# The Generation of Cryptic Crossword Clues

G. W. SMITH
*Logica UK Ltd, 64 Newman Street, London W1*
J. B. H. DU BOULAY*
*Cognitive Studies Programme, University of Sussex, Falmer, Brighton, Sussex BN1 9QN*

***A program is described which can generate certain classes of cryptic clues automatically. The program can be used in conjunction with programs developed elsewhere for crossword compilation and clue insertion.***

*Received March 1985*

## 1. INTRODUCTION

P. D. Smith[1,2] has shown how programs can assist the crossword compiler. One of his programs built an interlocking grid of words and blanks, the eventual solution of the crossword. The other selected prestored clues from a file according to various criteria. One drawback to the latter as a practical system is that it selected from the predefined clues and could not therefore produce a really novel crossword. It is clearly beyond the state of the art to expect a program to produce intelligent clues that rely on extensive real-world knowledge. However, many crosswords contain a fair number of cryptic clues. This paper describes a program that can generate certain classes of cryptic clue automatically. We start by describing what we mean by cryptic clues and then go on to describe the generation algorithm which we have implemented in Prolog.[3]

## 2. CRYPTIC CLUES

There are two main types of crossword clues, definitional and cryptic. Definitional clues are usually short in length and have a number of alternative answers. Cryptic clues are generally more like an English sentence and the solver usually has no doubt when an answer is correct. For example, the cryptic clue

*Is able follows old bob*

has the solution

*scan.*

Cryptic clues are difficult to solve if one is not familiar with them. Most clues adhere to a certain structure, so one must scan the clue for keywords and key phrases associated with that clue type in order to solve it. There are many different types of clue and each has its own peculiarities which, if recognised by the solver, give a hint as to the solution of the clue. To build a clue of a certain type we can, in effect, perform the reverse of the solving operation. That is, build the clue around these keywords.

The generation process is capable of producing eight separate types of clue, some of which can be further broken down to produce different clues. The eight basic types are:

(1) embedded words, e.g. *farmed* (*arm* inside *fed*)
(2) embedded word, e.g. *bevel* (*eve*)
(3) adjacent words, e.g. *hotdog*
(4) words beginning '*re*' e.g. *remember*
(5) part words, e.g. *apricot* (*apri april*)
(6) noun anagrams
(7) verb anagrams
(8) simple anagrams

Types (1), (3), (4) and (5) can be broken down further to cases where in (1) we can say that a word either 'surrounds' or 'is surrounded by' another; (3) one word can either come 'before' or 'after' the other; in (4) the word following 're' constitutes a noun or verb definition; and in (5) the part word comes 'before' or 'after' the other word.

Associated with each of the basic types there is a set of frames containing keywords and slots to be filled. The keywords indicate the kind of clue it is to the solver. Thus associated with type 1 is the frame

—— inside —— is ——

where the blanks are generated from a dictionary, as explained below.

We assume in the following discussion that a filled crossword skeleton has been produced and that the task is to generate a clue for each of the words in the skeleton.

The first problem is to split the word we are dealing with into chunks, or sublists, of characters that have some meaning. Armed with these sublists and the clue frame, we fit phrases associated with the sublists into the frame and join them using the keywords mentioned earlier.

These phrases are in fact the definitions associated with the sublist in a dictionary. Each definition has a flag to indicate whether it is a noun, verb or adjective definition. A word may take more than one class of definition. If this is so then each definition is kept in a list along with its class flag and the process can extract the class of definition it requires. We have taken the class of definition into account so that we can produce a clue that is either very nearly or exactly grammatically correct.

In very few cases will the grammar be exactly correct, but by using the simple grammar rules written into the process we can reduce the number of nonsense sentences being built.

Really silly clues can, of course, still be excluded by the crossword compiler. The following is an example of a less than sublime clue for the solution *let*:

*the French that beside almost a pronoun generates lease.*

* To whom correspondence should be addressed.

## 3. CLUE GENERATION

Each word used to fill the skeleton is held in a used-word list. The word itself is represented as a list of ASCII values in Prolog. When an attempt is made to generate a clue the word is taken from the used-word list and handed to the clue generation process.

