What is NLG?

Roger Evans and Paul Piwek and Lynne Cahill
Information Technology Research Institute
University of Brighton, UK
email: Firstname.Lastname@itri.brighton.ac.uk

Abstract

Giving an adequate general definition of the input to natural language generation (NLG), and hence to NLG itself, is a notoriously difficult problem, practically, theoretically and even methodologically. In this paper, we describe our recent experiences of implementing an NLG component of a larger question-answering system, and trying to understand and resolve some of these problems in this context. We examine the whole lifetime of an answer, from internal data structure to final expression as text, and look for characteristics of the processing which might help identify where NLG really begins. On the basis of this analysis we propose some principles to inform discussions on the scope of NLG as an individuated enterprise.

1 Introduction

Standard definitions of what a natural language generation (NLG) module should do are remarkably and notoriously asymmetrical. While there is broad agreement on what an NLG module should produce[1], there is virtually no agreed view about what its starting point should be. Existing implemented systems vary enormously in the kind of input they expect, what assumptions they make about it, and what they do with it before turning it into some kind of textual output. Historically, most implemented systems have been associated with a particular class of application (such as explanation, instruction or report generation), each with its own particular kind of input requirements (such as databases, logical forms or semantic networks).

This is a significant problem at a number of levels. For practical system-building, the NLG practitioner negotiating to provide a component of a larger system can find it difficult to address some very basic interface definition questions (and hence difficult to cost and resource the project accurately). At the theoretical level, it leads to a lack of detailed general proposals about the early stages of NLG processing – high level notions of *communicative goals* met through a process of *content selection* by reference to some kind of externally provided data seem to be as specific as one can get. Most fundamentally perhaps, at the methodological level it suggests a deep lack of clarity about exactly what NLG is or should be.

This paper results from our attempts to wrestle with these issues in the context of developing an NLG module as part of a larger legal advisory question-answering system, CLIME[2]. In this system, 'answers' arise initially as the output of a logical inference engine, in response to a user query, and are successively transformed into text describing, justifying and explaining the answer. Our initial perspective was primarily practical, as we attempted to determine where in a sequence of data transformations the processing should become the responsibility of

[1] Essentially natural language, although incorporation of layout and multi-media, and potential post-processing for the web or for speech output complicate the picture somewhat. The primary focus of this paper is on the input side, however.

[2] "Computerised Legal Information Management and Explanation" – see http://www.bmtech.co.uk/clime/ for further details.

the NLG module. We gradually realised, however, that answering this question was more than just a practical issue of negotiation with project partners – we really did not have a suitable theoretical position on which to base our judgement. The result was that the answer we came up with was rather arbitrary, and indeed not even consistent across the two different scenarios the system supported.

This paper presents a reflection on this process and an attempt to rationally reconstruct what we might have done. In section 2, we briefly review some relevant background from the literature, primarily discussing McDonald's early insightful contribution to this debate (McDonald, 1993), two more recent, and more practically oriented, positions (Reiter and Dale, 2000; Cahill et al., 1999), and some typical examples of actual systems. In section 3 we describe the two kinds of answer production in the CLIME system, and in section 4 we analyse these scenarios looking for aspects of the system's behaviour which distinguish NLG from non-NLG.

2 Background

2.1 McDonald's view

McDonald (1993) was one of the first authors to discuss the issue of NLG input (or 'source' in McDonald's terminology), with concerns very similar to our own: the lack of clear definition makes it difficult to convey our work to other computational linguists, and difficult to compare and evaluate work within our own community. He looked to Natural Language Understanding (NLU) research to provide a possible "pivot point", but found no useable answers, the logical forms delivered by the then current systems being too linguistically underspecified and logically overspecified for generation. These arguments hold good, perhaps even more forcefully, today: NLU is if anything *more* shallow now than it was in 1993, and McDonald's view that 'reasoning' components in generators are permeated with linguistic 'NLG' decisions is borne out by more recent analyses of systems such as Cahill and Reape (1998).

McDonald's conclusion was that NLG starts at "the first point where a speaker must appeal to her knowledge of language as she begins the process of carrying out some action through the use of language" (p.196). Aside from the procedural undertone, we find little to argue with in this definition – but it is somewhat lacking in the practical utility we aspire to here.

