

Interactive Authoring of Logical Forms for Multilingual Generation

Ofer Biller, Michael Elhadad, Yael Netzer

Department of Computer Science

Ben Gurion Univerisy

Be'er-Sheva, 84105, Israel

{billero, elhadad, yaeln}@cs.bgu.ac.il

Abstract

We present an authoring system for logical forms encoded as conceptual graphs (CG). The system belongs to the family of WYSIWYM (What You See Is What You Mean) text generation systems: logical forms are entered interactively and the corresponding linguistic realization of the expressions is generated in several languages. The system maintains a model of the discourse context corresponding to the authored documents.

The system helps users author documents formulated in the CG format. In a first stage, a domain-specific ontology is acquired by learning from example texts in the domain. The ontology acquisition module builds a typed hierarchy of concepts and relations derived from the WordNet and Verbnet.

The user can then edit a specific document, by entering utterances in sequence, and maintaining a representation of the context. While the user enters data, the system performs the standard steps of text generation on the basis of the authored logical forms: reference planning, aggregation, lexical choice and syntactic realization – in several languages (we have implemented English and Hebrew - and are exploring an implementation using the Bliss graphical language). The feedback in natural language is produced in real-time for every single modification performed by the author.

We perform a cost-benefit analysis of the application of NLG techniques in the context of authoring cooking recipes in English and Hebrew. By combining existing large-scale knowledge resources (WordNet, Verbnet, the SURGE and HUGG realization grammars) and techniques from modern integrated software development environment (such as the Eclipse IDE), we obtain an efficient tool for the generation of logical forms, in domains where content is not available in the form of databases.

1 Introduction

Natural language generation techniques can be applied to practical systems when the “input” data to be rendered in text

can be obtained in a cost-effective manner, and when the “output” requires such variability (multiple styles or languages, or customization to specific users or classes) that manually producing documents becomes prohibitively expensive.

The input data can be either derived from an existing application database or it can be authored specifically to produce documents. Applications where the data is available in a database include report generators (e.g., ANA [Kukich, 1983], PlanDoc [Shaw *et al.*, 1994], Multimeteo [Coch, 1998], FOG [Goldberg *et al.*, 1994]). In other cases, researchers identified application domains where some of the data is available, but not in sufficient detail to produce full documents. The “WYSIWYM” approach was proposed ([Power and Scott, 1998], [Paris and Vander Linden, 1996]) as a system design methodology where users author and manipulate an underlying logical form through a user interface that provides feedback in natural language text.

The effort invested in authoring logical forms – either from scratch or from a partial application ontology – is justified when the logical form can be reused. This is the case when documents must be generated in several languages. The field of multilingual generation (MLG) has addressed this need ([Bateman, 1997], [Stede, 1996]). When documents must be produced in several versions, adapted to various contexts or users, the flexibility resulting from generation from logical forms is also valuable. Another motivation for authoring logical forms (as opposed to textual documents) is that the logical form can be used for other applicative requirements: search, summarization of multiple documents, inference. This concern underlies the research programme of the Semantic Web, which promotes the encoding in standardized forms of ontological knowledge such as KIF [Berners-Lee *et al.*, 2001], [Genesereth and Fikes, 1992].

In this paper, we analyze an application of the WYSIWYM method to author logical forms encoded in Sowa’s Conceptual Graphs (CG) format [Sowa, 1987]. In a first stage, users submit sample texts in a domain to the system. The system learns from the samples a hierarchy of concepts and relations. Given this ontology, the author then enters expressions using a simple variant of the CG Interchange Format (CGIF) which we have designed to speed editing operations. The system provides realtime feedback to the author in English and Hebrew.

We evaluate the specific features of such a system which

make it cost-effective as a tool to author logical forms. We select the CG formalism as one of the representative of the family of knowledge encoding formalisms, which benefits from well-established inference and quantification mechanisms and standard syntax encodings in graphical and linear formats.

The editing system we developed can be seen as a CG editor motivated and expanded by natural language generation (NLG) techniques. The mixing of a practical ontology editing perspective with NLG techniques yielded the following benefits:

- Generation tasks such as aggregation and reference planning are easily expressed as operations upon CGs.
- The construction and maintenance of context according to models of text planning [Reiter and Dale, 1992], allow the author to break a complex CG into a manageable collection of small utterances. Each utterance links to a global context in a natural manner.
- We designed a compact form to edit a textual encoding of CGs taking into account defaults, knowledge of types of concepts, sets and individual instances and context. This format syntactically looks like a simple object-oriented programming language with objects, methods and attributes. We use an editing environment similar to a modern programming language development environment – with a browser of types and instances, intelligent typing completion based on type analysis, and context-specific tooltip assistance.
- The simultaneous generation of text in two languages (Hebrew and English) is important to distinguish between un-analyzed terms in the ontology and their linguistic counterpart.

