inside the parentheses. The roles are ordered in a canonical order that reflects their relative surface order: first available role is subject; second is object; and so on.

The field `:WN_SENSE` links the entry to its corresponding WordNet synset. The Lexicon entries use WordNet 1.6 senses (Fellbaum, 1998; Miller and Fellbaum, 1991).

The most important field in the lexicon entry is `:RLCS`, which introduces the uninstantiated LCS corresponding to the underlying meaning of the word entry in the lexicon. The top node in the RLCS for *jog* in Figure 2 has the structural primitive `go` in the locational field. Its subject is marked with a star "`*`" indicating that the node must be filled recursively with other lexical entries during semantic composition. The restriction on this particular LCS node is that the filler must be of type `thing`. The number `2` in that node specifies the thematic role, in this case, theme. The second and third child nodes are in argument positions filled with the primitives `FROM` and `TO`. The numbers `3` and `5` stand for source and goal particle respectively. The numbers `4` and `6` stand for source and goal. Table I contains a list

```
(DEFINE-WORD
  :DEF_WORD "jog"
  :CLASS "51.3.2.a.ii"
  :THETA_ROLES "_th,src(),goal()"
  :WN_SENSE (01315785 01297547)
  :LANGUAGE ENGLISH
  :RLCS
   (event go loc (* thing 2)
     ((* path from 3) loc (thing 2)
        (position at loc (thing 2) (thing 4)))
     ((* path to 5) loc (thing 2)
        (position at loc (thing 2) (thing 6)))
     (manner jog+ingly 26))
  :VAR_SPEC ((3 :optional) (5 :optional)))

(DEFINE-WORD
  :DEF_WORD "to"
  :LANGUAGE ENGLISH
  :LCS (path to loc
         (thing 2)
         (position in loc (thing 2) (* thing 6))))
```

Figure 2. Lexicon entries for *jog* and *to*.

of variable numbers with their associated thematic roles. The second argument in the *jog* RLCS is the substructure `(to loc ...)` that unifies with the RLCS for the preposition *to*. This secondary RLCS itself has a star-marked argument that must be instantiated with a `thing` such as *school*.

The variable specifications (indicated here as `:VAR_SPEC`) assign the arguments headed by `FROM` and `TO` an `:optional` status. Other possible variable specifications that appear in our lexicon include `:obligatory`, `:promote`, `:demote`, `:EXT` (external), `:INT` (internal) and `:conflated` (see Dorr (1993a) for more details).

The current English lexicon contains 10,000 RLCS entries such as those in Figure 2 (see also Figure 7 below). These entries correspond to different senses of over 4,000 verbs. Figure 3 compares four of the nine RLCS entries for the verb *run*. These entries are classified by verb class. Verb classes are used as templates to generate the RLCS entries of verbs in the class. For example, the lexical entry for *bake* in class 26.3 would be identical to the top RLCS entry shown in Figure 3, except that node 9 would instead contain the primitive `bake+ed` rather than `run+ed`.

As described in Dorr (1993b), the meaning of complex phrases is captured through a "composed LCS" (CLCS). A CLCS is constructed (or composed) from several RLCS entries corresponding to individual words. The composition process starts with a parsed tree of the input sentence and maps syntactic leaf nodes into

Table I. Inventory of thematic roles.

#	Thematic Role	Definition
0		No thematic role assigned
1	AG	Agent
2	TH, EXP, INFO	Theme or experiencer or information
3	SRC()	Source preposition
4	SRC	Source
5	GOAL(), PRED()	Goal or pred preposition
6	GOAL	Goal
7	PERC()	Perceived item particle
8	PERC	Perceived item
9	PRED	Identificational predicate
10	LOC()	Locational particle
11	LOC	Locational predicate
12	POSS	Possessional predicate
13	TIME()	Temporal particle preceding time
14	TIME	Time for TEMP field
15	MOD-POSS()	Possessional particle
16	MOD-POSS	Possessed item modifier
17	BEN()	Beneficiary particle
18	BEN	Beneficiary modifier
19	INSTR()	Instrumental particle
20	INSTR	Instrument modifier
21	PURP()	Purpose particle
22	PURP	Purpose modifier or reason
23	MOD-LOC()	Location particle
24	MOD-LOC	Location modifier
25	MANNER()	Manner
26		Reserved for conflated manner
27	PROP	Event or state
28	MOD-PROP	Event or state
29	MOD-PRED()	Identificational particle
30	MOD-PRED	Property modifier
31	MOD-TIME	Time modifier

26.3 Verbs of Preparing

```
(event cause (* thing 1)
   (event go ident (* thing 2)
      (path toward ident (thing 2)
         (position at ident (thing 2) (property run+ed 9))))
   ((* for 17) poss (*head*) (* thing 18)))
```

Example: *John ran the store for Mary.*
Other verbs: *bake boil clean cook fix fry grill iron mix prepare roast roll run wash ...*

47.7.a Meander Verbs (from to)

```
(event go_ext loc (* thing 2)
   ((* path from 3) loc (thing 2) (position at loc (thing 2) (thing 4)))
   ((* path to 5) loc (thing 2) (position at loc (thing 2) (thing 6)))
   (manner run+ingly 26))
```

Example: *The river runs from the lake to the sea.*
Other verbs: *crawl drop go meander plunge run sweep turn twist wander ...*

47.5.1.b Swarm Verbs (Locational)

```
(event act loc (* thing 2)
   ((* position [at] 10) loc (thing 2) (thing 11))
   (manner run+ingly 26))
```

Example: *The dogs run in the forest.*
Other verbs: *bustle crawl creep run swarm swim teem ...*

51.3.2.a.i Run Verbs (Locational,Theme only)

```
(event go loc (* thing 2)
   ((* path from 3) loc (thing 2) (position [at] loc (thing 2) (thing 4)))
   ((* path to 5) loc (thing 2) (position [at] loc (thing 2) (thing 6)))
   (manner run+ingly 26))
```

Example: *The horse ran into the field from the barn.*
Other verbs: *climb crawl fly jog jump leap race run swim walk ...*

Figure 3. Different RLCS entries for *run*.

RLCS entries whose argument positions are filled with other RLCS entries. For example, the two RLCS entries we have seen already can compose together with the constants for *John* and *school* to give the CLCS for the sentence (1a), shown in (1b) above. The star-marked node `(* path from 3)` is optional, and is left unfilled in this case. The same RLCS could also be used to compose different CLCS representations (in combination with other RLCS entries) to produce sentences like those in (2).

(2) a. John jogged from home.
    b. John jogged from home to school.

A CLCS can also be decomposed on the generation side in different ways depending on the RLCS entries from the target language. Figure 4 uses a compressed graphical representation of LCS to compare visually three different decompositions in three languages of a single CLCS. The CLCS generated can be paraphrased as (3).

(3) John caused himself to go to the inside of a room in a forceful manner.

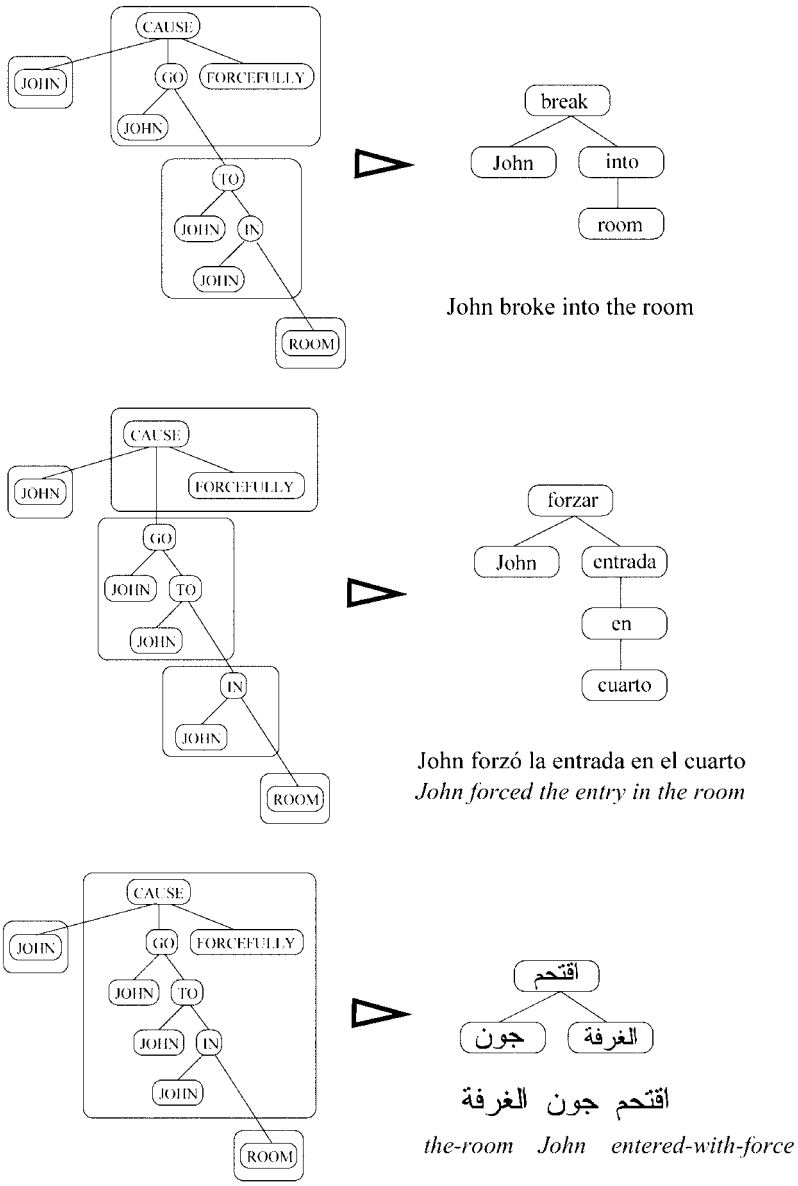

Figure 4. Different CLCS decompositions into English, Spanish and Arabic.

The input to Lexogen is a text representation of a CLCS in a format called "longhand". It is equivalent to the form shown in (1b), but makes certain information more explicit and regular (at the price of increased verbosity). The CLCS can be either a fully language-neutral interlingua representation, or one which still incorporates some aspects of the source-language interpretation process. The latter may include grammatical features on LCS nodes as well as "functional

nodes", which correspond to words in the source language that have no lexical content, serving merely as place-holders for feature information. Examples of these nodes include punctuation markers, coordinating conjunctions, grammatical aspect markers, and determiners.

An important new extension to the LCS input language is the in-place representation of ambiguous subtrees, using a "Possibles" node – denoted `:possibles` – which specifies the various possible candidate subtrees that could appear at a given point. Each candidate is a child of the Possibles node, and a fully disambiguated CLCS would include exactly one of the candidates in place of each Possibles node. For example, the structure in (4) (with some aspects elided for brevity) represents a node that could be one of three possibilities resulting from an ambiguous Chinese parse of *among developing countries*. The first and third possibilities are incorrect analyses. In the second, the root of the subtree is a functional node, passing its features to its child, `country+` in a fully language-neutral CLCS.

(4)
```
(:possibles
    (middle+ (country+ (developing+/p)))
    (functional (postposition among)
                (country+ (developing+/p)))
    (china+ (country+ (developing+/p))))
```

Possibles nodes can appear at any point in the LCS tree, indicating a set of subtrees that are "possible" fillers for the Possibles node. As with other nodes, Possibles may appear in multiple places in the LCS tree, and in the case of Possibles nodes, the same choice must be made for each appearance in the tree, in a given fully expanded LCS.

It is important to point out that in ChinMT, sentences were not quite as simple as the examples used so far to explain the LCS approach. Figure 5 displays a CLCS from ChinMT that was derived from the Chinese sentence in (5).