2.2 Reiter and Dale's standard architecture

Reiter and Dale (2000) present a 'standard' architecture for NLG systems which broadly accords with the actual architecture of many implemented systems (although see Cahill et al. (1999) for further discussion of this claim). This architecture includes (pp.42ff) a model of NLG input comprising a four-tuple $< k, c, u, d >$, where k is a knowledge source, c is a communicative goal, u is a user model and d is a discourse history. This definition covers a wide range of theoretical and practical systems, but primarily because it is very underspecified. In particular, as Reiter and Dale acknowledge, k is necessarily very application-dependent, and this makes it difficult to provide any more specific general characterisation of NLG input. We suggest that this point can be taken one step further: with this definition of NLG input, it is impossible to provide any general characterisation *of* NLG *itself*.

This input model is exemplified using the WEATHERREPORTER system, an idealised design study for an NLG system, similar to real weather-reporting systems such as FOG (Kittredge and Polguére, 1991). From the present perspective, the key points of interest relate to k and c. Reiter and Dale say (p.45):

> - k is a database of records that encode a number of meteorological data values ...
> - c is ... a communicative goal of type $SummariseMonth(m)$

Here we see that k is clearly completely non-linguistic, and c is linguistic, but high level (and application-specific). The consequences of this are that WEATHERREPORTER has to undertake three activities:

1. mapping non-linguistic into linguistic information
2. summarising a month's worth of information
3. producing text to communicate the summarised information

Unequivocally, we maintain, (1) and (3) are processes which any NLG system must undertake in some form. (2), however, is more controversial: it may be non-linguistic manipulation (such as averaging numbers) occurring before (1), it may be linguistic manipulation (such as genuine summarisation) occurring after (1), it may be both. Our question is *Is it* NLG*?*. The analysis we present below suggests that it is not (in either case), and that by taking this kind of manipulation out of the loop, NLG becomes a more coherent identifiable enterprise.

2.3 RAGS

The RAGS (Reference Architecture for Generation Systems) project (Cahill et al., 1999; Cahill et al., 2001a), also set out to define a reference architecture for NLG systems. Nevertheless it has very little to say about input – less, in fact, than Reiter and Dale. Cahill et al. (2001a) (pp. 1-2 and 5ff) is very clear about the kinds of information that an NLG system needs to manipulate (conceptual, semantic, rhetorical, document, syntactic and quote structures), but provides no details about how such systems obtain their input, or in what form. It does discuss ways in which NLG systems may interface to knowledge bases of conceptual facts, but does not explicitly claim them to be part of the *input*, indeed it allows for systems whose input is semantic, or in fact any other kind of RAGS representation. This flexibility was deemed necessary in order to account for systems whose input can range from templates with (conceptual or semantic) facts attached (Coch, 1996) to full linguistic texts to be abbreviated or rephrased for a particular reader (DiMarco et al., 1995).

2.4 Existing systems

The survey of applied NLG systems undertaken in the RAGS project (Cahill and Reape, 1998; Paiva, 1998) showed significant differences between systems that accept their input from a user and those which take their input from databases or knowledge bases. The former usually start with essentially linguistic input, with the user choosing semantic and rhetorical forms. In some cases (e.g. ModEx, (Lavoie et al., 1996)) the user actually chooses the nouns and verbs that will ultimately appear in the output text. So characterising the input to these systems does not pose a significant problem, because the user interacts more or less directly with the *internal* data structures of the generator, at some appropriate level.

In systems that accept input from an external source (such as database), however, the issue of input and the starting point of NLG is more difficult. Most of the systems surveyed by RAGS employ several initial modules that transform the input into a form that can be directly manipulated linguistically. A typical example of a system taking input from a database is PlanDoc (McKeown et al., 1994). The input to PlanDoc is a database of the activities performed by telephone engineers. This is transformed into a set of 'messages', and enriched with conceptual/semantic domain knowledge, before being passed to a content, rhetorical and syntactic planner and then on to realisation. In such a system, the start of NLG 'proper' is quite unclear – are the first two processes to be included or not?