We evaluate the overall effectiveness of the authoring environment in the specific domain of cooking recipes (inspired by [Dale, 1990]). We perform various usability studies to evaluate the overall cost of authoring cooking recipes as logical forms and evaluate the relative contribution of each component of the system: ontology, natural language feedback, user interface. We conclude that the combination of these three factors results in an effective environment for authoring logical forms.

In the paper, we first review the starting points upon which this study builds in generation and knowledge editing. We then present the tool we have implemented – its architecture, the knowledge acquisition module and the editor, we finally present the evaluation experiments and their results, and conclude with their analysis.

2 Related Work

Our work starts from several related research traditions: multilingual generation systems; WYSIWYM systems; knowledge and ontology editors. We review these in this section in turn.

2.1 Multilingual Generation

Multilingual texts generation (MLG) is a well motivated method for the automatic production of technical documents

in multiple languages. The benefits of MLG over translation from single language source were documented in the past and include the high cost of human translation and the inaccuracy of automatic machine translation [Stede, 1996], [Coch, 1998], [Bateman, 1997]. In an MLG system, users enter data in an interlingua, from which the target languages are generated.

MLG Systems aim to be as domain independent as possible (since development is expensive) but usually refer to a narrow domain, since the design of the interlingua refers to domain information. MLG systems share a common architecture consisting of the following modules:

- A language-independent underlying knowledge representation: knowledge represented as AI plans [Rösner and Stede, 1994] [Delin *et al.*, 1994], [Paris and Vander Linden, 1996], knowledge bases (or ontologies) such as LOOM, the Penman Upper-model and other (domain-specific) concepts and instances [Rösner and Stede, 1994].
- Micro-structure planning (rhetorical structure) - language independent - this is usually done by the human writers using the MLG application GUI.
- Sentence planning - different languages can express the same content in various rhetorical structures and planning must take it into consideration: either by avoiding the tailoring of structure to a specific language [Rösner and Stede, 1994] or by taking advantage of knowledge on different realizations of rhetorical structures in different languages at the underlying representation [Delin *et al.*, 1994].
- Lexical and syntactic realization resources (e.g., English PENMAN/German NIGEL in [Rösner and Stede, 1994])

As an MLG system, our system also includes similar modules. We have chosen to use Conceptual Graphs as an interlingua for encoding document data [Sowa, 1987]. We use existing generation resources for English – SURGE [Elhadad, 1992] for syntactic realization and the lexical chooser described in [Jing *et al.*, 2000] and the HUGG grammar for syntactic realization in Hebrew [Netzer, 1997]. For micro-planning, we have implemented the algorithm for reference planning described in [Reiter and Dale, 1992] and the aggregation algorithm described in [Shaw, 1995]. The NLG components rely on the C-FUF implementation of the FUF language [Kharitonov, 1999] [Elhadad, 1991] – which is fast enough to be used interactively in realtime for every single editing modification of the semantic input.

2.2 WYSIWYM

In an influential series of papers [Power and Scott, 1998], WYSIWYM (What You See Is What You Mean) was proposed as a method for the authoring of semantic information through direct manipulation of structures rendered in natural language text. A WYSIWYM editor enables the user to edit information at the semantic level. The semantic level is a direct controlled feature, and all lower levels which are derived

from it, are considered as presentational features. While editing content, the user gets a feedback text and a graphical representation of the semantic network. These representations can be interactively edited, as the visible data is linked back to the underlying knowledge representation.

Using this method, a domain expert produces data by editing the data itself in a formal way, using a tool that requires only knowledge of the writer’s natural language. Knowledge editing requires less training, and the natural language feedback strengthens the confidence of users in the validity of the documents they prepare.

The system we have developed belongs to the WYSIWYM family. The key aspects of the WYSIWYM method we investigate are the editing of the semantic information. Text is generated as a feedback for every single editing operation. Specifically, we evaluate how ontological information helps speed up semantic data editing.