(5) 在第21届东新澳中央银行组织行长研讨会上，中国人民银行副行长殷介炎就“资本大量流入情况下宏观经济政策的协调”问题发表意见

zai di 21 jie dong-xing-ao zhongyang yinhang zuzhi hangzhang yantaohui shang, zhongguo renmin yinghang fu hangzhang yin jieyan jiu " ziben daliang liuru qingkuang xia hongguan jingji zhence de xietiao " wenti facbiao yijian

AT card 21 SESSION SEA-SINGAPORE-MACAO CENTRAL BANK ORGANIZATION PRESIDENT SEMINAR ON, CHINA-PEOPLE'S-BANK DEPUTY PESIDENT YINJIEYAN THEN " CAPITAL LARGE INFLUX SITUATION UNDER LARGE ECONOMY POLICY sub AGREEMENT " QUESTION EXPRESS OPINION

'At the 21st Southeast Asia–Singapore–Macao Central Bank Organization Presidents' Symposium, vice president of the People's Bank of China Yin

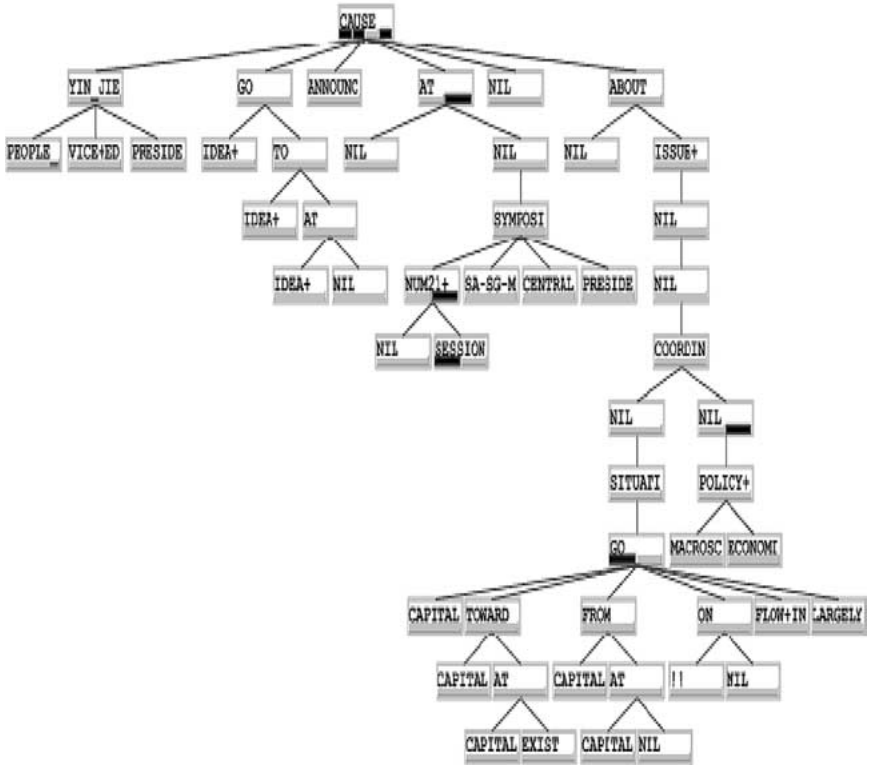

Figure 5. Large-scale CLCS for sentence (5).

> Jieyan expressed his opinion on "coordination of macro-economic policy with a large capital inflow".'

Figure 5 is a static view from a dynamic LCS viewer, which compactly shows one of many possible ambiguous readings, while providing access to other readings as well as an indication of other possibilities. Possibles nodes are represented as nodes with small black boxes underneath, one box per possibility, with a particular choice shown in the node and subtree below. By clicking on another of the boxes, another view is shown, changing the node and subtree presented for this Possibles node. For example, in Figure 5, the top node has four distinct possibilities corresponding to the verbs *issue*, *publish*, and *announce* (two instances of the latter). The number of distinct possible CLCS representations in this example is 128, with an average of 50 nodes per CLCS. Compare these figures to those for the example in Figure 4: zero ambiguity, one CLCS, and ten nodes.

The rest of the examples in this paper will refer to a less complex sentence (6).

(6) 美 单方 削减 中国 纺织品 出口
mei danfang *xuejian* *zhongguo fangzhipin* *chukou*
US UNILATERAL REDUCE CHINA TEXTILE-PRODUCT EXPORT
配额
pei'e
QUOTA
'The United States unilaterally reduced the China textile export quota.'

The representation for this example is shown in (7a), which roughly corresponds to the gloss (7b). This LCS is presented here without all the additional features, or type and function markers, for sake of clarity. Moreover, it is only one of eight possible LCS compositions produced by the analysis component from the input Chinese sentence (6).

(7) a.
```
(cause (united_states+)
  (go ident (quota+ (china+) (textile+) (export+))
    (toward ident (quota+ (china+) (textile+) (export+))
      (at ident (quota+ (china+) (textile+) (export+))
                (reduce+ed))))
  (with instr (*HEAD*) nil)
  (unilaterally+/m))
```

b. 'in a unilateral manner, the United States caused the quota (modified by China, textile and export) to go identificationally (or transform) towards being at the state of being reduced'

5. The Lexogen Generation System

The architecture of the Lexogen generation system is presented in Figure 6, showing the main modules and submodules, and flow of information between them. In the generation process, the first phase, "Lexical Choice", uses language-specific lexicons that relate lexical items in the target language to their LCS representation. The output of this phase is a target-language representation of the sentence in a modified form of the AMR interlingua called LCS-AMR. The second phase, "Realization", first handles the linearization and morphology to generate lattices of target-language sequences from the LCS-AMR and then statistically extracts preferred sequences using a bigram language model. For linearization, we use our own language-independent linearization engine, Oxygen (Habash, 2000). For the statistical extraction (and morphological generation), we use the Nitrogen generation system, from ISI (Langkilde and Knight, 1998b,c).

As mentioned in Section 3, the modular design of Lexogen allows for independent usage of each of its components. Each component operates on its input and produces its output, which could come from the components indicated in Figure 6,

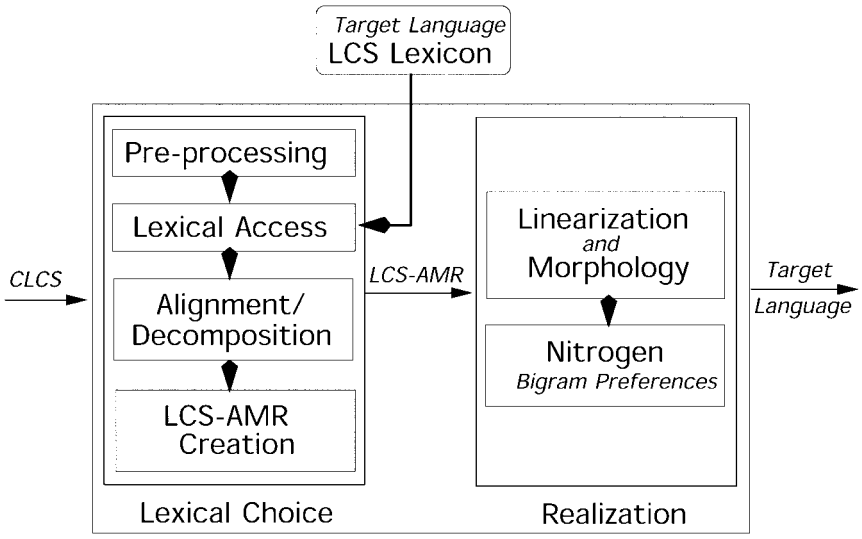

Figure 6. Lexogen architecture.

or other components that produce the appropriate input structures. We have used this advantage in maintaining and improving the system. For example, we have tested three different linearization components, and can use a separate realizer, which produces all output sentences rather than the best one according to n-gram statistics.

5.1. LEXICAL CHOICE

The first major component, divided into four pipelined submodules as shown in Figure 6, transforms a CLCS structure into an LCS-AMR structure. This new representation is a modified form of the AMR interlingua that uses words and features specific to the target language, and also includes syntactic and semantic information from the LCS representation that is relevant for realization.

5.1.1. *Pre-processing*

The pre-processing phase converts the text input format into an internal graph representation for efficient access of components (with links for parents as well as children). This phase also removes extraneous source-language features, such as the distinction between postposition and preposition, or classifier type information. For example, the CLCS in (4) is converted during pre-processing to remove the functional node and promote `country+` to be the head of one of the possible subtrees. This involves a top-down traversal of the tree, including some complexities when functional nodes without children (which then assign features to their parents) are direct children of Possibles nodes. There is no ASCII-printable form for the data-structure output of the pre-processing phase; however, the result of

post-processing the CLCS in (4) would be equivalent to the CLCS in (8), with a feature "among" as part of the node country+ in the middle reading.

(8)
```
(:possibles
    (middle+ (country+ (developing+/p)))
    (country+  (developing+/p))
    (china+ (country+ (developing+/p))))
```

5.1.2. *Lexical Access*

The lexical access phase compares the internal CLCS form to the target-language lexicon, "decorating" the CLCS tree with the RLCS entries of target-language words which are likely to match substructures of the CLCS. The matching between a given CLCS and the target-language lexicon is potentially a complex process, given the large amount of structural similarity between the entries of the lexicon. For example, the differences between the RLCS entries for *run* and *bake* in class 26.6 would be distinguished only by looking down five nodes deep from the root (cf. Figure 3 and the discussion of verb classes above). In a previous version of the system, we represented the lexicon in a trie structure, so that individual entries were only consulted at appropriate points in the CLCS tree traversal. This still proved a fairly complex and inefficient procedure given the large number of places that complex structures can be embedded (e.g., complement events). Our current approach uses a multi-phase process, in which RLCS entries are first located based on the distinguishing information (e.g., run+ed vs. bake+ed) and then placed in the appropriate matching node (CAUSE) for later comparison.

The lexical-access process thus proceeds as follows. In an off-line lexicon processing phase, each word in the target-language lexicon is stored in a hash table, with each entry keyed on a designated primitive, an "anchor" that is the most distinguishing node in the RLCS. Information is also kept about how deep from the root of the RLCS this primitive's node is to be found. For example, the designated primitive for the RLCS entries corresponding to class 26.3 would be run+ed (or bake+ed), and the depth would be 5. On-line decoration then proceeds in a two-step process, recursively examining each node in the CLCS:

1. Look for RLCS entries stored in the lexicon under the CLCS node's primitive.
2. Store retrieved RLCS entries at the node in the CLCS that matches the root of this RLCS (following a number of parent links from the CLCS node corresponding to the depth of the designated primitive).

Figure 7 shows some of the English entries matching the CLCS in (7a). For most of these words, the designated primitive is the only node in the corresponding LCS for that entry. For *reduce*, however, reduce+ed is the designated primitive. When traversing the CLCS nodes in (7a), this entry will be retrieved at the reduce+ed node in step 1; it will then be stored at the root node of the LCS in (7a) in accordance with step 2.

```
(:DEF_WORD "reduce"
  :CLASS "45.4.a"
  :THETA_ROLES "_ag_th,instr(with)"
  :WN_SENSE (00154752 00162871 00163072 00163532)
  :LANGUAGE ENGLISH
  :LCS (event cause (* thing 1)
        (event go ident (* thing 2)
           (path toward ident (thing 2)
              (position at ident (thing 2) (reduce+ed 9))))
        ((* position with 19) instr (*head*) (thing 20)))
  :VAR_SPEC ((1 (animate +))))

(:DEF_WORD "United States" :LCS (thing united_states+ 0))

(:DEF_WORD "China" :LCS (thing china+ 0))

(:DEF_WORD "quota" :LCS (thing quota+ 0))

(:DEF_WORD "with"
  :LCS (position with instr (thing 2) (* thing 20)))

(:DEF_WORD "unilaterally"
  :LCS (manner unilaterally+/m 0))
```

Figure 7. Lexicon entries.

5.1.3. *Alignment/Decomposition*

The heart of the lexical choice phase is the "decomposition" process. In this phase, we attempt to align RLCS entries selected by the lexical access portion with parts of the CLCS, to find coverings of the CLCS graph that satisfy the "full coverage constraint" of the original algorithm described in Dorr (1993b). LCS "graph matching" is a special case of generalized graph matching. In LCS graph matching we exploit the constraints inherent in the LCS representation such as the availability of anchoring points discussed earlier to increase the efficiency of the matching algorithm without any effect on final quality. Our algorithm differs from that in Dorr (1993b) in its inclusion of some extensions to handle the in-place ambiguity represented by the Possibles nodes.

The algorithm recursively checks whether CLCS nodes match corresponding RLCS nodes coming from the lexical entries retrieved and stored in the previous phase. If incompatibilities are found (obligatory nodes of the RLCS do not match any nodes in the CLCS input), then the RLCS lexical entry is discarded. If, on the other hand, all (obligatory) nodes in the RLCS match against nodes in the CLCS,

then the rest of the CLCS is recursively checked against other lexical entries stored at the remaining unmatched CLCS nodes.