Systems whose input is some type of logical form generally tend to undertake less processing before becoming obviously 'linguistic' (although the CLIME example discussed in detail below is an exception to this). For example, the Proverb system (Huang and Fiedler, 1997), which generates texts presenting a verbalised description of a natural deduction proof, translates its input into a set of Proof Communicative Acts (PCAs), each one of which represents some content to be expressed together with a communicative goal. However, even in a system such as this, the actual start of NLG is unclear, as the ordering and vene the rhetorical structure of the PCAs is implicit in the input proof.

3 A case study – CLIME

In this section we discuss the complete process of the generation of answers as it takes place in the CLIME system. CLIME is a web-based multilingual legal advisory system, which answers queries relating to ship-building and ship-operating regulations. The core knowledge source for the system is a set of such regulations encoded as (a) a conceptual graph and (b) a set of legal inference rules. Queries to the system are constructed using a WYSIWYM (Power et al., 1998) natural language interface and submitted to a **legal inference server** (LIS), described more fully in Winkels et al. (1998), Winkels et al. (2002).

The output of the LIS is the initial form of an 'answer', which is transformed into natural language (as HTML in English and French) and displayed to the user.

The system can respond to two kinds of query, corresponding to the two encodings of the regulations. **Conceptual Retrieval** (CR) queries consist simply of a list of concepts: the system searches the conceptual graph identifying individual regulations which relate to these concepts, either directly or indirectly via ontological links (such as `subtype-of`, `part-of`, `described-by`, `connected-to`). **Normative Assessment** (NA) queries consist of a description of a situation on a ship: the system applies the legal inference rules (derived from the regulations) to determine whether the situation described is legal according to the regulations, and also whether small changes to the situation would change this verdict. The two types of query lead to two types of answer, whose handling by the system illustrates two different views of the 'start' of NLG processing.

3.1 Conceptual Retrieval answers

For Conceptual Retrieval queries, the user enters a list of concepts, and the LIS returns the following information:

- an expanded set of concepts, found by following ontological links (for example, if `oil-tanker` was in the query, its superclass `ship` would be included;
- for each concept, the legal rules which refer to it;
- for each concept, a positive integer which represents the *centrality* of the concept, a measure of how many other concepts are related to it;
- for each of the referenced rules, the number of concepts in the rule that were in the original query.

This data structure undergoes the following transformations to produce textual answers:

1. The rules are ranked according to relevance to the user's query. Ranking depends on how many query concepts the rule refers to, and how central those concepts are (more central concepts are more common and so less relevant).
2. The top 30 rules in the ranked order are selected and the rest discarded.
3. For each concept in both the expanded set and these remaining 30 rules, a trace of how the concept is linked to concepts in the query is calculated – e.g., if the query contains `water` and the expanded set contains `liquid`, the fact that `water` is a subtype of `liquid` would be added.
4. Sentences describing each trace relation are generated – these serve as explanations of why the concept is included in the answer.
5. The answer is composed using a simple template-based approach: the query is repeated, followed by a canned text introduction to the answer, then each rule name is listed, together with the concepts it contains. Rule names are hypertext links to the corresponding rule texts (from the original regulations), concept names are hypertext links to the corresponding explanatory trace relation sentences.

3.2 Normative Assessment answers

For Normative Assessment queries, the user enters a set of facts describing a situation and the LIS returns the following information:

- a list of inference rule applications which apply to the situation, each consisting of (a) the rule name (b) the facts in the situation which triggered the rule and (c) the status which the rule assigns to the situation – allowed, disallowed or silent;
- a partial ordering over the rule applications according to legal precedence – more specific, or more recent, rules take precedence;
- a list of 'continuations': rule applications which would be triggered by additional facts and the status which the rule would assign to the extended situation.

The mapping from this input to HTML is achieved in the following stages:

1. Using the partial order for legal precedence, the system determines whether the situation in question is allowed or not (this is the actual

answer to the query) and which rules support, contradict or say nothing about this conclusion.

2. A subset of rule applications is selected to be expressed, namely those which contribute to the conclusion (the highest precedence rules), and those directly overruled by them.
3. Each of these selected rule applications and each continuation is transformed into a textual form describing the circumstance which caused the rule to be applicable. This is the most significant 'real' NLG the system undertakes, with simple aggregation of predicates with the same subject and referring expression generation.
4. Finally, the whole answer document is pieced together and transformed into HTML. The final answer consists of the conclusion (one of allowed, disallowed and silent), the rules supporting the conclusion, the rules against but overruled by those supporting, any rules which apply but draw no conclusion (usually because they include phrases like 'Normally'), and continuations (of the form "If ... were also the case, the conclusion would have been ...").