2.3 Controlled Languages

A way to ensure that natural language text is unambiguous and “easy to process” is to constrain its linguistic form. Researchers have designed “controlled languages” to ensure that words in a limited vocabulary and simple syntactic structures are used (see for example [Pulman, 1996]). This notion is related to that of *sublanguage* [Kittredge and Lehrberger, 1982], which has been used to analyze and generate text in specific domains such as weather reports.

With advances in robust methods for text analysis, it is becoming possible to parse text with high accuracy and recover partial semantic information. For example, the DIRT system [Lin and Pantel, 2001] recovers thematic structures from free text in specific domains. Combined with lexical resources (WordNet [Miller, 1995] and Verbnet [Kipper *et al.*, 2000]), it is now possible to confirm the thesis that controlled languages are easy to process automatically.

Complete semantic interpretation of text remains however too difficult for current systems. In our system, we rely on automatic interpretation of text samples in a specific sublanguage to assist in the acquisition of a domain-specific ontology, as described below.

2.4 Graphical Editors for Logical Forms

Since many semantic encodings are described as graphs, knowledge editing tools have traditionally been proposed as graphical editors – where concepts are represented as nodes and relations as edges. Such a “generic graphical editor” is presented for example in [Paley *et al.*, 1997].

Conceptual graphs have also been traditionally represented graphically, and there is a standard graphical encoding for CGs. Graphical editors for CGs are available (e.g., [Delugach, 2001]).

While graphical editors are attractive, they suffer from known problems of visual languages: they do not scale well (large networks are particularly difficult to edit and understand). Editing graphical representations is often slower than editing textual representations. Finally, graphical representations convey too much information, as non-meaningful data may be inferred from graphical features such as layout of

font, which is not constrained by the underlying visual language.

2.5 Generation from CG

CGs have been used as an input to text generation in a variety of systems in the past [Cote and Moulin, 1990], [Bontcheva, 1995] and others.

In our work, we do not view the CG level as a direct input to a generation system. Instead, we view the CG level as an ontological representation, lacking communicative intention levels, and not linked directly to linguistic considerations. The CG level is justified by its inferencing and query retrieval capabilities, while taking into account sets, quantification and nested contexts.

Processing is required to link the CG representation level (see Fig. 1) to linguistically motivated rhetorical structures, sentence planning and lexical choice. In our work, CGs are formally converted to an input to a generation system by a text planner and a lexical chooser, as described below. Existing generation components for lexical choice and syntactic realization based on functional unification are used on the output of the text planner.

```
[Situation: #U7
  [Add]-
  (Agnt)->[Person: {YOU} ]
  (Ptnt)->[Eatable: {sa,pe} ]
  (Efct)->[Object]->(Otcmaf)->[Situation: {U3} ]
]->(Mood)->[Imperative]
```

Figure 1: Conceptual Graph in a linear representation.

3 Method and Architecture

We now present the system we have implemented, which we have called SAUT (Semantic AUthoring Tool). Our objective is to perform usability studies to evaluate:

- How ontological knowledge in the form of concept and relation hierarchies is useful for semantic authoring;
- How natural language feedback improves the authoring – and how feedback in two languages modifies the authoring process;
- How user interface functionality improves the speed and accuracy of the authoring.

The architecture of the system is depicted in Fig. 2.

The two key components of the system are the knowledge acquisition system and the editing component. The knowledge acquisition system is used to derive an ontology from sample texts in a specific domain. In the editing component, users enter logical expressions on the basis of the ontology.

3.1 Knowledge Acquisition

For the acquisition of the concepts/relations database, we use two main sources: Verbnet [Kipper *et al.*, 2000] and WordNet [Miller, 1995].