A word may satisfy the conditions for several clue types, therefore the most difficult clue type to solve is attempted first. The easiest types to solve are tried last.

### 3.1 Word splitting

Depending on which clue type we are attempting to build, the word is split into either two or three sublists. In the case where two sublists are produced the program looks for adjacent words, or parts of words.

e.g. (1) *hotdog* becomes *hot dog*.

(2) *apricot* becomes *apri cot*.

When three sublists are formed the program looks for embedded words.

e.g. *farmed* becomes *arm fed*.

the three sublists were (*f*) (*arm*) (*ed*). The first and last sublists are appended to form (*fed*).

In both cases, after an initial attempt at splitting, the appropriate dictionaries are searched for words which match the sublists. If a match is found for each sublist the splitting stops and the program goes on to the next stage. If no match is found the attempt to find a suitable splitting continues.

At present, because the test dictionary is small, this is a reasonable way to proceed. With a larger dictionary it would be more sensible to perform all possible splits, put the fragments into alphabetic order and then make a single pass through the dictionary.

### 3.2 Definition gathering

When matches to the sublists are found the definitions are gathered and their class flags checked to make sure that there exists the required class of definition. If the required class of definition is not found then that clue type fails and the next clue type is attempted. Once definitions of the required class are obtained, they are joined together by the connectives (keywords) or form the clue.

### 3.3 Keywords

Keywords are used to build up the clue sentence. Their function is to help the solver decide which type of clue is present and to give hints as to the position of the sublists within the answer.

e.g. *further inside read only memory is uncontrollable.*

This is one clue produced by the system. The first keyword is *inside* – this indicates that the answer may involve embedded words. The second keyword is *is*, which indicates that the following phrase may be the definition of the answer as a whole word. The answer is *random*.

Each clue type may have a set of keywords associated with it, therefore a random number generator is used to pick a keyword from the set. In this way the process can produce clues that are of the same type but which have different keywords. As more clue types are added to the system the number of keywords can be increased accordingly.

### 3.4 Building the clue

Once the system has chosen the keywords the clue is built up by using the keywords to connect the definitions obtained earlier.

Each clue type has a sentence frame which it uses to decide where the definitions and keywords are to be placed within the clue. It is then a simple case of inserting the keywords and definitions and printing the clue number and the clue itself.

### 3.5 Summary

There are four stages in clue generation: (1) word splitting, (2) dictionary search and definition gathering, (3) selection of keyword(s), (4) building of clue.

Each has its problems, limitations and drawbacks. Stage (1) may take very little time, but combined with stage (2) the process can slow down considerably if a lot of backtracking is required until suitable sublists are found.

Stages (3) and (4) are relatively straightforward and are dependent only on the types of clue the process can produce. With future extensions the number of different types possible should increase and therefore provide aesthetically pleasing puzzles. Some examples of clues are given in Table 1.

**Table 1. Sample clues**

| | |
|---|---|
| *Further inside read only memory is uncontrollable* | *random* |
| *Shape edge day before within beautiful boundaries* | *bevel* |
| *With reference to subscription payer bring to mind* | *remember* |
| *American snack very warm domesticated animal* | *hotdog* |
| *Bed after almost a month produces fruit* | *apricot* |
| *Look at visual organ* | *eye* |
| *Hated hiding the end* | *death* |
| *Formation in rugby next to large bag* | *rucksack* |
| *Is able follows old bob* | *scan* |

## 4. CONCLUSION

The clue generation is simple in design but can produce relatively challenging clues. It is slow, this being due to the time taken to search the dictionaries after each word split; however, we believe that it is still faster at producing clues than a human compiler would be. At present it is implemented in Prolog running on a PDP11/44 under Unix.

The aim of the project was to show that a cryptic puzzle could be generated without a great deal of real-world knowledge.

New clue types can be easily added to the generation process; this would increase the variety and allow puzzles to contain many more types of clues, e.g. quotations and homophones. The time taken to search the dictionaries for matches is considerable and will increase as the dictionaries grow in size. A better dictionary structure and avoiding multiple passes through the dictionary will cut down the search time and speed up the overall system.

With fine tuning the process could grow from a basic test model to a sophisticated puzzle producing system.