4 Analysis

In this section, we use the CLIME examples to help us try and articulate a possible 'definition' of NLG. Our approach is somewhat 'by approximation' from above and below: we propose two principles which we would like to think of as characterising NLG but which are too abstract to be directly useful, and then we look at the CLIME system to identify concrete instances of the principles in action, which are in turn too specific to be generally applicable. Our aim is to demonstrate that these upper and lower bounds are a useful initial step in refining our intuitions about what NLG really is.

> PRINCIPLE **A**: NLG is *linguistic* manipulation of data.

This principle is similar in spirit to part of McDonald's conclusion, but slightly more restrictive: McDonald's position allows *any* processing that occurs after the speaker first appeals to her knowledge of language to count as NLG, whereas principle **A** constrains NLG to be only linguistic manipulations (wherever they occur). We will attempt to characterise *linguistic* shortly, but two immediate observations here are that this principle (a) allows the data manipulated to be non-linguistic and (b) excludes non-linguistic manipulation, even of linguistic data.

Principle **A** alone is not specific to *generation*, and so a second principle is also required:

> PRINCIPLE **B**: NLG manipulates (linguistically) deeper information to produce shallower information.

The intuition here is that NLG does generation, not, say, parsing, where manipulation is in the opposite direction. However, the contrast is not only with parsing. This principle also excludes situations where the 'depth' of information is maintained, such as summarisation or translation (viewed as a unit – clearly if translation is achieved by parsing and regeneration, then the system may include a *bone fide* NLG component).

With just these two principles, we can return to the claim we made in section 2.2, that step (2) of WEATHERREPORTER is not NLG. If step (2) is non-linguistic, then it is ruled out by principle **A**; if it is linguistic then it is ruled out by principle **B** – it is *summarisation*, mapping from a more to a less detailed linguistic representation at the same 'depth'. This supports the underlying intuition that a goal like $SummariseMonth(m)$ is surely not just a generation task – there must be more to it than that.

A significant issue still remains with these principles in general of course: what do we mean by *linguistic manipulations* and *linguistic depth* of information. We do not have deep philosophical answers to these questions, but we can propose some practical observations. We consider linguistic depth first, because we can give a quite straightforward answer – the RAGS data hierarchy. As we have already noted, RAGS (Cahill et al., 1999; Cahill et al., 2001a) defined a data model for NLG comprising six types of information: conceptual semantic, rhetorical, document, syntactic and quote. Although RAGS makes no specific claims about processing dependence, it seems a plausible first approximation, borne out by actual implementations (e.g. Cahill et al. (2001b)), that this ordering corresponds to linguistic depth – that in the generation process, levels further down the list are in general

derived from levels above them. So for the present paper, we suggest the RAGS hierarchy as concrete support for principle **B**.

Let us now turn to the definition of *linguistic manipulations*. Our intuition is that a linguistic manipulation is an operation on data that is motivated or informed by linguistic considerations. We proceed by proposing two conditions on operations which classify them definitely linguistic or definitely not linguistic. We contend that a significant class of operations falls within the scope of these conditions (as illustrated by the CLIME examples). However, it is also true that a grey area of operations remains which eludes our conditions[3].

> CONDITION **I**: An operation is linguistic IF it is (potentially) language specific or requires only language specific resources.

A resource or operation is language specific if it is *not* valid for all languages, i.e., if there is a language to which it does not apply. In both CR and NA, a non-trivial amount of work is undertaken between logical form and text. Some of it clearly involves operations which are language specific or require language specific resources. Step 4 of CR requires language specific templates and a lexicon. Similarly, step 3 of NA is language specific. It involves rules which map conceptual representations of rule applications to semantic representations. The semantic representations are tailored to the grammatical templates which are available for the supported languages. Furthermore, it requires language specific aggregation, referring expression generation and surface realisation operations[4].