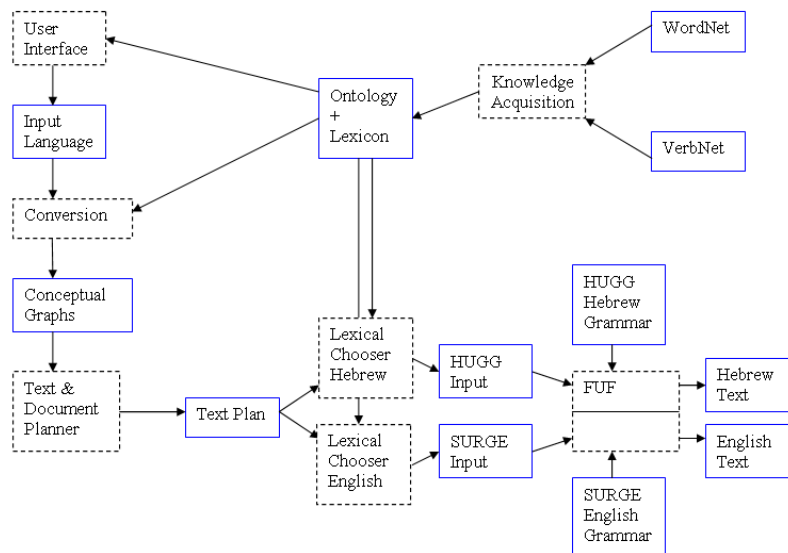


Figure 2: Architecture of the SAUT system

We use the information for bootstrapping concept and relation hierarchies. Given sample texts in the target domain, we perform shallow syntactic analysis and extract nouns, verbs and adjectives from the text. Dependency structures for verbs and nouns are also extracted. We currently perform manually anaphora resolution and word sense disambiguation, since automatic methods do not produce accurate enough results.

Given the set of nouns and adjectives, we induce the hypernym hierarchy from WordNet, resulting in a tree of concepts – one for each synset appearing in the list of words in the sample texts.¹

In addition to the concept hierarchy, we derive relations among the concepts and predicates by using the Verbnets lexical database [Kipper *et al.*, 2000]. Verbnets supplies information on the conceptual level, in the form of selectional restrictions for the thematic roles.

These relations allow us to connect the concepts and relations in the derived ontology to nouns, verbs and adjectives. The selectional restrictions in Verbnets refer to the WordNet conceptual hierarchy. In Verbnets, verbs are classified following Levin’s classes [Levin, 1993] and thus its representation is easily adjustable with our verb lexicon [Jing *et al.*, 2000], which combined information on argument structure of verbs from Levin, Comlex [Macleod and Grishman, 1995] and WordNet. The rich information on argument structure and selectional restrictions can be automatically adopted to the domain concepts database. Thus, by connecting a concept to a verb, given all the concepts that stand in relation to it in a specific CG (the verb’s arguments and circumstantials) – our lexical chooser finds the suitable structure (alternation) to map the CG to a syntactic structure.

The outcome of this process is useful in the lexical and syn-

tactic module of the system due to the flexibility it offers to the lexical chooser (a general word can be used instead of a specific word i.e. *vehicles* instead of *cars*, and for the generality of selectional restrictions on verb/adjective arguments).

Since there are no Hebrew parallels to WordNet/verbnets, we use a “naive” scheme of translating the English LC to Hebrew, with manual corrections of specific structures when errors are found.

Once the knowledge is acquired, we automatically updated a lexical chooser adopted to the domain. The lexical chooser maps the ontological concepts and relations to nouns, verbs and adjectives in the domain.

3.2 The SAUT Editor

To describe the SAUT editor, we detail the process of authoring a document using the tool. When the authoring tool is initiated, the next windows are presented (see Fig. 3):

- Input window
- Global context viewer
- Local context viewer
- CG feedback viewer
- Feedback text viewer
- Generated document viewer.

The user operates in the input window. This window includes three panels:

- *Defaults*: rules that are enforced by default on the rest of the document. The defaults can be changed while editing. Defaults specify attribute values which are automatically copied to the authored CGs according to their type.
- *Participants*: a list of objects to which the document refers. Each participant is described by an instance (or a

¹Although hypernym relations in WordNet define a forest of trees, we connect all trees with a general node.