A CLCS node matches an RLCS node, if the following conditions hold:

- The primitives are the same (or the primitive for one is a wild-card, represented as `nil`).
- The types (*Thing*, *Event*, *State*, etc.) are the same (or `nil`).
- The fields (identificational, possessive, locational, etc.) are the same (or `nil`).
- The positions (e.g., subject, argument, or modifier) are the same.
- All obligatory children of the RLCS node have corresponding matches (recursively invoking this same definition) to children of the CLCS.

Star-marked nodes in an RLCS (see discussion above) require not just a match against the corresponding CLCS node, but also a match against another lexical entry. Thus, in (7a), the node (`united_states+`) must match not only with the corresponding node from the RLCS for *reduce* in Figure 7 `(* thing 1)`, but also with the RLCS for *United States*, `(thing united_states+ 0)`.

Subject and argument children are obligatory unless specified as optional, whereas modifiers are optional unless specified as obligatory (see Figure 2 for an example of an optional marking). In the RLCS for *reduce* in Figure 7, the nodes corresponding to agent and theme (numbered 1 and 2, respectively) are obligatory, while the instrument (the node numbered 19) is optional. Thus, even though there is no matching lexical entry for node 20 (star-marked in the RLCS for *with*), the main RLCS for *reduce* is allowed to match, though without any realization for the instrument.

A complexity in the algorithm occurs when there are multiple possibilities, i.e., a Possibles node in a CLCS. In this case, only one of these possibilities is required to match all the corresponding RLCS nodes in order for a lexical entry to match. In the case where some of these possibilities do not match any RLCS nodes (meaning there are no target-language realizations for these constructs), these possibilities are pruned at this stage. On the other hand, ambiguity can also be introduced at the decomposition stage, if multiple lexical entries can match a single structure.

The result of the decomposition process is a match structure indicating the hierarchical relationship between all lexical entries which, together, cover the input CLCS.

5.1.4. *LCS-AMR Creation*

The match structure resulting from decomposition is then converted into the appropriate input format used by the Nitrogen generation system. Nitrogen's input, AMR, is a labeled directed feature graph written using the syntax for the Penman Sentence Plan Language (Penman, 1989). A BNF structural description of an AMR is shown in (9).

(9) AMR = <concept> | (<label> / <concept> {<role> <AMR>}*)

```
(a7537 / |reduce|
    :LCS-NODE 6253520
    :LCS-VOICE ACTIVE
    :CAT V
    :TELIC +
    :LCS-AG (a7538 / |United States|
        :LCS-NODE 6278216
        :CAT N)
    :LCS-TH (a7539 / |quota|
        :LCS-NODE 6278804
        :CAT N
        :LCS-MOD-THING (a7540 / |China|
            :LCS-NODE 6108872
            :CAT N)
        :LCS-MOD-THING (a7541 / |textile|
            :LCS-NODE 6111224
            :CAT N)
        :LCS-MOD-THING (a7542 / |export|
            :LCS-NODE 6112400
             :CAT N))
    :LCS-MOD-MANNER (a7543 / |unilaterally|
        :LCS-NODE 6279392
        :CAT ADV))
```

Figure 8. LCS-AMR corresponding to the CLCS in (7a).

An AMR is either a basic concept such as |run|, |john| or |quickly| or a labeled instance of a concept that is modified by a set of feature–value pairs. Features, or "roles", can be syntactic (such as :subject) or semantic (such as :agent). The basic notation "/" is used to specify an instance of a concept in a non-ambiguous AMR.

We have extended the AMR language to accommodate the thematic roles and features provided in the CLCS representation; the resulting representation is called an LCS-AMR. We use the prefix :LCS- to distinguish the LCS terms from those used by Nitrogen which have similar names. Figure 8 shows the LCS-AMR corresponding to the CLCS in (7a), decomposed using the lexicon entries in Figure 7.

The LCS-AMR in Figure 8 can be read as an instance of the concept |reduce| whose category is a verb and whose voice is active. The concept |reduce| has two thematic roles related to it, an agent (:LCS-AG) and a theme (:LCS-TH); and it is modified by the concept |unilaterally|. The different roles modifying |reduce| come from different origins. The :LCS-NODE value comes directly from the unique node number in the input CLCS. The category, voice and telicity are

derived from features of the RLCS entry for the verb *reduce* in the English lexicon. The specifications `agent` and `theme` come from the RLCS representation of the verb *reduce* in the English lexicon as well, as can be seen by the node numbers 1 and 2, in the lexicon entry in Figure 7. The role `:LCS-MOD-MANNER` combines the fact that the corresponding AMR has a modifier role in the CLCS and that its type is a Manner.

We have additionally extended the AMR syntax by providing the ability to specify an ambiguous AMR as an "instance-less" conglomeration of different AMRs; this is achieved by means of the special role `:OR`. For example, a variant of the LCS-AMR in Figure 8 in which the root concept is three-way ambiguous would appear as in (10) (details below the root omitted). `:OR` represents the same sort of subtree ambiguity in LCS-AMRs as Possibles nodes represent in LCSs. In addition, there is an AMR construct `*or*`, which represents simple ambiguity of objects (rather than whole subtrees).

(10)
```
(# :OR (# / |reduce| . . . )
   :OR (# / |cut| . . . )
   :OR (# / |decrease| . . . ))
```

5.2. REALIZATION

The lexical choice phase concludes with the production of an LCS-AMR, as described in the previous section. This LCS-AMR representation is then passed to the realization module, which produces target-language strings for the complete sentence. The realization module uses the Nitrogen approach to generation: allowing overgeneration of possible sequences of target-language words from the ambiguous or underspecified AMRs and then deciding amongst them based on *n*-gram frequency. The interface between the linearization module and the statistical extraction module is a word lattice of possible renderings. The Nitrogen package offers support for both subtasks, linearization and statistical extraction. Initially, we used the Nitrogen grammar to do linearization, but complexities in recasting the LCS-AMR roles as standard AMR roles as well as efficiency considerations (that will be discussed later in detail) compelled us to create our own linearization engine for writing target-language grammars, Oxygen (Habash, 2000).

In the realization module, we force linear order on the unordered parts of an LCS-AMR. This is done by recursively calling grammar rules that create various phrase types (NP, PP, etc.) from aspects of the LCS-AMR. The result of the linearization phase is a word lattice specifying the sequence of words that make up the resulting sentence and the points of ambiguity where different generation paths may be taken. Example (11) shows the word lattice corresponding to the LCS-AMR in Figure 8.

(11)
```
(SEQ (WRD "*start-sentence*" BOS)
     (WRD "united states" NOUN)
     (WRD "unilaterally" ADJ)
     (WRD "reduced" VERB)
     (OR (WRD "the" ART)
         (WRD "a" ART)
         (WRD "an" ART))
     (WRD "china" ADJ)
     (OR (SEQ (WRD "export" ADJ)
              (WRD "textile" ADJ))
         (SEQ (WRD "textile" ADJ)
              (WRD "export" ADJ)))
     (WRD "quota" NOUN)
     (WRD "*end-sentence*" EOS))
```

The keyword `SEQ` specifies that what follows is a list of sublattices in their correct linear order. The keyword `OR` specifies the existence of disjunctive paths for generation. In our example, the noun *quota* is given a disjunction of all possible determiners since its definiteness is not specified. Also, the relative order of the words *textile* and *export* is not resolved so both ordering possibilities are inserted into the lattice.

Finally, the Nitrogen statistical extraction module evaluates the different paths represented in the word lattice and orders the different word renderings using uni- and bigram frequencies calculated based on two years of the *Wall Street Journal* (Langkilde and Knight, 1998c). Example (12) shows Nitrogen's ordering of the sentences extracted from the lattice in (11).

(12) United States unilaterally reduced the China textile export quota.
United States unilaterally reduced a China textile export quota.
United States unilaterally reduced the China export textile quota.
United States unilaterally reduced a China export textile quota.
United States unilaterally reduced an China textile export quota.
United States unilaterally reduced an China export textile quota.

The next two sections discuss linearization issues and linearization implementation using Oxygen.

5.2.1. *Linearization Issues*

The unordered nature of siblings under an LCS-AMR node complicates the mapping between roles and their surface positions, yielding several interesting linearization issues. In this section, we look at some of the choices made for our English realizer for ordering linguistic constituents.

5.2.1.1. *Sentential Level Argument Ordering* English sentences are realized according to the pattern in (13). That is, first subordinating conjunctions, if any, then

modifiers in the temporal field (e.g., *now*, *in 1978*), then the subject, then most other modifiers, the verb (with collocations if any) then spatial modifiers (*up*, *down*), then the indirect object and direct object, followed by prepositional phrases and relative clauses. In the case of multiple modifiers corresponding to a specific part of the pattern, for example temporal modifiers, all permutations of these modifiers are realized. This specific linear order pattern was produced by consulting standard grammatical sources (e.g., Quirk et al. 1985) as well as experimentation with different variants. Nitrogen's morphology component was also used, for example, to give tense to the head verb. In the example above, since there was no tense specified in the input LCS, the telicity of the verb was used to infer past tense, thus producing *reduced* in (11)–(12).[6]

(13) (SubConj ,) (TempMod)* Subj (Mod)* V (coll) (SpaceMod)* (IObj) (Obj) (PP)* (RelS)*

5.2.1.2. *Thematic Role Ordering* Given the general shape for a sentence (13), there is still an issue of which thematic role should be mapped to which argument positions. This situation is complicated by the lack of one-to-one mapping between a particular thematic role and an argument position. For example, a theme can be the subject in some cases and it can be the object in others or even an oblique. Observe *cookie* in (14).

(14) a. John ate *a cookie*. (object)

b. *The cookie* contains chocolate. (subject)

c. She nibbled *at a cookie*. (oblique)

To solve this problem, a thematic hierarchy is used to determine the argument position of a thematic role based on its co-occurence with other thematic roles. Several researchers have proposed different versions of thematic hierarchies (Jackendoff, 1972; Giorgi, 1984; Nishgauchi, 1984; Carrier-Duncan, 1985; Kiparsky, 1985; Bresnan and Kanerva, 1989; Larson, 1988; Wilkins, 1988; Baker, 1989; Alsina and Mchombo, 1993; Grimshaw and Mester, 1988).[7] Ours differs from these in that it separates arguments (e.g., agent and theme) from obliques (e.g., location and beneficiary) and provides a more complete list of thematic roles (30 roles overall, see Table I) than those of previous approaches (maximum of eight roles).[8]

The final thematic hierarchy for arguments, shown in (15), was extracted by analyzing subcategorization information in the `:THETA_ROLES` field for all the verbs in our English lexicon. The thematic hierarchy is paired with a syntactic hierarchy that specifies the syntactic positions the thematic roles are mapped to.

(15) Thematic Hierarchy:
ext > ag > instr > th > perc > goal > src > ben > *

Syntactic Hierarchy:
Subject > Object > Indirect-Object

Thus, in the case where a theme occurs alone, this role is mapped to the subject position, which is then realized as the first argument position, as in (16a). If a theme and an agent co-occur, the agent is mapped to the subject position and the theme is mapped to the object position. Subject and object are realized as first and second argument positions respectively, as in (16b). When three roles co-occur, typically an agent and a theme occurring with a goal, a source or a beneficiary, they are mapped to subject, object and indirect object respectively. Of course, the surface order of the syntactic positions, as described in (13), forces the indirect object to appear before the direct object, as in (16c). In this example, the decision to realize $Mary_{ben}$ as an argument not an oblique (i.e., a PP such as *for Mary*) is done as part of the decomposition step in the lexical choice module (Section 5.1). In the alternative case, only the non-oblique arguments, the agent and theme, are processed through the thematic hierarchy. As for the ordering of obliques, all possible permutations are generated after the arguments.

(16) a. $Cookies_{th}$ are cheap.
b. $John_{ag}$ baked $cookies_{th}$.
c. $John_{ag}$ baked $Mary_{ben}$ $cookies_{th}$.

The pseudo-role `ext` is used when the `:VAR_SPEC` field in the lexical entry of a verb includes an `:EXT` marker indicating that the verb violates the normal thematic hierarchy. The `ext` marker refers to an externally marked thematic role such as the perceived *John* in (17). For the LCS-AMR in Figure 8, the thematic hierarchy is what determines that *United States* is the subject and *quota* is the object of the verb *reduce*. A more detailed discussion is available in Dorr et al. (1998) and Habash and Dorr (2001b). We will return to discuss thematic hierarchies later in this paper when evaluating English and Spanish realization.