Note that condition **I** makes no reference to the input of linguistic operations: an operation can be linguistic even if its input is non-linguistic (e.g., the transformation from conceptual to semantic representations). What matters is whether the operation itself or the resources it uses are language specific. What is more, the fact that the input is linguistic does also not guarantee that the operations which operate on it are linguistic. An example is step 4 of NA. The input is clearly linguistic (consisting of an annotated text). In step 4. this annotated text is transformed into an HTML document. This mapping to HTML does, however, not fall within the scope of condition **I**: it is not language specific and does not require language specific resources.

> CONDITION **II**: An operation is *not* linguistic IF it involves removing, adding or changing information on non-linguistic grounds.

This condition is, of course, dangerously circular. However the CLIME system does provide several examples of the kind of 'non-linguistic grounds' that may exist. Quite a lot of work in CR queries is concerned with ranking of parts of the answer relative to each other. Although the answer formulation in this case is very simple, we feel this ranking behaviour is symptomatic of a more general and important property. Without this ranking, the answer would just be a set of facts to be expressed arbitrarily. An NLG component should not have to (or be able to) determine the relative importance of input facts itself – this is a non-linguistic judgement[5]. This puts step 1 of CR clearly outside of NLG.

Step 1 of NA is an even more extreme case. The user's query is really a yes/no question, but the LIS output does not give an answer directly. The first step, therefore is to derive the actual answer from the data the LIS provides, a process of essentially logical manipulation. We note that step 1 also organises information in what will eventually amount to rhetorical structure in the answer. However, we suggest that the rhetorical content is hardwired into the later stages of the generator; step 1 is identifying fillers for a potential rhetorical structure tree, but it is not actually generating new rhetorical structure. So we maintain that condition **II** places step 1 of NA outside of NLG.

An example of a potential grey area is step 3 of CR. Its data manipulations are non-linguistic (just

[3] In particular, language universals in Chomsky's sense are not captured – our conditions cannot identify a boundary between universal linguistic and more general cognitive knowledge.

[4] This condition also distinguishes some more classical cases: determining whether to the bottom storey of a building is the *ground* or *first* floor is linguistic, while determining whether a temperature is Celsius or Farenheit is not.

[5] This is not to say that it cannot sometimes be achieved by linguistic means. Indeed many system track the effect of previous utterances on salience and use this to determine linguistic behaviour. But salience itself is not linguistic, and may arise, for example, simply from the context.

building ontological chains) and it is not language-dependent. However it is arguable that it *is* linguistically motivated – these structures are created to support generation of explanatory text fragments. However, the NLG system has not *chosen* to produce an explanation, say as a rhetorical strategy: including explanatory text is as hard-wired as including the answer for NA, as just discussed. So like step 1 of NA, we conclude that step 3 of CR is outside of NLG.

Finally, it seems a reasonable property of an output module that it should output what it is told to output. In NLG terms, this particularly means it should communicate everything it is asked to. This does not mean it has to express everything, merely that if it chooses not to express something, it must be able to justify that on linguistic grounds. Turning this on its head, any manipulation of an answer which throws information away *without* such justification should therefore not count as part of NLG. In the present example step 2 of both CR and NA undertake just such a move, which again suggests that step 2 is not linguistic in both cases.

In summary, condition **I** classifies CR 4 and NA 3 as linguistic. Condition **II** tells us that CR 1,2 and 3 and NA 1 are not linguistic. Finally, CR 5 and NA 4 are not purely linguistic. They draw togther the components of the answer, but they deal also with rendering the answer text in HTML and thereby change and add information on non-linguistic grounds.

5 Conclusions

In this paper we have looked at the problem of defining a satisfactory notion of NLG. The problem is in some ways like trying to define 'water' by example: most convenient examples of water include a container which is not part of the water itself, but is difficult to precisely distinguish from it. Most concrete examples of NLG are similarly embedded in their own 'containers' which crucially provide their input, but also make drawing sharp distinctions often difficult. But as water is 'the liquid part', we have attempted to put some substance on the notion that NLG is 'the linguistic part' of a communicative output system.