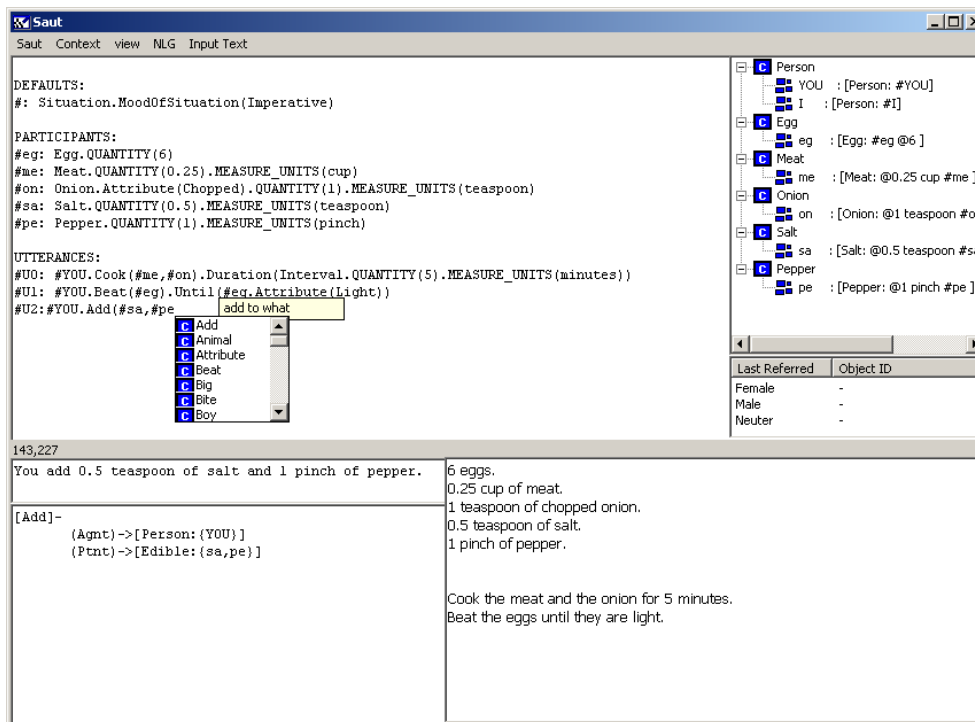


Figure 3: Snapshot of editing state in the SAUT system

generic) CG, and is given an alias. The system provides an automatic identifier for participants, but these can be changed by the user to a meaningful identifier.

- *Utterances*: editing information proposition by proposition.

The system provides suggestions to complete expressions according to the context in the form of popup windows. In these suggestion windows, the user can either scroll or choose with the mouse or by entering the first letters of the desired word, when the right word is marked by the system, the user can continue, and the word will be automatically completed by the system. For example, when creating a new participant, the editor presents a selection window with all concepts in the ontology that can be instantiated. If the user chooses the concept type "Dog" the system creates a new object of type dog, with the given identifier. The user can further enrich this object with different properties. This is performed using the "." notation to modify a concept with an attribute. While the user enters the instance specification and its initial properties, a feedback text and a conceptual graph in linear form are generated simultaneously. When the user moves to the next line, the new object is updated on the global context view. Each object is placed in a folder corresponding to its concept type, and will include its instance name and its description in CG linear form.

In the Utterances panel, the author enters propositions involving the objects he declared in the participants section. To create an utterance, the user first specifies the object which is the topic of the utterance. The user can choose one of the par-

ticipants declared earlier from an identifiers list, or by choosing a concept type from a list. Choosing a concept type will result in creating a new instance of this concept type. Every instance created in the system will be viewed in the context viewer. After choosing an initial object in an utterance, the user can press the dot key which indicates that he wants to enrich this object with information. The system will show the user list of expressions that can add information on this object. In CG terms, the system will fill the list with items which fall in one of the following three categories:

- *Relations* that can be created by the system and their selectional restrictions are such that they allow the modified object as a source for the relation.
- *Properties* that can be added to the concept object such as name and quantity.
- *Concept types* that expect relations, the first of whom can connect to the new concept. For example the concept type "Eat" expects a relation "Agent" and a relation "Patient." The selectional restriction on the destination of "Agent" will be for example "Animate". Therefore the concept "Eat" will appear on the list of an object of type "Dog".

The author can modify and add information to the active object by pressing the dot key. An object which itself modifies an object previously entered, can be modified with new relations, properties and concepts in the same manner. The global context is updated whenever a new instance is created

in the utterances. When the author has finished composing the utterance, the system will update the local context and will add this information to the generated natural language document.

The comma operator (“,”) is used to define sets in extension. For example, in Fig.3, the set “salt and pepper” is created by entering the expression #sa,#pe. The set itself becomes an object in the context and is assigned its own identifier.

The dot notation combined with named variables allows for easy and intuitive editing of the CG data. In addition, the organization of the document as defaults, participants and context (local and global) – provides an intuitive manner to organize documents.

Propositions, after they are entered as utterances, can also be named, and therefore can become arguments for further propositions. This provides a natural way to cluster large conceptual graphs into smaller chunks.