(17) $John_{perc}$ pleases $Mary_{th}$.

5.2.1.3. *NP Modifier Ordering* In most cases, our input CLCS representations had little hierarchical information about multiple modifiers of a noun. Our initial, brute-force solution was to generate all permutations and depend on the existing statistical extraction (in Nitrogen) to decide amongst them. This technique worked well for noun phrases of about six words, but was too costly for larger phrases (of which there were several examples in our test corpus). We improved both the cost of permutation generation and the fluency of the top choices by ordering adjectives within classes, inspired by the adjective ordering scheme in Quirk et al. (1985). The classification scheme is shown in Table II. Each adjective in the target-language lexicon was assigned to one of these classes.

Table II. Classification of adjectives.

Type	Examples
Determiner	*all*, *few*, *several*, *some*
Most adjectival	*important*, *practical*, *economic*
Age	*old*, *young*
Color	*black*, *red*
Participle	*confusing*, *adjusted*, *convincing*, *decided*
Provenance	*China*, *southern*
Noun	*Bank_of_China*, *difference*, *memorandum*
Denominal	nouns made into adjectives by adding *-al*, e.g., *individual*, *coastal*, *annual*

If multiple words fall within the same group, permutations are generated for them. This situation can be seen for the LCS-AMR in Figure 8 with the ordering of the modifiers of the word *quota*: *China*, *export* and *textile*. The word *China* fell within the Provenance class of modifiers which gives it precedence over the other two words. The words *export* and *textile*, on the other hand, fell in the Noun class and therefore both permutations were passed on to the statistical component. Without this ordering, more permutations would be given to the statistical component, which, in this case, would also get a less appropriate result: **textile China export quota* rather than *China textile export quota*.

Similar solutions have been used before by other generation systems (Mann and Matthiessen, 1985; Shaw and Hatzivassiloglou, 1999) albeit never within a hybrid NLG approach as is done here. Another approach to modifier ordering that is fully statistical was proposed by Malouf (2000).

5.2.2. *Oxygen Linearization Implementation*

The linearization module is an implementation of a grammar (comprised of a set of rules) governing the relative word ordering (syntax) and word form (morphology) of an LCS-AMR in the target language. This section discusses Oxygen (Habash, 2000), the linearization engine used to build the target-language linearization module in Lexogen. To highlight the advantages of Oxygen, we preface the discussion with brief descriptions of two other linearization modules that we have used: one from Nitrogen, and an English linearizer written in Lisp.

5.2.2.1. *Nitrogen Linearization* The Nitrogen generation system provides its own linearization module in which a linearization engine performs on-line interpretation of a linearization grammar. The grammar is written in a special grammar description language that utilizes two basic operations: “recast” and “linearize”. A recast transforms an AMR into another AMR based on features of the original

AMR. One example of recasting is the conversion of an AMR containing thematic roles into an AMR containing arguments in their surface position through the use of a thematic hierarchy. A linearize decomposes an AMR into linearly ordered constituents, recursively applying the grammar to each. The grammar description language provides tools for defining conditions on which to make decisions to recast and/or linearize an AMR.

The advantages of this approach are reusability, extensibility, and language independence. Its main drawback is speed. Another drawback for Nitrogen's linearization grammar is a limited and inflexible grammar formalism. First, conditions of application are limited to equality of concepts or existence of roles at the top level of an AMR only. Second, recasting operations are limited to adding feature–value pairs and introducing new nodes. Finally, there is no mechanism to perform range-unbounded or computationally complex transformations such as multiplication or division to format numbers correctly in the target language. The first two issues can be addressed by writing multiple rules and cascading information in order to implement complex decisions. However, this solution increases the size of the grammar and further reduces the performance speed. As for the third issue, handling complex transformations is simply impossible to implement with the current formalism. A deeper look at these issues is provided in Habash (2000).

5.2.2.2. *Lisp-based Linearization* To overcome Nitrogen's drawbacks of speed, limitations on size of input structures, and inflexibility of transformation constructs, we built a linearizer directly in Common Lisp, implementing the grammar rules using special-purpose functions. While this overcomes the drawbacks, it has its own disadvantage in that this hard-coding of grammar rules can make the system rather redundant, difficult to understand and debug, non-reusable and language specific. After exploring both approaches in the Lexogen system, we adopted a hybrid implementation that maximizes the advantages and minimizes the disadvantages of the previous two linearizers. The result is the linearization module Oxygen.

5.2.2.3. *Oxygen Linearization* Oxygen uses a linearization grammar description language to write grammar rules which are then compiled into a programming language for efficient performance. Oxygen contains three elements: a linearization grammar description language (OxyL), an OxyL-to-Lisp compiler (oxyCompile) and a run-time support library (oxyRun). Except for Nitrogen's morphological generator submodule, all of the Oxygen components were built in our lab. Target-language linearization grammars written in OxyL are compiled off-line into Oxygen Linearizers using oxyCompile (Figure 9).

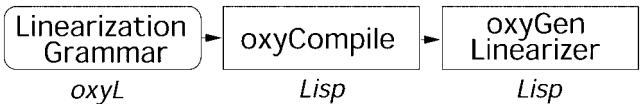

Figure 9. Oxygen compilation step.

Oxygen Linearizers are Lisp programs that require the oxyRun library of basic functions in order to execute (Figure 10). They take AMRs as input and create word lattices that are passed to a statistical extraction unit.

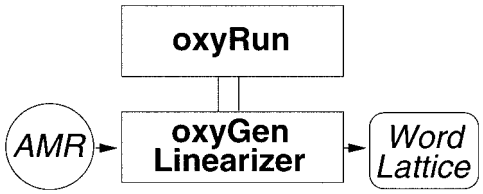

Figure 10. Oxygen runtime step.

The separation of the linearization engine (oxyRun) from the linearization grammar (OxyL) combines in one system the best of two worlds: the simplicity and focus of a declarative grammar with the power and efficiency of a procedural implementation. The approach provides language independence and reusability since target-language-specific aspects are contained only in the specific OxyL grammar. The separation of language-specific code (compiled OxyL) from language-independent code (oxyRun) is efficient, especially when running multiple linearizers for different languages at the same time as in multilingual generation.

Moreover, Oxygen's linearization grammar description language, OxyL, is as powerful as a general-purpose programming language but with a focus on linearization needs. This is accomplished through providing powerful recasting mechanisms and allowing the embedding of code in a standard programming language (Lisp). This allows for efficient implementation of a variety of realization problems such as number formatting. OxyL linearization grammars are also simple, clear, concise and easily extensible. An example of the simplicity of OxyL grammars is the reduction of redundancy. For example, the handling of `:OR` ambiguities in each phrase rule (see (10) for example) is hidden from the linearization grammar designer and is treated only in the compiler and support library. For a detailed presentation of OxyL's syntax, see Habash (2001).

Figure 11 presents a small OxyL grammar fragment that is enough to linearize the LCS-AMR in Figure 8. In this grammar, the user-defined recast operation `&TH-order` uses the OxyL special hierarchical recast operator "`<!`" to recast a small hierarchy of (agent, instrument, theme, source and goal) into subject and object positions. Rules `%S` and `%NP` linearize the different LCS-AMRs associated with specific categories (V and N respectively). Tokens prefixed with "`@`" are pointers to specific LCS-AMRs. For example, `@subj` refers to the LCS-AMR paired with the role `:subj`. However, note that since `@lcs-mod-thing` matches three roles (i.e. *China*, *export* and *quota*), an ambiguous LCS-AMR is created and all its permutations are explored linearly. This is done at the engine level and is hidden from the user.

A linearization can specify hard-coded elements such as the determiners in `%NP`. Alternatives are generated by inserting an `*or*`ed list of words or by using the `OR`

```
:Recast &TH-order
  (@this  <! ((:subj :obj) /
        (:lcs-ag :lcs-instr :lcs-th :lcs-src :lcs-goal)))
:Rule %S
  (-> (OR (@subj (@inst +- past) @obj @lcs-mod-manner)
          (@obj (*or* "was" "were") (@inst +- pastp)
                "by" @subj @lcs-mod-manner)))
:Rule %NP
  (-> ((*or* "a" "an" "the") @lcs-mod-thing @inst))

:MainRule
  ((?? (&eq @cat V) -> (do %S (&TH-order @this))
    ?? (&eq @cat N) -> (do %NP)
    -> (@inst))
```

Figure 11. A simple OxyL grammar.

operator to generate multiple paraphrases. In Rule `%S`, two alternative realizations are specified in the past tense. The first is an active voice realization and the second is a passive voice realization that hard-codes the passive auxiliaries *was* and *were*. In both cases, the morphological recast operator "`+-`" is used to generate the correct form of the verb (past and past-participle respectively).[9]

The rule `:MainRule` determines which phrase-level rule to apply by considering the category, i.e. part of speech, of the LCS-AMR instance. This is accomplished using the automatically defined function `@CAT`, which returns the value associated with the field `:CAT` in the LCS-AMR. The sequence "`?? X -> Y -> Z`" roughly corresponds to "if X then Y else Z". The rule `:MainRule` is applied recursively until no more LCS-AMRs exist.

The complete English Linearization grammar used in Lexogen is much larger and more complex than the one shown in Figure 11. It includes 14 different phrase-structure rules and four user-defined recast operations and it is about 300 lines of code long. In terms of maintainability, the length of the grammar should be compared with the Nitrogen grammar, which is over 1,500 lines. In this respect, the Oxygen grammar is more maintainable.

The simplicity of the linearization grammar is a central feature of the hybrid NLG approach which depends on the later statistical components to make the final decision based on n-gram information. The grammar specifies the simpler long-distance phrase linearization decisions which n-gram statistics cannot capture, while the n-gram statistics encode the complex local preferences that are harder to specify in a grammar. The quality of the English output produced is evaluated in Section 8.

In the next section, we discuss how to generate other languages using Lexogen.

6. Generation into Multiple Languages

Multilinguality or language independence refers to the degree of reusability of a system's components and resources for different languages. Not all NLG approaches are compatible with multilinguality as a goal, e.g., template-based generation. On the other hand, some researchers have invested a great deal of effort in creating environments for the development of NLG lexicons and systems with a multilingual spirit. One such environment is the Komet-Penman Multilingual Development Environment which provides an interface for developing large-scale sets of multilingual systemic-functional linguistic descriptions (Bateman, 1997; Bateman et al., 1999). A distinction can be made between "weak" multilinguality, where only components are shared, and "strong" multilinguality, where components and parts of the resources are shared (Rösner, 1994). Most multilingual NLG systems exhibit weak multilinguality (Kittredge et al., 1988; Okumura et al., 1991; Polguère, 1991; Vander Linden and Scott, 1995).

The design of Lexogen and its resources is highly compatible with multilinguality as a goal. The Lexogen algorithms are language-independent and the LCS lexicons are reusable for a new language without modification for both analysis and generation. While most of our effort has been spent on generation into English, in the context of Chinese–English translation, there has been some work using these components for generation into other languages. The retargeting of the system for other languages involves only the following language-specific resources:

- target-language LCS lexicon: a set of RLCS entries linking target-language words to LCSs, as described in Section 4;
- target-language linearization grammar, in OxyL (see Section 5.2.2);
- word n-gram statistics for the target language, for use by lattice extractor.

In addition, the following pre-processing steps are also needed for creating a generation system for the target language:

- hashing of target-language lexicon by "designated primitives", for on-line rapid retrieval (see Section 5.1.2);
- running oxyCompile on the linearization grammar to create an Oxygen Linearizer for the target language (see Section 5.2.2);
- creation of a target-language n-gram database, for use by the statistical lattice extractor.

An important feature of a translation approach using an interlingua such as LCS is that the same lexicon can be used for analysis and generation. Thus, we already have a major component for a Chinese generation system. Likewise, large LCS lexicons also exist for other languages such as Spanish and Arabic (Dorr, 1997a).

We have also created a linearization component for Spanish, using a simple OxyL Spanish linearization grammar. This grammar concentrates on argument word order relative to the verb. It utilizes a thematic hierarchy mapping that is very similar to that of English. We avoided dealing with complex Spanish morphology

by using the simple "near-future" construction (*va a* + INF). An example is shown in (18).[10]

(18) $alguien_{ag}$ *va a* *colocar* $algo_{th}$ *en* $algo_{goal}$
SOMEONE GOES TO PLACE SOMETHING IN SOMETHING
'Someone will place something in something'

In addition to the lack of a complete phrase structure for parts of speech other than verbs, the Spanish linearization grammar currently does not generate null-subject clauses or clitics. In principle, both phenomena can be handled with a recast rule that would fire after the thematic hierarchy recast: in null-subject cases, this rule would conjugate the verb and make the subject empty. In the case of clitics, it would add a clitic that matches the gender and number of the object.