## REFERENCES

1. P. D. Smith, XENO: computer-assisted compilation of crossword puzzles. *The Computer Journal* **26** (4), (1983).
2. P. D. Smith and S. Y. Steen, A prototype crossword compiler. *Computer Journal* **24** (2), 107 (1981).
3. Clocksin and Mellish, *Programming in Prolog* (1981).

# Announcements

23 MARCH 1987

Because mere automation is no longer the primary objective of managers the Office Automation Conference will become the Business Systems and Applications Conference when it opens on 23 March, 1987 at New York City's Jacob K. Javits Convention Center.

The thrust of the new conference will be to examine how people and machines can collaborate for greater creativity and productivity rather than how machines can replace human labour. This is a result of recent advancements in more sophisticated technologies (software, communications, CAD/CAM and image systems). From the introduction of OAC in 1980, the focus has continually been expanding into these new technologies and now the time has come to reflect what the conference has become: a Business Systems and Applications Conference.

The new conference will be the best possible combination of educational sessions, to advance the state of knowledge, with comprehensive exhibits, to demonstrate the state of the art.

We wanted to let you know of this development as soon as possible. Full press materials will be available from OAC '86 (24–26 March) at Houston, Texas.

In the meantime, please call Katherine Stormont (publicist) if you have any questions, on (703) 620–8936.

22–23 SEPTEMBER 1986

**TW2–The Second Tesseral Workshop.** University of Reading, UK.

*Objectives*

The goal of the workshop is to explore further the application and theory of hierarchical data structures in general and tesseral methods in particular. Such methods touch many areas (spatial databases, image processing, computer graphics, computerised cartography, mathematical combinatorics and tessellations). A second aim is to provide a forum where common problems can be shared and solutions proposed.

*Topics of Interest*

Database – the use of tesseral addresses, interleaved bit representations, search strategies, system architectures, knowledge representation.

Theory – multi-dimensions, constructive/axiomatic approaches, scale independence, amalgamator theory, lattice point rings, pattern algebra, generalised balance ternary representation, number representations.

Applications – image segmentation, $2\frac{1}{2}$ dimensions, solid modelling, computer assisted cartography, connected components.

Hard/software – tabular methods instruction sets, programming languages, implementation architectures, display devices and algorithms, tesseral algorithms.

*Organisation*

The Programme Committee invites papers in the areas of interest outlined above. Please send initial details and an abstract to:

Sarah Bell, TW2 Programme Committee, NUTIS, University of Reading, Whiteknights, Reading RG6 2AH, United Kingdom.

as soon as possible. (The Proceedings will also contain transcripts of the TW1 Discussion Sessions).

For further details regarding the Workshop contact:

Gary Robinson, TW2 Organising Committee, NUTIS, University of Reading, Whiteknights, Reading RG6 2AH, United Kingdom. Telephone: Reading (0734) 87512. Telex 847813 RULIB G.

14–16 JULY 1986

**BNCOD–5: Fifth British National Conference on Databases** University of Kent at Canterbury. BNCOD is a series of conferences on database techniques and theory, organised in association with the British Computer Society. The aim is to bring new developments and current research to the attention of the widest possible range of database practitioners. This fifth conference in the series will feature the following main themes:

– databases, knowledge bases and semantics
– wider applications of databases.

Professor A. Sølveberg, from the University of Trondheim Norwegian Institute of Technology, has been invited to present a paper, and he has chosen as his subject area databases, knowledge bases and systems analysis.

A range of both complementary and contrasting papers in the general area of knowledge-base enhancements for databases and semantic integrity have been selected for presentation, and a panel discussion has been arranged to provide a forum for discussion of this topical subject area. For the theme of wider applications of databases, papers have been selected on statistical database systems, systems for scientists, an expert system using Prolog linked to a relational database, and databases for software engineering. In addition, papers will also be presented on a novel graphics-based user interface, the integration of databases at the conceptual level, and the implementation of a relational view of a Codasyl DBMS.

*For further details of the conference, and a booking form, please contact*:
Dr Elizabeth Oxborrow, Computing Laboratory, University of Kent at Canterbury, Canterbury, Kent CT2 7NF. 0227-66822, ext. 7960.