The text generation component proceeds from this information, according to the following steps:

- Pronouns are generated when possible using the local and global context information.
- Referring expressions are planned using the competing expressions from the context information, excluding and including information and features of the object in the generated text, so the object identity can be resolved by the reader, but without adding unnecessary information.
- Aggregation of utterances which share certain features using the aggregation algorithm described in [Shaw, 1995].

Consider the example cooking recipe in Fig.3. The author uses the participants section in order to introduce the ingredients needed for this recipe. One of the ingredients is “six large eggs”. The author first chooses an identifier name for the eggs, for example “eg”. From the initial list of concept types proposed by the system, we choose the concept type “egg”. Pressing the dot key will indicate we want to provide the system with further information about the newly created object. We choose “quantity” from a given list by typing “qu”. Seeing that the word “quantity” was automatically marked in the list. Pressing the space key will automatically open brackets, which indicates we have to provide the system with an argument. A tool tip text will pop to explain the user what is the function of the required argument. After entering number, we will hit the space bar to indicate we have no more information to supply about the “quantity”, the brackets will be automatically closed. After the system has been told no more modification will be made on the quantity, the “egg” object is back to be the active one. The system marks the active object in any given time by underline the related word in the input text.

Pressing the dot will pop the list box with the possible modifications for the object. We will now choose “attribute”. Again the system will open brackets, and a list of possible concepts will appear. The current active node in the graph is “attribute”. Among the possible concepts we will choose the “big” concept, and continue by clicking the enter key (the lexical chooser will map the “big” concept to the collocation

“large” appropriate for “eggs”). A new folder in the global context view will be added with the title of “egg” and will contain the new instance with its identifier and description as a CG in linear form.

Each time a dot or an identifier is entered, the system converts the current expression to a CG, maps the CG to a FUF Functional Description which serves as input to the lexical chooser; lexical choice and syntactic realization is performed, and feedback is provided in both English and Hebrew.

The same generated sentence is shown without context (in the left part of the screen), and in context (after reference planning and aggregation).

When generating utterances, the author can refer to an object from the context by clicking on the context view. This enters the corresponding identifier in the utterance graph.

4 Evaluation

The objectives of the SAUT authoring system are to provide the user with a fast, intuitive and accurate way to compose semantic structures that represent meaning s/he wants to convey, then presenting the meaning in various natural languages. Therefore, an evaluation of these aspects (speed, intuitiveness, accuracy and coverage) is required, and we have conducted an experiment with human subjects to measure them. The experiment measures a snapshot of these parameters at a given state of the implementation. We have isolated in the error analysis parameters which depend on specifics of the implementation and those which require essential revisions to the approach followed by SAUT.

4.1 User Experiment

We have conducted a user experiment, in which ten subjects were given three to four recipes in English (all taken from the Internet) from a total pool of ten, and the subjects had to compose semantic documents for these recipes using SAUT². The ontology and lexicon for the specific domain of cooking recipes were prepared in advance, and we have tested the tool by composing these recipes with the system. The documents the authors prepared are later used as a ‘gold standard’ (we refer to them as “reference documents”). The experiment was managed as follows: first, a short presentation of the tool (20 minutes) was given. Then, each subject got a written interactive tutorial which took approximately half an hour to process. Finally, each subject composed a set of 3 to 4 documents. The overall time taken for each subject was 2.5 hours.

4.2 Evaluation

We have measured the following aspects of the system during the experiment.

Coverage - answers the questions “can I say everything I mean” and “how much of the possible meanings that can be expressed in natural language can be expressed using the input language”. In order to check the coverage of the tool, we examined the reference documents. We compared the text generated from the reference documents with the original recipes and checked which parts of the information were

²All subjects were computer science students.

included, excluded or expressed in a partial way w.r.t. the original. We counted each of these 3 counts as a percentage of the words in the original recipe. We sum up the result as a coverage index which combines the 3 counts (correct, missing, partial) with a factor of 70% for the partial count.

The results were checked by two authors independently and we report here the average of these two verifications. On a total of 10 recipes, containing 1024 words overall, the coverage of the system is 91%. Coverage was uniform across recipes and judges. We perform error analysis for the remaining 9% of the un-covered material below.