A similar but even less sophisticated linearization grammar was created to generate Chinese. A preliminary study showed some promising results as far as thematic hierarchy mapping. However Chinese seems to require more complex linearization rules and post-lexical selection manipulations especially for obliques. Although Chinese thematic–syntactic linking is similar to Spanish and English, mapping from syntactic roles to surface positions is different. For example, the basic word order of Chinese is SVO, but the location of certain prepositional phrases is pre-verbal rather than post-verbal. This is not a "problem" for Lexogen. It is a problem as far as figuring out the governing linguistic rules since n-gram language models might not be able to differentiate among overgenerated alternatives if they included large constituents.

We have not yet built an n-gram extractor for other languages. Preliminary evaluation of Spanish generation is given in Section 8.4.

7. Comparison with Nitrogen and FERGUS

As discussed earlier, the closest "relatives" to Lexogen are Nitrogen and FERGUS, both hybrid NLG systems. The next two sections discuss these two systems. These are followed by a section comparing Lexogen to Nitrogen and FERGUS.

7.1. NITROGEN

Nitrogen was the first hybrid NLG system (Langkilde and Knight, 1998a,b,c). It was developed at the Information Science Institute at the University of Southern California as part of JAPANGLOSS (Knight et al., 1994, 1995), a Japanese–English newspaper MT system. Nitrogen breaks the generation process into two steps: overgeneration and statistical extraction. The first step is implemented symbolically using a simple generation grammar. The second step is implemented using a statistical uni/bigram model of the English language.

The input to Nitrogen is the AMR, as described in Section 5.1.4. The nodes of an AMR are concepts from the Sensus ontology (Knight and Luk, 1994).[11]

AMR labels are a mix of semantic (`:agent`, `:patient` and `:source`) and syntactic relations (`:subject`, `:object` and `:mod`).

The Nitrogen symbolic generator uses morphological and syntactic knowledge to transform the input AMR into a word lattice that efficiently compresses the space of possible paths. Morphological knowledge is implemented using a morphology derivation grammar that handles both derivations and inflections. Nitrogen overgenerates and depends on the statistical extractor to discard bad cases. Syntactic knowledge in Nitrogen is implemented using a grammar database that contains two types of transformation rules: linearization and recasting. Linearization rules transform an AMR into a word sequence or several ambiguous sequences. Recasting rules transform an AMR into another AMR by redefining the original relations or AMR structure. The Nitrogen grammar is organized around the semantic input rather than the syntax of English. The resulting word lattice is passed on to a statistical language model of English to rank and extract the most fluent paths. The language model estimates unigram and bigram probabilities from a large collection of 46 million words of the *Wall Street Journal*.

7.2. FERGUS

FERGUS (Flexible Empiricist/Rationalist Generation Using Syntax) extends the use of n-gram models with a tree-based statistical model and a traditional tree-based syntactic grammar (Alshawi et al., 2000; Bangalore et al., 2000; Bangalore and Rambow, 2000). Modeling syntax in FERGUS is done using an existing wide-coverage grammar of English, the XTAG grammar developed at University of Pennsylvania (XTAG-Group, 1999). XTAG is a Tree Adjoining Grammar (TAG) (Joshi, 1987) that has been extended to include lexicalization and unification-based feature structures (XTAG-Group, 1999). In TAG, the elementary structures are phrase tree structures that are composed using two operations: (a) substitution (which appends one tree at the frontier of another) and (b) adjunction (which inserts a tree into the middle of another). Morphosyntactic constraints such as subject–verb agreement and subcategorization can be specified within the tree structure associated with the lexical items. There is no distinction between the grammar and lexicon in lexicalized TAGs.

FERGUS is composed of three modules: the Tree Chooser, the Unraveler, and the Linear Precedence (LP) Chooser. The input to FERGUS is an unordered dependency tree labeled only with lexemes but no syntactic annotation. First the Tree Chooser uses a stochastic tree model to choose TAG trees for the nodes in the input tree. Then the Unraveler uses the XTAG grammar to produce a lattice of all possible linearizations that are compatible with the semi-specified input tree and XTAG. And finally, the LP Chooser selects the most likely traversal of this lattice given the language model. The tree model used by the Tree Chooser and the trigram model used by the Linear Precedence Chooser are both based on 1 million words from the *Wall Street Journal* corpus.

7.3. COMPARISON

Lexogen differs from Nitrogen and FERGUS on many dimensions. We focus here on input, hybridity and resource complexity.

First, the input to FERGUS is much shallower and closer to the target-language surface form than Lexogen's LCS or, for that matter, Nitrogen's AMRs. The AMR and LCS are both deeper representations, but each encompasses different types of knowledge. LCS contains information relevant to translation divergences and, thus, is more powerful in handling cross-linguistic mismatches. The AMR is more flexible in allowing both semantic and shallow syntactic roles to coexist in its representation.

Secondly, although the three systems are hybrids, they have a different balance of statistical and symbolic components. FERGUS relies on more powerful processing – both statistically and symbolically – than either of the other two approaches. FERGUS constrains its lattice of possible paths by symbolically handling agreement and other morphology issues using the XTAG component. Nitrogen, on the other hand, uses an explosively overgenerating simple grammar. Statistically, FERGUS uses both a stochastic tree model and a trigram language model whereas Nitrogen uses bigrams only. Lexogen falls between the two: symbolically, its LCS lexicons are more rich and semantically deeper than the XTAGs used in FERGUS but statistically, it uses only surface n-grams, essentially the same engine used in Nitrogen's statistical extractor.

Finally, Nitrogen's simplicity, which can be seen in its simple grammar and morphology is its biggest weakness due to: (a) interactions between simple rules in Nitrogen's grammar, which must be heavily controled in order produce reasonable output; (b) the simplicity of the grammar, which results in an explosive overgeneration requiring inefficient use of time and space; (c) dependencies between non-contiguous words, which cannot be captured by bigrams alone and for which the simplicity of the grammar provides no compensation.

Lexogen addresses the first and second points by using the LCS lexicon and a sophisticated Oxygen grammar that limits the overgeneration substantially. However, Lexogen would benefit from the techniques used in FERGUS for handling long-distance dependencies, which are addressed explicitly by the XTAG grammar at the symbolic level and by the Tree Chooser at the statistical level.

8. Evaluation

The evaluation of MT and NLG systems is more of an art than a science. Evaluation of generation systems is difficult, because the ultimate criterion is translation quality, which can, itself, be difficult to judge. Moreover, it can be hard to attribute specific deficits to the analysis phase, the lexical resources, or the generation system proper. A wide range of metrics and techniques have been developed over the last 50 years to assess "how good" a system is. Evaluation schemas vary in their focus from addressing the system's interface to system scalability, fidelity, space/time

complexity, etc. Another dimension of variation is human versus automatic evaluation. Fully automatic evaluation, a task that is AI-complete (i.e., encompassing all components of any system that would be deemed "intelligent"), is the ultimate goal in the field.[12]

In Church and Hovy (1993), three categories of MT evaluation metrics are described: system-based, text-based and cost-based. System-based metrics count internal resources: size of lexicon, number of grammar rules, etc. These metrics are easy to measure although they are not comparable across systems. In addition, their value is questionable since they are not necessarily related to utility.

Text-based metrics can be divided into sentence-based and comprehensibility-based. Sentence-based metrics examine the quality of single sentences out of context. These metrics include accuracy, fluency, coherence, etc. Typically, subjects evaluating sentences are given a description of the metric with examples and are asked to rate the sentences on an n-point scale. These scales range from 3-point to 100-point. Comprehensibility metrics measure the comprehension or informativeness of a complete text composed of several sentences. The subjects are typically given questionnaires related to the processed text.

Text-based metrics are much more related to utility than system-based metrics, but they are also much harder to measure. There are some automatic text-based evaluation metrics that measure the amount of post-editing needed for a sentence given a gold standard. These are variations on edit-distance, i.e., the number of deletions, additions or modifications measured by words or keystrokes per page or sentence. These techniques are not necessarily related to utility, however; Bangalore and Rambow (2000) showed that the smarter tree-based edit distance might actually correlate better to human judgment.

A more recent automatic evaluation system is IBM's Bleu (BiLingual Evaluation Understudy), which is quick, inexpensive, language-independent and, most importantly, highly correlating with human judgment (Papineni et al., 2002). Bleu uses n-gram precision variation, which compares the generated string and the reference string by taking the ratio of n-gram sequences in the generated string that appear in the reference string to the total number of n-gram sequences in the generated string. Bleu's major contribution is the use of multiple references to score translation candidates.

Cost-based metrics evaluate a system with respect to how much money/time it saves/costs per unit of text, say a page. These are secondary metrics since they depend on other metrics to evaluate how much post-/pre-processing is necessary for a commercially functional system.

Other evaluation type distinctions are black-box vs. glass-box and intrinsic vs. extrinsic evaluations. Black-box evaluations assess the behavior of the system as a whole, while glass-box evaluations are component-wise evaluations. Within a glass-box evaluation, one can distinguish between intrinsic and extrinsic evaluations: intrinsic evaluations focus on how a particular component is behaving in its

own terms, while extrinsic evaluations focus on how that component contributes to the performance of the whole system (Cole et al., 1997).

There are many aspects of a system that can be evaluated; some are harder than others. For example, to evaluate the efficiency of a system extrinsically, other aspects of the systems compared need to be constant. This is hard since efficiency and other features of the system, especially quality of output, are interconnected. To be able to make a reasonable comparative statement on efficiency of two systems, their outputs must be quite similar. This makes the evaluation much harder to create. In this paper we demonstrate the efficiency of our system based on modular improvements in efficiency with comparable outputs such as in using Oxygen linearization instead of Nitrogen's grammar. This and other previous intrinsic evaluations are discussed in the next section. Later, we evaluate the whole system on three extrinsic aspects: translation quality, which is what matters at the end of the day, coverage of linguistic phenomena, and retargetability to other languages.

8.1. INTRINSIC EVALUATIONS

Several intrinsic evaluations of specific components of Lexogen have been presented in previous papers. In Dorr et al. (1998) and also in more recent work (Habash and Dorr, 2001a), the thematic hierarchy implementation proved successful and the generation was demonstrated to be a diagnostic tool for fixing the lexicon, algorithmic errors, and inconsistencies in English and Spanish output.

Another major evaluation addressed the general performance of the Oxygen module (Habash, 2000). Oxygen was evaluated based on speed of performance, size of grammar, expressiveness of the grammar description language, reusability and readability/writability. The evaluation context was provided by comparing an Oxygen linearization grammar for English to two other completed implementations, one procedural (using Lisp) and one declarative (using Nitrogen linearization module). The three comparable linearization grammars were used to calculate speed and size. Overall, Oxygen had the highest number of advantages and its only disadvantage, speed, ranked second to the Lisp implementation.[13]

The generation component has also been used on a broader scale, generating thousands of simple sentences – at least one for each verb sense in the English LCS lexicon – creating sentence templates to be used in a cross-Language information retrieval system (Dorr et al., 2000).

In the rest of this section, we report on both quantitative and qualitative extrinsic evaluations of the system in several dimensions: translation quality, coverage and retargetability. Translation quality can be seen as a system-depth evaluation whereas coverage is a system-breadth evaluation. Retargetability focuses on the extendibility of the system to other languages.