In pursuit of this goal, we have proposed two principles which capture the intuition of 'NLG'. Firstly, principle **A** restricts NLG to linguistic manipulations. The meat of principle **A** resides in the conditions **I** and **II** which provide a partial characterization of the notion of a *linguistic operation*. Secondly, principle **B** distinguishes NLG from other possible linguistic functions in a system. In order to give some concrete justification to these principles, we showed how they can be used to analyse the behaviour of the CLIME system answer generation modules, in order to say more precisely which parts of the system are doing NLG.

We do not present these as the final word on characterising NLG, but as the beginning of a discussion, which we hope others will take forward with us. We note that if even these conclusions are useful, some parts of what traditionally has been viewed as content selection may come to be seen as outside of NLG proper. We do not intend this as an attack or criticism of this valuable work in the field, but merely as a realignment of boundaries in the hope that a little more coherence might thereby accrue to NLG as a maturing discipline.

Finally, the punchline: where *were* the boundaries of the NLG components in the CLIME system? Answer: for CR queries it was dead right, at least on the input side: after step 3. For NA queries it was dead wrong: *all* the processing described was undertaken by the NLG module!

Acknowledgements

The CLIME system was developed by British Maritime Technology Ltd., Bureau Veritas, TXT E-Solutions SPA, the University of Amsterdam and the University of Brighton, supported by the European Commission ESPRIT initiative, project number EP 25.414. The authors also acknowledge the contribution made by Neil Tipper to the development of CLIME; discussions with colleagues at ITRI about this paper, especially Kees van Deemter; and the support and comments of the anonymous reviewers.

References

L. Cahill and M. Reape. 1998. Component tasks in applied NLG systems. Technical Report ITRI-99-05, ITRI, University of Brighton.

L. Cahill, C. Doran, R. Evans, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper. 1999. In Search of

a Reference Architecture for NLG Systems. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 77–85, Toulouse, France.

L. Cahill, R. Evans, C. Mellish, D. Paiva, M. Reape, and D. Scott. 2001a. The RAGS Reference Manual . ITRI, University of Brighton.

Lynne Cahill, John Carroll, Roger Evans, Daniel Paiva, Richard Power, Donia Scott, and Kees van Deemter. 2001b. From RAGS to RICHES: exploiting the potential of a flexible generation architecture. In *Proceedings of ACL/EACL 2001*, pages 98–105, Toulouse, France.

J. Coch. 1996. Overview of AlethGen. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 25–28, Herstmonceux, Sussex, UK.

C. DiMarco, G. Hirst, L. Wanner, and J. Wilkinson. 1995. Healthdoc: Customizing patient information and health education by medical condition and personal characteristics. In *First International Workshop on Artificial Intelligence in Patient Education*, Glasgow, UK.

X. Huang and A. Fiedler. 1997. Proof verbalization as an application of nlg. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Nagoya, Japan.

R. Kittredge and A. Polguére. 1991. Generating extended bilingual texts from application knowledge bases. In *Proceedings on Fundamental Research for the Future Generation of Natural Language Processing*, pages 147–160, Kyoto, Japan.

B. Lavoie, O. Rambow, and E. Reiter. 1996. The modelexplainer. In *Proceedings of the Eighth International Workshop on Natural Language Generation*, pages 9–12, Herstmonceux, Sussex, UK.

D. McDonald. 1993. Issues in the choice of a source for natural language generation. *Computational Linguistics*, 19(1):191–197.

K. McKeown, K. Kukich, and J. Shaw. 1994. Practical issues in automatic documentation generation. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 7–14, Stuttgart, Germany.

D. Paiva. 1998. A survey of applied natural language generation systems. Technical Report ITRI-98-03, ITRI, University of Brighton.

R. Power, D. Scott, and R. Evans. 1998. What You See Is What You Meant: direct knowledge editing with natural language feedback. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI'98)*, Brighton, UK.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK.

R. Winkels, A. Boer, J. Breuker, and D. Bosscher. 1998. Assessment Based Legal Information Serving and Cooperative Dialogue in CLIME. In *Proceedings of JURIX-98*, pages 131–146, GNI, Nijmegen, Netherlands.

R. Winkels, A. Boer, and R. Hoekstra. 2002. CLIME: Lessons Learned in Legal Information Serving. In F. van Harmelen, editor, *ECAI-2002: Proceedings of the 15th European Conference on Artificial Intelligence*, Amsterdam, NL. IOS Press.