Intuitiveness - to assess the ease of use of the tool, we measured the "learning curve" for users first using the system, and measuring the time it takes to author a recipe for each successive document (1st, 2nd, 3rd, 4th). For 10 users first facing the tool, the time it took to author the documents is as follows:

Document #	Average Time to author
1st	36 mn
2nd	28 mn
3rd	22 mn
4th	14 mn

The time distribution among 10 users was extremely uniform. We did not find variation in the quality of the authored documents across users and across number of document. The tool is mastered quickly, by users with no prior training in knowledge representation or natural language processing. Composing the *reference documents* (approximately 100-words recipes) by the authors took an average of 12 minutes.

Speed - we measured the time required to compose a document as a semantic representation, and compare it to the time taken to translate the same document in a different language. We compare the average time for trained users to author a recipe (14 minutes) with that taken by 2 trained translators to translate 4 recipes (from English to Hebrew).

Semantic Authoring Time	Translation Time
14 (minutes)	6 (minutes)

The comparison is encouraging - it indicates that a tool for semantic authoring could become cost-effective if it is used to generate in 2 or 3 languages.

Accuracy - We analyzed the errors in the documents prepared by the 10 users according to the following breakup:

- Words in the source document not present in the semantic form
- Words in the source document presented inaccurately in the semantic form
- Users' errors in semantic form that are not included in the former two parameters.

We calculated the accuracy for each document produced by the subjects during the experiment. Then we compared each document with the corresponding *reference document* (used here as a gold standard). Relative accuracy of this form estimates a form of confidence - "how sure can the user be that s/he wrote what s/he meant"? This measurement depends on the preliminary assumption that for a given

recipe, any two readers (in the experiment environment - including the authors), will extract similar information. This assumption is warranted for cooking recipes. This measure takes into account the limitations of the tool and reflects the success of users to express all that the tool can express:

Document #	Accuracy
1st	93%
2nd	92%
3rd	95%
4th	90%

Accuracy is quite consistent during the experiment sessions, *i.e.*, it does not change as practice increases. The average 92.5% accuracy is quite high.

We have categorized the errors found in subjects' documents in the following manner:

- Content can be accurately expressed with SAUT (*user error*)
- Content will be accurately expressed with changes in the SAUT's lexicon and ontology (*ontology deficit*)
- Content cannot be expressed in the current implementation, and requires further investigation of the concept (*implementation and conceptual limitations*)

Document #	Accuracy
User error	44%
Ontology deficit	23%
Tool limitations	33%

This breakdown indicates that the tool can be improved by investing more time in the GUI and feedback quality and by extending the ontology. The difficult conceptual issues (those which will require major design modifications, or put in question our choice of formalism for knowledge encoding) represent 33% of the errors - overall accounting for 2.5% of the words in the word count of the generated text.

5 Analysis

The current prototype of SAUT proves the feasibility of semantic authoring combined with natural language generation. The system includes a lexical chooser of several hundred verbs and nouns derived from WordNet in a specific domain.

The system is easy to use and requires training of less than one hour. User interface features make it very fast to enter CGs of the type required for a recipe. If the documents are generated in more than 2 languages, the tool can even become cost effective at its current level of ergonomics.

The current prototype indicates that combining techniques from NLG with User Interfaces techniques from programming languages editors results in an efficient knowledge editor. In future work, we intend to evaluate how to use semantic forms for summarization and inferencing. We also will evaluate how rhetorical information can be managed in the system, by applying the tool to different domains.

References

[Bateman, 1997] John Bateman. Enabling technology for multilingual natural language generation: the KPML de-