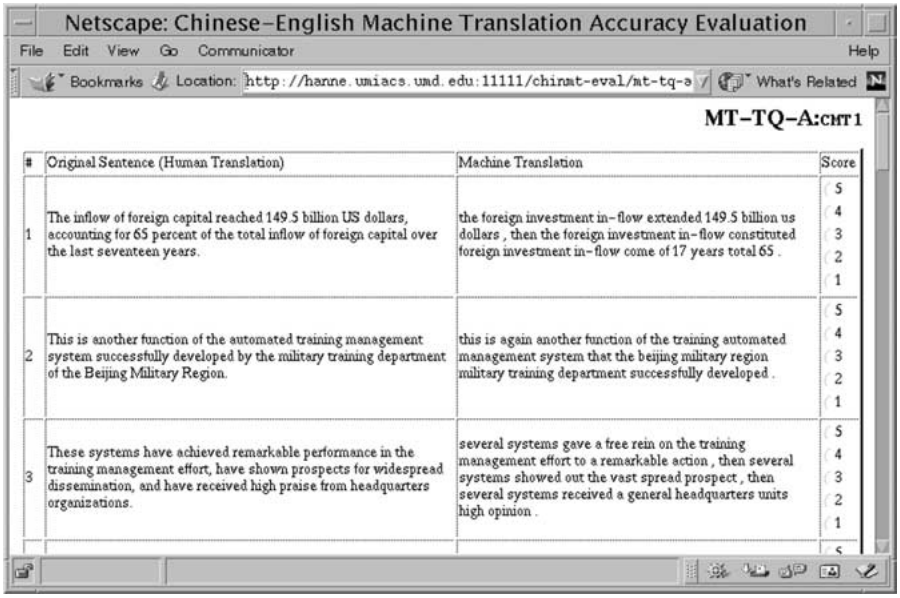

Figure 12. MT evaluation interface: Accuracy.

8.2. TRANSLATION QUALITY EVALUATION

The Lexogen generation system has been used as part of ChinMT, a Chinese–English Translation system focusing on a corpus of ten newspaper articles from *Xinhua* (Chinese People's Daily). The articles included 80 sentences that our translation system was able to parse, compose into LCS interlingua, and generate into English successfully. Although the number of sentences is small, some of them are quite complex, and represent a cross-section of the types of complex phenomena handled in a large-scale MT effort. To measure the translation quality of the system, we performed two human evaluations: one for accuracy (fidelity) and one for fluency (intelligibility). Both tests used a set of 25 sentences randomly selected from the 80 original Chinese sentences that completely pass our translation system. For comparison purposes, we also used a commercial Chinese–English translation system to translate these sentences: *Chinese–English Systran 3.0 Professional edition*. Thus, we have both absolute quality metrics and a comparison to state-of-the-art MT.

The test suite is a 2×2 grid: (accuracy, fluency) $\times$ (ChinMT, Systran). The total number of subjects is 80, all of whom are native speakers of English. Each subject participated in only one of the four possible evaluations (e.g., ChinMT accuracy or Systran fluency) for all 25 sentences.[14] The evaluation was performed online using a web interface (see Figure 12).

Table III. Accuracy criteria.

5	Contents of original sentence conveyed (might need minor corrections)
4	Contents of original sentence conveyed *but* errors in word order
3	Contents of original sentence generally conveyed *but* errors in relationship between phrases, tense, singular/plural, etc.
2	Contents of original sentence not adequately conveyed, portions of original sentence incorrectly translated, missing modifiers
1	Contents of original sentence not conveyed, missing verbs, subjects, objects, phrases or clauses

8.2.1. *Accuracy Evaluation*

This evaluation measures the accuracy or fidelity of the translation system, i.e., how well a system preserves the meaning of the original text whether the target language is fluent or not. The subjects were given 25 pairs of sentences. Each pair consists of a human translation of the Chinese original and a machine-translated version. Subjects were asked to rate the translation accuracy on a 5-point scale (see Table III).[15]

A score of 5 is given where the content of the original sentence is fully conveyed (might need minor corrections). A score of 1 is given where the content of the original sentence is not conveyed at all. An earlier pilot study indicated that subjects had a hard time with descriptions of the scale and preferred examples instead. Thus subjects were provided with a table containing two manually constructed examples per score to illustrate the idea behind the scoring scheme (see Table IV). Figure 12 displays a screen capture of the web interface showing the first three pairs of sentences in an accuracy evaluation form.

8.2.2. *Fluency Evaluation*

In the fluency evaluation, the subjects were given 25 machine-translated sentences. The purpose of this evaluation is to measure the fluency (or intelligibility) of the translation system. Subjects were asked to rate the fluency of machine-translated sentences on a 5-point scale again loosely based on Nagao's (1989) intelligibility scale metric. The scale ranges from 5 (clear meaning, fluent sentence) to 1 (meaning absolutely unclear, sentence not fluent). Table V details the criteria used in measuring fluency.

Because fluency and intelligibility are both important for MT evaluation, we chose a composed metric that includes both. Table VI describes the examples given to the subjects to help them understand and use the scale. The actual evaluation input looked like the examples provided in Figure 12 without the first column.

Table IV. Accuracy scale examples.

Original Sentence (Human Translation) *The United States unilaterally reduced China's textile export quotas.*	
Machine Translation	Score
– United States reduced China's textile export quota unilaterally.	5
– United States reduced China textile export quota unilaterally.	5
– United States cut China quota export textile unilaterally down.	4
– United States China quota export textile cuts down unilaterally down.	4
– United States down to slash of a export textile Chinese the quotas.	3
– Some states united slash down reducingly down China textile of export ration.	3
– Beautiful folk slashed porcelain export on own way.	2
– State reduce quota.	2
– It cut.	1
– China.	1

Table V. Fluency criteria.

5	Clear meaning, good grammar, terminology and sentence structure
4	Clear meaning *but* bad grammar, bad terminology or bad sentence structure
3	Meaning graspable *but* ambiguities due to bad grammar, bad terminology or bad sentence structure
2	Meaning unclear *but* inferable
1	Meaning absolutely unclear

Table VI. Fluency scale examples.

Machine Translation	Score
– The United States unilaterally reduced China's textile export quotas.	5
– The United States unilaterally reduced China textile export quotas.	5
– United States cutted China export textile ration lonely.	4
– United States reduce down China quota textile export.	4
– United States reduce an quotas export textiling of the porcelain for the only busy a decision.	3
– A chinese ration united states cut it down.	3
– States united unilateral cut an China textile speaks ration downwardly down.	2
– Cause states go quotas to reduced.	2
– Beautiful folk remedy partage China exportation filament on own shaving.	1
– Alone cut it up rations alone.	1

Table VII. Chinese–English translation quality results.

	LCS-based MT	Systran
Accuracy	3.08	3.01
Fluency	3.15	3.12

8.2.3. *Translation Quality Evaluation Results*

The results of the evaluation are presented in Table VII. The number in each cell represents the average score given by all subjects on all sentences for each evaluation. ChinMT did slightly better than Systran but the difference is statistically insignificant. Overall, the scores given show an average performance for both systems, glossed as follows: for accuracy, contents of original sentence are generally conveyed but there are errors in the relationship between constituents (cf. Table III) and for fluency, the meaning is graspable but ambiguities exist (cf. Table V).

The ChinMT system was able to perform as well as a commercial system that took many person-years to develop, a promising result, given that our development time for Chinese–English MT was significantly shorter. The Systran Chinese–English MT system is the result of an estimated 20 person-years of work.[16] It utilizes a large lexicon of 150,000 root stems, 6,000 expressions, 1,000–2,000 Cantonese terms, 2,500 names, a 300,000-word safety-net lexicon (CETA dictionary) and an optional 2,000 military terms. With this coverage, the system's strength is in military, computer science, and electronics domains.

ChinMT was developed over six person-years. The English LCS lexicon includes about 12,000 entries, of which 9,500 are verbs and 900 are prepositions. The remaining 1,200 are nouns and adjectives, which may be dynamically generated based on specific domain needs.

ChinMT, as a modular interlingua-based system, has resources that are readily extensible for use with other languages for both analysis and generation. A case in point is a previous project for Language Tutoring using LCS resources was retargeted from Arabic to Spanish in one sixth of the time it took to build the original project (Dorr, 1997b).

8.2.4. *Analysis of Translation Quality Results*

For the most part, the Nitrogen strategy of overgenerating translation hypotheses coupled with selection according to bigram likelihoods works very well (Langkilde and Knight, 1998b). There are some difficulties that can be seen as responsible for the *average* scores received. One major issue is that, especially with the bigram language model's bias for shorter sentences, fluency is given preference over translation accuracy. Thus, if there is some material that is considered optional (e.g., by

the decomposition process) and there are lattice entries both with and without this information, the extractor will tend to pick the path without this information. While this technique is also very successful at picking out more fluent, terse formulations (19), further work is needed to assess the right ratio of terseness to informativeness. Also, bigrams are obviously inadequate for capturing long-distance dependencies, and so, if things like agreement are not carefully controled in the symbolic component, they will be incorrect in some cases.

(19) a. John went to the bank *vs.* *John went to at the bank

b. convincing proof *vs.* proof having convincingness

8.3. COVERAGE EVALUATION

For this evaluation, a test corpus of 453 simple CLCS representations corresponding to all LVD classes (see Section 4) was constructed semi-automatically.[17] The size of the test corpus guarantees large-scale coverage over verb behavior and thematic role combinations, which is exhaustive for our purpose. The CLCS representations were constructed by randomly selecting an LCS verb entry from each English verb class and filling all its argument positions with simple noun phrases (e.g. *something*$_{th}$, *someone*$_{ag}$, etc.) or simple subordinate clauses (e.g. *to do something*$_{prop}$, *to be someproperty*$_{mod\text{-}prop}$, etc.) Table VIII shows some sample English sentences corresponding to the CLCS representations in the test corpus.

Table VIII. CLCS test corpus examples.

Class	Example
2	Someone$_{ag}$ wanted something$_{th}$ (to do something$_{th}$)$_{prop}$
10.5	Someone$_{ag}$ stole something$_{th}$ from something$_{src}$ for something$_{ben}$
22.1.C	Someone$_{ag}$ mixed something$_{th}$ into something$_{goal}$
29.1.B	Someone$_{th}$ considered something$_{perc}$ (to be someproperty$_{pred}$)$_{mod\text{-}pred}$
45.2.A	Someone$_{ag}$ folded something$_{th}$ with something$_{inst}$
55.1.C	Someone$_{th}$ continued (to do something$_{th}$)$_{prop}$

For this evaluation, statistical extraction was disabled to evaluate the whole range of possible outputs generated by the system. For example, each of the two subclasses defining the dative alternations for the verb *send* are expected to generate both alternations (20). Out of 453 input CLCS representations, 25 failed the lexical-selection process due to misformatted lexicon entries.

(20) a. John sent a book to Paul.

b. John sent Paul a book.

Table IX. Coverage evaluation.

N = 428	Argument error	Preposition error	Word-order error
Class-based	6 (1%)	53 (12%)	5 (1%)
Verb-based	1%	9%	3%

In the remaining 428 cases, the lexical selection process appropriately generated multiple sentences for each CLCS. All of these correctly corresponded to various related alternations of the main verb. However, there were also cases of overgeneration resulting from preposition underspecification, which is inconsequential to our evaluation (e.g. *go* (*to*, *toward*, *towards*, *to at*) *somewhere*). The average number of sentences generated per class was four.

8.3.1. *Coverage Evaluation Criteria*

The results of generation were passed to a speaker of English who was asked to mark sentences as being acceptable or not acceptable on three criteria: (a) argument generation, (b) prepositional-phrase generation, and (c) word order. Acceptable argument generation is defined as the generation of all arguments of the verb whether subject, object, or oblique. Acceptable prepositional-phrase generation is defined as the generation of good preposition choices such as goal prepositions versus source prepositions with an oblique goal and the generation of a prepositional object. Finally, acceptable word order is word order that reflects the correct relation of the arguments to the verb.

8.3.2. *Coverage Evaluation Results*

Table IX displays the results of this evaluation. The first row represents the number and ratio of classes that generated no correct output for each error criterion. Some classes generated both correct and incorrect outputs. These are counted as correct with the assumption that given a good statistical extractor, the correct answer would rank highest. The second row is an estimate of the percentage of unsuccessful generation of verb senses, where the raw class results are weighted by the number of verbs in each class. On average each class contains 21 verbs, but since some classes have more verbs in them than others, this second value seems a more appropriate measure to evaluate coverage over the full lexicon (estimating actual verbs covered rather than verb classes). Another useful metric might be to normalize based on the probability of occurrence of verbs, giving more weight to frequently occurring verbs. But this is a much more complicated task because it requires a corpus that tags verb senses with appropriate LCS structures.

The results of this evaluation are quite encouraging in that they show a high percentage of coverage over the LCS lexicon. Argument errors and word-order

errors were due to incorrect lexical entries. For example, in the case of word-order errors, specific realization information such as `:EXT` was missing from some entries. This problem appeared in three subclasses of class 41.3.1 (Simple Verbs of Dressing: *don*, *doff* and *wear*). In our lexicon, *clothes*, the object for all three verbs, is considered the theme, and the subject of the sentence is the goal, source and location respectively. Fixing these cases is a matter of adding the appropriate piece of information in the lexicon. Preposition errors are more severe in that complete entries for some prepositions were not found in the lexicon. These errors will be fixed once the proper entries have been added. The generation system has thus been quite helpful as a diagnostic tool for determining errors and inconsistencies in the lexicon.

8.4. RETARGETABILITY

Finally, we examine Lexogen's language independence. For this evaluation task we used as input the same corpus of simple CLCS entries developed for the coverage evaluation presented in the previous section, but we replaced the English components with the Spanish ones as described in Section 6. For the purposes of this evaluation, statistical extraction was disabled because we do not have a Nitrogen bigram model for Spanish and because we wanted to examine the range of alternations produced.

The results of the generation were passed to a speaker of Spanish to evaluate in a similar manner to the coverage evaluation. One extra criterion in this evaluation is a check on sense generation correctness, i.e., whether this Spanish verb is a proper translation of the English verb given the argument structure presented in the verb class.

As in the case of the English generation results presented in the previous section, some of the Spanish sentences failed to produce any acceptable decompositions due to problems with lexicon entries. However, there were many more sentences that were produced which should not have been generated in Spanish. In theory, the lexical selection process limits the number of choices using the LCS entry of the Spanish verbs. But that process is only as good as the lexicon entries. In cases where a bad sense is generated, the sentence involved is dropped from the evaluation. Most failures in Spanish generation are due to missing verb entries (29% of all input classes). Erroneous lexicon entries were responsible for another 10% of generation failures and an additional 5% of classes were dropped out of the evaluation because there was no correct sense output. As a result only 254 out of 453 classes (56%) have been evaluated on argument, preposition and word-order correctness.

Table X displays the results of this evaluation. The first row represents the number and ratio of classes that generated no correct output for each error criterion. The second row represents the same ratios including class verb count as weights.

Table X. Retargetability evaluation.

N = 254	Argument error	Preposition error	Word-order error
Class-based	15 (6%)	85 (33%)	4 (2%)
Verb-based	10%	44%	0%

The Spanish output is not as clean as the English output. It has more overgeneration, more failures, and a higher error rate (except for word-order errors). Argument errors are due to lexicon entries that were incorrect or missing. Most of preposition errors were due to incorrect overgeneration resulting from extra incorrect entries which were added to the lexicon automatically and were not manually checked.

A recent analysis of the Spanish lexicon indicates that 300 out of 453 semantic verb classes (about 65%) were correctly ported automatically from English to Spanish. (See Dorr (1997a) for more details of the porting process.) The remaining 35% were rapidly hand-corrected by one native speaker in less than a month. More recently, the same approach has been taken for Chinese (Dorr et al., forthcoming), with a similarly successful outcome.

However, the focus of this evaluation was not on the quality or coverage of Spanish in our system. It was on the ease of retargetability of the system to another language. And given this criterion, this evaluation is quite positive since the amount of work that was needed was minimal. The Spanish lexicon already existed for analysis purposes and the OxyL grammar was created in a short period of time (two days). Of course improving the quality of the system will need more work on both frontiers, the lexicon and the linearization grammar. There is also a role to play in statistical extraction of the best generated sentence, especially for cases of overgeneration that included both good and bad results.

9. Conclusions and Future Work

We have presented Lexogen, a system for hybrid NLG from LCSs. Lexogen is used as part of a larger MT effort, and we have presented an evaluation of some key components of the results. The system has been used both to generate very long, complex, multiply ambiguous sentences (outputs of Chinese–English translations), as well as thousands of simple sentence templates (spanning the whole of the English verb and preposition lexicons). Evaluation of the quality and correctness of both modes has been carried out, showing comparable translation quality with a commercial translation system which took longer to build. The generation system can also be straightforwardly extended to other languages, given appropriate target-language specific resources (lexicon and grammar), and this has been demonstrated and evaluated for Spanish.

Many Lexogen modules are designed with efficiency in mind. Examples include (a) converting text input format into a graph for efficient access to components, (b) accessing anchor points to increase the efficiency of the matching algorithm during alignment/decomposition, (c) compiling linearization grammar rules into a programming language for efficient performance, and (d) transforming AMRs used to encode the input sentence into a word lattice that efficiently compresses the space of possible paths.[18]

In addition to its utility for generating target-language sentences, the generation system provides a crucial step in the development cycle for analysis and lexicon resources. Changes to a lexicon, both extensions and corrections – which might be done either manually or automatically – can be evaluated based on how they will affect generation of sentences into that language. This has been a valuable diagnostic tool for discovering both specific errors and lacunae in lexicon coverage.

The biggest remaining step is a more careful evaluation of different subsystems and preference strategies to process more efficiently very ambiguous and complex inputs, without substantially sacrificing translation quality. Also a current research topic is how to combine other metrics coming from various points in the generation process with the bigram statistics, to result in better overall outputs.

Acknowledgements

This work has been supported, in part, by a DOD Contract and NSF PFF/PECASE Award IRI-962910. The second author is also supported by NSF CNRS INT-9314583, DARPA/ITO Contract N66001-97-C-8540, Alfred P. Sloan Research Fellowship Award BR3336, and NSF CISE Research Infrastructure Award EIA0130422. The third author has also been supported during the writing of this paper by the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

We would like to thank Clara Cabezas and Yi Ching Su from the Linguistics Department at the University of Maryland and Gina Levow from the University of Chicago for their help evaluating the system. We would also like to thank members of the Computational Linguistics and Information Processing Lab (CLIP) at the University of Maryland for helpful conversations, particularly David Clark, Scott Thomas, Philip Resnik, and Amy Weinberg. Also, we would like to thank Mari Olsen from Microsoft for more helpful conversations. Carol Van-Ess Dykema helped with the word-ordering scheme for noun modifiers. And finally, we would like to thank Kevin Knight and Irene Langkilde for making the Nitrogen system available and help with understanding the Nitrogen grammar formalism.

The editor would like to thank Paul Bennett for help with the Chinese examples.

Notes

[1] The most recent version of Nitrogen is called Halogen (Langkilde-Geary, 2002).

[2] Fields are specified for all primitives except `CAUSE`, which cuts across all possible fields.

[3] Such relations are specified declaratively, in a Lisp structure with two-way parent–child links.

[4] Levin's original database contained 192 classes, numbering between 9.1 and 57; our refined version contains 492, with more specific identifiers. For example, Levin's class 51.3.2 "Run Verbs" has been subdivided into 51.3.2.a.i "Run verbs–Theme Only", 51.3.2.a.ii "Run verbs–Theme/Goal", 51.3.2.b.i "Run verbs–Agent and Theme", 51.3.2.b.ii "Run verbs–Agent/Theme/Goal", and 51.3.2.c "Run verbs–Locational".

[5] Certain components of the LCS occur in multiple places (e.g., `(thing 2)`) to facilitate compatibility checking during the process of semantic composition.

[6] See Dorr and Olsen (1996) and Olsen et al. (2001) for a detailed study of the use of telicity for tense and aspect realization.

[7] For an excellent overview and a comparison of different thematic hierarchies see Levin and Rappaport Hovav (1996).

[8] There are other generation systems with a larger set of semantic/thematic roles, e.g., the Nigel generation system's 100+ roles (Mann and Matthiessen, 1985). However, no thematic hierarchy is used in generation in such systems.

[9] Oxygen does not provide its own morphological generation component since that is a language-specific module. Oxygen and previous realizers in Lexogen have used Nitrogen's English morphological generation component.

[10] Our choice of present tense over future is consistent with findings of researchers who have investigated temporal preferences based on a large-scale corpus analysis (Dorr and Gaasterland, forthcoming).

[11] This is a knowledge base of 70,000 nodes derived from several sources such as WordNet, Longman's Dictionary and the Penman upper model.

[12] For excellent surveys of MT evaluation metrics and techniques, see Hovy (1999) and ISLE (2000).

[13] A more efficient implementation of Oxygen is currently used with a larger grammar than the one evaluated in Habash (2000). A Lisp implementation of the same larger grammar is still faster than the newer Oxygen implementation. However, the gap in speed between the Lisp and Oxygen implementations shrunk from Oxygen being 24 times slower than Lisp in Habash (2000) to only 1.5 times.

[14] To avoid order bias that can result from degradation in subject performance over time, each grid cell has two versions with different sentence display: (1 to 25) and (13 to 25, 1 to 12).

[15] Loosely based on Nagao's (1989) 7-point scale for fidelity.

[16] This estimate is computed based on information provided through personal communication with Mr Dale Bostad from NAIC (National Air Intelligence Center), the agency that sponsored the development of this product.

[17] Currently, the number of classes in LVD is 492. But at the time of conducting this evaluation, there were only 453 classes.

[18] A recent technique for encoding AMRs into forests improves on lattice compression even more (Langkilde-Geary, 2002).

References

Alshawi, Hiyan, Srinivas Bangalore, and Shona Douglas: 2000, 'Learning Dependency Translation Models as Collections of Finite-State Head Transducers', *Computational Linguistics* **26**, 45–60.

Alsina, Alex and Sam A. Mchombo: 1993, 'Object Asymmetries and the Chichewa Applicative Construction', in Sam A. Mchombo (ed.): *Aspects of Automated Natural Language Generation*, Stanford, CA: CSLI Publications, pp. 1–46.

Baker, Carl Lee: 1989, *English Syntax*, Cambridge, MA: The MIT Press.

Bangalore, Srinivas and Owen Rambow: 2000, 'Corpus-Based Lexical Choice in Natural Language Generation', in *38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, pp. 464–471.

Bangalore, S., O. Rambow, and S. Whittaker: 2000, 'Evaluation Metrics for Generation', in *Proceedings of the 1st International Conference on Natural Language Generation (INLG 2000)*, Mitzpe Ramon, Israel, pp. 1–8.

Bateman, John A.: 1997, 'Enabling technology for multilingual natural language generation: the KPML development environment', *Natural Language Engineering* **3**, 15–55.

Bateman, J., C. Matthiessen, and L. Zeng: 1999, 'Multilingual natural language generation for multilingual software: a functional linguistic approach', *Applied Artificial Intelligence* **13**, 607–639.

Bresnan, J. and J. Kanerva: 1989, 'Locative Inversion in Chichewa: A Case Study of Factorization in Grammar', *Linguistic Inquiry* **20**, 1–50.

Carrier-Duncan, J.: 1985, 'Linking of Thematic Roles in Derivational Word Formation', *Linguistic Inquiry* **16**, 1–34.

Chandioux, John: 1989, 'MÉTÉO: 100 Million Words Later', in D. L. Hammond (ed.), *American Translators Association Conference 1989: Coming of Age*, Medford, NJ: Learned Information, pp. 449–453.

Charniak, E.: 2000, 'Statistical Techniques in Natural Language Processing', in *The MIT Encyclopedia of the Cognitive Sciences*, Cambridge: MIT Press.

Church, Kenneth W. and Eduard H. Hovy: 1993, 'Good Applications for Crummy Machine Translation', *Machine Translation* **8**, 239–258.

Cole, R., J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue: 1997, *Survey of the State of the Art in Human Language Technology*, Cambridge University Press, Cambridge, UK.

Dorr, Bonnie Jean: 1993a, 'Interlingual Machine Translation: A Parameterized Approach', *Artificial Intelligence* **63**, 429–492.

Dorr, Bonnie Jean: 1993b, *Machine Translation: A View from the Lexicon*, Cambridge, MA: The MIT Press.

Dorr, Bonnie J.: 1994, 'Machine Translation Divergences: A Formal Description and Proposed Solution', *Computational Linguistics* **20**, 597–633.

Dorr, Bonnie J.: 1997a, 'Large-Scale Acquisition of LCS-Based Lexicons for Foreign Language Tutoring', in *Fifth Conference on Applied Natural Language Processing*, Washington, DC, pp. 139–146.

Dorr, Bonnie J.: 1997b, 'Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Machine Translation', *Machine Translation* **12**, 271–322.

Dorr, B. J.: 2001, 'LCS Verb Database', Technical Report Online Software Database, University of Maryland, College Park, MD.
`http://www.umiacs.umd.edu/~bonnie/LCS_Database_Docmentation.html`.

Dorr, B. J. and T. Gaasterland: 2002, 'Constraints on the Generation of Tense, Aspect, and Connecting Words from Temporal Expressions', Technical Report CS-TR-4391, UMIACS-TR-2002-71, LAMP-TR-091. University of Maryland, College Park, MD, USA.