- velopment. *Natural Language Engineering*, 1(1):1 – 42, 1997.
- [Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. Semantic web. *Scientific American*, 2001.
- [Bontcheva, 1995] Kalina Bontcheva. Generation of multilingual explanations from conceptual graphs. In *Proc. of RANLP'97*, Batak, Bulgaria, 1995.
- [Coch, 1998] J. Coch. Interactive generation and knowledge administration in multimeteo. In *Proc. of the 9th Workshop INLG*, pages 300–303, Canada, 1998.
- [Cote and Moulin, 1990] D. Cote and B. Moulin. Refining sowa's conceptual graph theory for text generation. In *Proc. of IEA/AIE90*, volume 1, pages 528–537, Charleston, SC, 1990.
- [Dale, 1990] Robert Dale. Generating recipes: An overview of epicure. In Michael Zock Robert Dale, Chris Mellish, editor, *Current Research in Natural Language Generation*, pages 229–255. Academic Press, New York, 1990.
- [Delin *et al.*, 1994] Judy Delin, Anthony Hartley, Cécile L. Paris, Donia Scott, and Keith Vander Linden. Expressing Procedural Relationships in Multilingual Instructions. In *Proc. of the 7th. Int. Workshop on NLG*, pages 61 – 70, 1994.
- [Delugach, 2001] Harry Delugach. Charger: A graphical conceptual graph editor. In *Proc. of ICCS 2001 CGTools Workshop*, 2001.
- [Elhadad, 1991] Michael Elhadad. FUF user manual - version 5.0. Technical Report CUCS-038-91, University of Columbia, 1991.
- [Elhadad, 1992] Michael Elhadad. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. PhD thesis, Columbia University, 1992.
- [Genesereth and Fikes, 1992] M.R. Genesereth and R.E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
- [Goldberg *et al.*, 1994] E. Goldberg, N. Driedger, and R. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53, 1994.
- [Jing *et al.*, 2000] Hongyan Jing, Yael Dahan Netzer, Michael Elhadad, and Kathleen McKeown. Integrating a large-scale, reusable lexicon with a natural language generator. In *Proceedings of the 1st INLG*, pages 209–216, Mitzpe Ramon, Israel, 2000.
- [Kharitonov, 1999] Mark Kharitonov. Cfuf: A fast interpreter for the functional unification formalism. Master's thesis, BGU, Israel, 1999.
- [Kipper *et al.*, 2000] K. Kipper, H. Trang Dang, and M. Palmer. Class-based construction of a verb lexicon. In *Proceeding of AAI-2000*, 2000.
- [Kittredge and Lehrberger, 1982] R. Kittredge and J. Lehrberger. *Sublanguage: Studies of Language in Restricted Semantic Domains*. De Gruyter, Berlin, 1982.
- [Kukich, 1983] Karen Kukich. Knowledge-based report generation: A technique for automatically generating natural language reports from databases. In *Proc. of the 6th International ACM SIGIR Conference*, 1983.
- [Levin, 1993] Beth Levin. *English Verb Classes and Verb Alternations: A Preliminary Investigation*. University of Chicago Press, 1993.
- [Lin and Pantel, 2001] Dekang Lin and Patrick Pantel. DIRT@SBT@discovery of inference rules from text. In *Knowledge Discovery and Data Mining*, pages 323–328, 2001.
- [Macleod and Grishman, 1995] C. Macleod and R. Grishman. *COMLEX Syntax Reference Manual*. Proteus Project, NYU, 1995.
- [Miller, 1995] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
- [Netzer, 1997] Yael Netzer. Design and evaluation of a functional input specification language for the generation of bilingual nominal expressions (hebrew/english). Master's thesis, BGU, Israel, 1997.
- [Paley *et al.*, 1997] S.M. Paley, Lowrance, J.D., and P.D. Karp. A generic knowledge-base browser and editor. In *Proc. of the 1997 National Conference on AI*, 1997.
- [Paris and Vander Linden, 1996] Cécile Paris and Keith Vander Linden. DRAFTER: An interactive support tool for writing multilingual instructions. *IEEE Computer*, 29(7):49–56, 1996.
- [Power and Scott, 1998] Roger Power and Donia Scott. Multilingual authoring using feedback texts. In *Proc. of COLING-ACL 98*, Montreal, Canada, 1998.
- [Pulman, 1996] Stephen Pulman. Controlled language for knowledge representation. In *Proc. of the 1st Int. Workshop on Controlled Language Applications*, pages 233 – 242, 1996.
- [Reiter and Dale, 1992] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proc. of the 14th COLING*, pages 232–238, Nantes, France, 1992.
- [Rösner and Stede, 1994] D. Rösner and M. Stede. Generating multilingual documents from a knowledge base: The techdoc project. In *Proc. of COLING'94*, pages 339–346, Kyoto, 1994.
- [Shaw *et al.*, 1994] J. Shaw, K. Kukich, and K. McKeown. Practical issues in automatic documentation generation. In *Proceeding of the 4th ANLP*, pages 7–14, 1994.
- [Shaw, 1995] James Shaw. Conciseness through aggregation in text generation. In *Proc. of the 33rd conference on ACL*, pages 329 – 331, Morristown, NJ, USA, 1995.
- [Sowa, 1987] J. F. Sowa. Semantic networks. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence 2*. John Wiley & Sons, New York, 1987.
- [Stede, 1996] Manfred Stede. *Lexical semantics and knowledge representation in multilingual sentence generation*. PhD thesis, University of Toronto, 1996.