Dorr, Bonnie J., Joseph Garman, and Amy Weinberg: 1995, 'From Syntactic Encodings to Thematic Roles: Building Lexical Entries for Interlingual MT', *Machine Translation* **9**, 221–250.

Dorr, Bonnie, Nizar Habash, and David Traum: 1998, 'A Thematic Hierarchy for Efficient Generation from Lexical-Conceptal Structure, in David Farwell, Laurie Gerber and Eduard Hovy (eds), *Machine Translation and the Information Soup: Third Conference of the Association for Machine Translation in the Americas, AMTA'98*, Berlin: Springer, pp. 333–343.

Dorr, Bonnie J., Gina-Anne Levow, and Dekang Lin: 2000, 'Building a Chinese-English Mapping Between Verb Concepts for Multilingual Applications', In White (2000), pp. 1–12.

Dorr, Bonnie J., Gina-Anne Levow, and Dekang Lin: forthcoming, 'Construction of a Chinese–English Verb Lexicon for Embedded Machine Translation in Cross-Language Information Retrieval', to appear in *Machine Translation* (Special Issue on Embedded MT).

Dorr, Bonnie J. and Mari Broman Olsen: 1996, 'Multilingual Generation: The Role of Telicity in Lexical Choice and Syntactic Realization', *Machine Translation* **11**, 37–74.

Elhadad, Michael, Kathleen McKeown, and J. Robin: 1997, 'Floating Constraints in Lexical Choice', *Computational Linguistics* **23**, 195–240.

Elhadad, Michael and Jacques Robin: 1992, 'Controlling Content Realization with Functional Unification Grammars', in Robert Dale, Eduard Hovy, Dietmar Rösner, and Oliviero Stock (eds), *Aspects of Automated Natural Language Generation: The 6th International Workshop on Natural Language Generation*, Berlin: Springer, pp. 89–104.

Fellbaum, C.: 1998, *WordNet: An Electronic Lexical Database*, Cambridge, MA: The MIT Press.

Giorgi, A.: 1984, 'Toward a Theory of Long Distance Anaphors: A GB Approach', *The Linguistic Review* **3**, 307–361.

Grimshaw, J. and A. Mester: 1988, 'Light Verbs and Theta-Marking', *Linguistic Inquiry* **19**, 205–232.

Habash, Nizar: 2000, 'Oxygen: A Language Independent Linearization Engine', in White (2000), pp. 68–79.

Habash, Nizar: 2001, A Reference Manual to the Linearization Engine oxyGen version 1.6', Technical report CS-TR-4295, University of Maryland, College Park, MD.

Habash, Nizar and Bonnie Dorr: 2001a, 'Large Scale Language Independent Generation Using Thematic Hierarchies', in *MT Summit VIII: Machine Translation in the Information Age*, Santiago de Compostela, Spain, pp. 139–144.

Habash, Nizar and Bonnie J. Dorr: 2001b, 'Large Scale Language Independent Generation: Using Thematic Hierarchies', Technical report LAMP-TR-075, CS-TR-4280, UMIACS-TR-2001-59, University of Maryland, College Park, MD.

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

Hirst, Graeme: 1987, *Semantic Interpretation and the Resolution of Ambiguity*, New York: Cambridge University Press.

Hovy, Eduard: 1988, 'Generating Natural Language under Pragmatic Constraints', Ph.D. thesis, Yale University.

Hovy, Eduard: 1999, 'Toward Finely Differentiated Evaluation Metrics for Machine Translation', in *Proceedings of the EAGLES Workshop on Standards and Evaluation*, Pisa, Italy, pp. 127–133.

International Standards for Language Engineering (ISLE): 2000, 'The ISLE Classification of Machine Translation Evaluations', `http://www.isi.edu/natural-language/mteval/`, [accessed 9/8/2003].

Jackendoff, Ray S.: 1972, *Semantic Interpretation in Generative Grammar*, Cambridge, MA: The MIT Press.

Jackendoff, Ray S.: 1983, *Semantics and Cognition*, Cambridge, MA: The MIT Press.

Jackendoff, Ray S.: 1990, *Semantic Structures*, Cambridge, MA: The MIT Press.

Jackendoff, Ray S.: 1996, 'The Proper Treatment of Measuring Out, Telicity, and Perhaps Even Quantification in English', *Natural Language and Linguistic Theory* **14**, 305–354.

Joshi, A. K.: 1987, 'An Introduction to Tree Adjoining Grammars', in A. Manaster-Ramer (ed.): *Mathematics of Language*, John Benjamins, Amsterdam, pp. 87–115

Jurafsky, Daniel and James H. Martin: 2000, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Upper Saddle River, New Jersey: Prentice Hall.

Kay, Martin: 1979, 'Functional Grammar', in *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*, Berkeley, CA, pp. 142–158.

Kiparsky, P.: 1985, 'Morphology and Grammatical Relations', unpublished ms., Stanford University.

Kittredge, Richard, Lidija Iordanskaja, and Alain Polguère: 1988, 'Multi-Lingual Text Generation and the Meaning-Text Theory', in *Second International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Pittsburgh, Pennsylvania.

Knight, Kevin, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K. Luk, Akitoshi Okumura, Richard Whitney, and Kenji Yamada: 1994, 'Integrating Knowledge Bases and Statistics in MT', in *Technology Partnerships for Crossing the Language Barrier: Proceedings of the First Conference of the Association for Machine Translation in the Americas*, Columbia, Maryland, pp. 134–141.

Knight, Kevin, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard H. Hovy, Masayo Iida, Steve K. Luk, Richard Whitney, and Kenji Yamada: 1995, 'Filling Knowledge Gaps in a Broad-Coverage Machine Translation System', in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, pp. 1390–1397.

Knight, Kevin and Vasileios Hatzivassiloglou: 1995, 'Two-Level, Many-Paths Generation', in *33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, pp. 252–260.

Knight, Kevin and Steve K. Luk: 1994, 'Building a Large Knowledge Base for Machine Translation', in *Proceedings of the 12th National Conference on Artificial Intelligence*, Seattle, WA, pp. 773–778.

Langkilde, Irene and Kevin Knight: 1998a, 'Generating Word Lattices from Abstract Meaning Representation', Technical report, Information Sciences Institute, University of Southern California, Marina del Rey, CA.

Langkilde, Irene and Kevin Knight: 1998b, 'Generation that Exploits Corpus-Based Statistical Knowledge', in *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Quebec, pp. 704–710.

Langkilde, Irene and Kevin Knight: 1998c, 'The Practical Value of n-Grams in Generation', in *Proceedings of the 9th International Natural Language Workshop (INLG'98)*, Niagara-on-the-Lake, Ontario, pp. 248–255.

Langkilde-Geary, Irene: 2002, 'An Empirical Verification of Coverage and Correctness for a General-Purpose Sentence Generator', in *Proceedings of International Natural Language Generation Conference (INLG'02)*, Harriman, NY, pp. 17–24

Larson, R.: 1988, 'On the Double Object Construction', *Linguistic Inquiry* **19**, 335–391.

Levin, Beth: 1993, *English Verb Classes and Alternations: A Preliminary Investigation*, Chicago, IL: University of Chicago Press.

Levin, B. and M. Rappaport Hovav: 1996, 'From Lexical Semantics to Argument Realization', Technical report, Northwestern University, Evanston IL, and Bar Ilan University, Ramat Gan, Israel. ftp://ftp.ling.nwu.edu/pub/beth/borer96.ps.

Malouf, Robert: 2000, 'The Order of Prenominal Adjectives in Natural Language Generation', in *38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, pp. 85–92.

Mann, William C. and Christian Matthiessen: 1985, 'Demonstration of the Nigel Text Generation Computer Program', in James D. Benson and William S. Greaves (eds), *Systemic Perspectives on Discourse, Volume 1*, Norwood, NJ: Ablex, pp. 50–83.

Marcu, D., L. Carlson, and M. Watanabe: 2000, 'An Empirical Study in Multilingual Natural Language Generation: What Should a Text Planner Do?', in *First International Natural Language Generation Conference (INLG'2000)*, Mitzpe Ramon, Israel, pp. 17–23.

Miller, George A. and Christiane Fellbaum: 1991, 'Semantic Networks of English', in Beth Levin and Steven Pinker (eds), *Lexical and Conceptual Semantics*, Amsterdam: Elsevier, pp. 197–229.

Nagao, Makoto: 1989, 'Two Years After the MT SUMMIT', in *MT Summit II: Final Programme, Exhibition, Papers*, Munich, Germany, pp. 100–105.

Nishgauchi, T.: 1984, 'Control and the Thematic Domain', *Language* **60**, 215–260.

Okumura, Akitoshi, Kazunori Muraki, and Susumu Akamine: 1991, 'Multi-lingual Sentence Generation from the PIVOT Interlingua', in *Machine Translation Summit III Proceedings*, Wash-

ington, D.C., pp. 67–71. Repr. in: Sergei Nirenburg (ed.) *Progress in Machine Translation*, Amsterdam/Tokyo (1993): IOS Press/Ohmsha, pp. 119–125.

Olsen, Mari Broman: 1997, *A Semantic and Pragmatic Model of Lexical and Grammatical Aspect*, New York: Garland Press.

Olsen, Mari, David Traum, Carol Van Ess-Dykema, and Amy Weinberg: 2001, 'Implicit Cues for Explicit Generation: Using Telicity as a Cue for Tense Structure in a Chinese to English MT System', in *MT Summit VIII: Machine Translation in the Information Age, Proceedings*, Santiago de Compostela, Spain, pp. 259–264.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu: 2002, 'Bleu: A Method for Automatic Evaluation of Machine Translation', in *40th Annual Meeting of the Association of Computational Linguistics*, Philadelphia, PA, pp. 311–318.

Paris, C., K. Vander Linden, M. Fischer, A. Hartley, L. Pemberton, R. Power, and D. Scott: 1995, 'A Support Tool for Writing Multilingual Instructions', in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montréal, Canada, pp. 1398–1404.

Penman: 1989, 'The Penman Reference Manual', ISI, University of Southern California, Marina del Rey, CA.

Polguère, A.: 1991, 'Everything Has Not Been Said about Interlinguae: The Case of Multilingual Text Generation Systems', in *Proceedings of Natural Language Processing Pacific Rim Symposium (NLPRS '91)*, Singapore, pp. 314–320.

Quirk, Randolf, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik: 1985, *A Comprehensive Grammar of the English Language*, London: Longman.

Ratnaparkhi, Adwait: 2000, 'Trainable Methods for Surface Natural Language Generation', in *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA, pp. 194–201.

Reiter, Ehud: 1995, 'NLG vs Templates', in *Proceedings of the Fifth European Workshop on Natural Language Generation*, Leiden, The Netherlands, pp. 95–106.

Rosetta, M. T.: 1994, *Compositional Translation*, Dordrecht: Kluwer.

Rösner, Dietmar: 1994, *Automatische Generierung von mehrsprachigen Instruktionstexten aus einer Wissensbasis* [Automatic knowledge-based generation of multilingual instruction texts]. Habilitationsschrift, Fakultät für Informatik, Universität Stuttgart.

Rösner, Dietmar and Manfred Stede: 1994, 'TECHDOC: Multilingual Generation of Online and Offline Instructional Text', in *4th Conference on Applied Natural Language Processing*, Stuttgart, Germany, pp. 209–210.

Shaw, James and Vasileios Hatzivassiloglou: 1999, 'Ordering Among Premodifiers', in *37th Annual Meeting of the Association for Computational Linguistics*, College Park, MD, pp. 135–143.

Vander Linden, K. and D. Scott: 1995, 'Raising the Interlingual Ceiling with Multilingual Text Generation', in *Proceedings of the IJCAI Workshop on Multilingual Text Generation*, Montreal, Canada, pp. 95–101.

White, John S. (ed.): 2000, *Envisioning Machine Translation in the Information Future: 4th Conference of the Association for Machine Translation in the Americas, AMTA 2000*, Berlin: Springer.

Wilkins, Wendy: 1988, 'Thematic Structure and Reflexivization', in Wendy Wilkins (ed.), *Syntax and Semantics 21: Thematic Relations*, San Diego, CA: Academic Press, pp. 191–213.

XTAG-Group, T.: 1999, 'A Lexicalized Tree Adjoining Grammar for English', Technical report, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA.