

# Computational rules for generating punning riddles<sup>1</sup>

KIM BINSTED and GRAEME RITCHIE

## *Abstract*

*Riddles based on simple puns can be classified according to the patterns of word, syllable or phrase similarity they depend upon. We have devised a formal model of the semantic and syntactic regularities underlying some of the simpler types of punning riddle. We have also implemented this preliminary theory in a computer program which can generate riddles from a lexicon containing general data about words and phrases; that is, the lexicon content is not customized to produce jokes. An informal, formative evaluation of the program's results by a set of human judges suggests that the riddles produced by this program are of comparable quality to those in general circulation among school children.*

## **Introduction**

One very common form of humor is the question-answer joke, or riddle. A significant subset, probably the vast majority, of these jokes are based on some form of pun (question-answer puns make up almost a third of the riddles in (Webb 1978)). For example:

What do you get when you cross a sheep with a kangaroo?  
*A woolly jumper*

Riddles of this general sort are of particular interest for a number of reasons. The linguistics of riddles has been investigated before (Pepicello and Green 1984). Also, there is a large corpus of riddles to examine: books such as *The Crack-a-Joke Book* (Webb 1978) record them by the

thousand. Finally, riddles exhibit regular structures and mechanisms, which could be modeled and used to generate new riddles.

There appear to be certain well-defined subclasses of this genre, distinguished by the type of pun, or the arrangement of punning words and phrases. We have devised a formal model of the punning mechanisms underlying some of these subclasses; it is a symbolic account of the configurations of lexical entities required to make up each of these types of riddle. This model is intended to cover a subclass of punning riddles: namely, those that use the substitution to exchange phonologically-similar words in a noun phrase. This subtype is quite common, and is exemplified by the “woolly jumper” joke, above.

In addition, we have implemented a computer program which uses these symbolic rules to construct punning riddles from a lexicon (a store of words or phrases with syntactic and semantic information about each entry). Our program is more restricted than our model, in that it substitutes only homophones (phonologically-identical words) into noun phrases, and these phrases can only be used as the punchline of the constructed riddle. We have used this system to generate a large number of jokes, including some which were unknown to the authors (and are perhaps novel). An informal formative evaluation of the performance of this program suggests that its output, although poor, is not significantly worse than that produced by human composers of such riddles.

In the methodological issues section, below, we will set out the assumptions of our approach, which relies more on generative linguistics and artificial intelligence than on literary or psychological studies. Then we shall outline the scope of our project, introduce the theoretical constructs we use to describe punning riddles, and the computer program ( $J_A^P E-1$ ) which embodied these ideas. After summarizing the evaluation which we carried out, we shall relate the work to other research, and suggest some ways in which these ideas could be developed further.

## **Methodological issues**

### *Generative linguistics*

One of the major contributions which Chomsky (1957, 1965) made to the field of theoretical linguistics was the establishment of a methodologi-

cal framework for the study of language. There were several tenets to this approach, either explicitly or implicitly:

1. The aim is to define symbolic rules and structures which characterize what constitutes a sentence of a language and what does not.
2. These descriptions should be sufficiently precise and detailed that there is no doubt about what they predict, to the extent that it would in principle be possible to check the predictions mechanically, for instance using a computer.
3. These symbolic accounts can be empirical without large scale collection of data or statistical studies. The linguist compares what sentences are known to exist in the language with those predicted by the rules. Often, falsification of a proposed set of rules can be achieved from a relatively small set of examples.
4. The rules should capture regularities in the data. That is, sentences which have some inherent similarity (syntactically, semantically, etc.) should be described in a similar way by the rules, and systematic alterations to the symbolic descriptions should correspond to systematic changes in the language phenomena being described.

We have to a large extent adopted these attitudes in our study of riddles, as have various other humor researchers (some tacitly). We have attempted to devise abstract symbolic accounts of the detailed mechanisms underlying our chosen set of phenomena (certain types of punning riddle), we have defined these rules precisely (as shown by the computer implementation), and we believe that they show regularities in exactly the way that linguists expect grammars to display generalizations about sentences.

### *Artificial intelligence*

Within artificial intelligence, a research paradigm sometimes known as experimental programming is common. In this, a computer program is used to explore ideas, rather than to provide a final and conclusive test of a single well-articulated theory or to operate as a polished piece of software. Although the methodology owes a lot to the notion of an “experiment” in traditional science, it does not rely on critical experiments to falsify abstract theories (although it would be satisfying if this were the case). Rather, the researcher attempts to clarify his/her ideas by posing the question: “What would it take to have a computer program perform this task?” Concrete fleshing out of embryonic ideas then consists of trying to construct a computer program. This not only forces a degree

of detail and precision (see comments above on linguistic methodology), it also provides a readily testable version of the draft theory. Running the program and observing its behavior (not just its final results but also how it achieves them) can provide useful insights into weaknesses of the ideas, and may even inspire possible amendments or extensions to the proto-theory. (See Buchanan 1988; Newell and Simon 1976; Ritchie 1994 for further discussion of this approach).

The work described here can be seen as exemplifying this approach. The central task of the project was to design, implement, and test a computer program, but the important product of the work was not the program itself; rather, it was the set of ideas that we developed in the course of the work.

### *Humor and artificial intelligence*

Artificial intelligence (AI) has delved into so many areas of human behavior — vision, mathematics, planning and natural language processing, among others — that it is sometimes difficult to find an area which is not within AI's research domain. If, as Minsky claims, a suitable goal for AI research is to get a computer to do "... a task which, if done by a human, requires intelligence to perform" (1963), then most of human experience is a fit subject for research.

For theoretical research, however, there is also the requirement that the results must be falsifiable — that is, it must be possible to do an experiment which could disprove any claims of success. This makes the artistic side of human nature hard to investigate because, although art is definitely within the domain claimed by AI, it is difficult (if not impossible) to disprove the claim "this is good art" (or even "this is art") no matter who or what produced the work in question. Poetry, painting, and music all suffer from this problem.

A second factor to take into account is whether or not a formal model of the task has been, or could feasibly be, devised. For a task to be computationally tractable, it is necessary (but not sufficient) that a formal description of the task can be developed.

These two constraints, falsifiability and formalisability, reduce the domain of AI research considerably, particularly when looking at artistic intelligent behavior.

Humor generation is falsifiable in that there is a simple test of its success: whether the audience is amused or not. Designing an experiment to measure the humor response is not so simple; however, suitable experimental methods have been developed by other researchers (Derks, Lewis, and White Jr. 1981; Derks 1984; Ruch, McGhee, and Hehl 1990). Used with care, these give us a relatively rigorous way of testing the success of a humor generator (human or machine).

Although no computationally tractable model of humor as a whole has yet been developed, we believe that by tackling a limited and linguistically-based set of phenomena, it is realistic to develop a formal symbolic account.

## Phenomena considered

### *Pepicello and Green's riddle theory*

There are numerous collections and analyses of riddles, from the viewpoints of anthropology, sociology, literature and related fields. Apart from providing an overview of humor (and possibly some target jokes to replicate), however, these works were too discursive or informal to be of direct use in this project. In fact, the only work that explores this genre of joke in enough detail is Pepicello and Green (1984). We follow their terminology throughout this paper.

In their book, Pepicello and Green describe the various grammatical, written and visual strategies incorporated in riddles.<sup>2</sup> They hold the common view that humor is closely related to ambiguity, whether it be linguistic (such as the phonological ambiguity in a punning riddle) or contextual (such as riddles that manipulate social conventions to confuse the listener). Moreover, they claim that humor depends on that ambiguity being ‘unsolvable’ by the listener, at least until the punchline resolves it in some unexpected way.

Linguistic ambiguity can take place at the phonological, morphological, or syntactic levels of grammar. For example, the sentence “John lives near the bank” is phonologically ambiguous, since the noun “bank” can refer to either a building where money is stored, or the shore of a river.<sup>3</sup> The sentences “The book is read,” and “The book is red”, however, are morphologically ambiguous, since “read” is only phonetically identical with “red” in its past participle form. Finally, the sentence “John looked

over the car" is syntactically ambiguous, since it has two distinct grammatical analyses.

Each kind of ambiguity, or a combination, can be used in riddles. For example:

1. Phonological: What bird is lowest in spirits? A bluebird.
2. Morphological: Why is coffee like soil? It is ground.
3. Syntactic: Would you rather have an elephant kill you or a gorilla? I'd rather have the elephant kill the gorilla.

As can be seen from these examples, the ambiguity can occur either in the question (3) or the punchline (1 and 2). Other kinds of linguistic ambiguity — such as semantic and pragmatic ambiguities — can also be found in punning riddles, but our interest at present is in the three types listed above.

Pepicello and Green go on to describe many different strategies used in riddles to produce and manipulate these linguistic ambiguities. However, what all these strategies have in common is that they ask the 'riddlee' to accept a similarity on a phonological, morphological, or syntactic level as a point of semantic comparison, and thus get fooled. For example, the riddle:

Why is a river lazy? Because it seldom gets out of its bed. (Webb 1978)

uses the phonological ambiguity in the word "bed" to imply that a river bed is semantically identical with a sleeping bed, and therefore that not getting out of a river bed is a sign of laziness.

Using this analysis, we can choose a subset of riddles which share underlying properties (type of ambiguity, strategy for deceiving the riddlee), and which are thus able to be generated by closely related mechanisms. This is in contrast to jokes which have in common merely superficial aspects of their subject matter, for example jokes that ridicule a stereotyped group by describing how they might go about screwing in a lightbulb.

### *A source of data*

Most of the riddles discussed below come from *The Crack-a-Joke Book* (Webb 1978), a collection of jokes chosen by British children. This is an ideal source for several reasons. The riddles are simple, requiring only

basic English to understand; their humor generally arises from their punning nature, rather than their subject matter (children do not seem eager to joke about God, politics or taxes, and *The Crack-a-Joke Book* was undoubtedly edited for sex and toilet humor); and there are a huge number of riddles to choose from. For much the same reasons, however, *The Crack-a-Joke Book* riddles are unlikely to make an adult reader laugh. This is unfortunate, but unavoidable — it would be over-ambitious to tackle sophisticated adult humor at this stage.

*The Crack-a-Joke Book* includes some non-riddle humor, as well: limericks, dialogues, and punning book titles, for example. None of these types will be discussed below, as they are outside the range of this project.

Some of the jokes in the corpus come from sources other than *The Crack-a-Joke book*. If a given riddle comes from a book, it is cited; if there is no citation, then the riddle was remembered from casual conversation.

### *Punning riddles*

Riddles can be divided into groups in a variety of ways — by subject matter, narrative strategy, source, etc. We decided to select a subset of riddles which exploited linguistic ambiguity in a similar way. Although there are riddles that use two or more types of ambiguity, it is relatively straightforward to divide the bulk of question-answer riddles according to the primary level of ambiguity they use. Most of the riddles in Webb (1978) use phonological, morphological, or syntactic ambiguity. Of these three types, phonologically ambiguous riddles seem most directly amenable to formalization in a computationally tractable manner. From now on, riddles of this type will be called *puns*<sup>4</sup> or *punning riddles*.

There are three main strategies used in puns to exploit phonological ambiguity: syllable substitution, word substitution, and metathesis. This is not to say that other strategies do not exist; however, none were found among the large number of punning jokes examined.

*Syllable substitution.* Puns using this strategy confuse a syllable (or syllables) in a word with a similar- or identical-sounding word. For example:

What do short-sighted ghosts wear? Spooktacles. (Webb 1978)

The word containing the syllable which is to be replaced usually appears in the punchline of the riddle, while the question part of the riddle refers to some constructed ‘meaning’ (not the real meaning) of the word. However, the reverse can also occur:

What is an octopus? An eight-sided cat. (Webb 1978)

Sometimes, several word-syllable confusions are made:

Where are whales weighed? In a whaleweigh station. (Webb 1978)

Note that sometimes the confused syllable is actually replaced with a similar-sounding word (e.g. in “spooktacles” and “whaleweigh”), whereas other times the possible substitution is just referred to, for example “octopus” was not changed to “octopuss”, probably because the appearance of the changed word in the question would give the joke away.

Pepicello and Green have argued that some riddles of this type are in fact morphologically ambiguous, since a whole word is being confused with a morpheme, such as the “pus” suffix in “octopus”. However, jokes that use syllable substitution are very similar in structure, whether the syllable being confused is a morpheme or not, so we will regard them as essentially the same.

*Word substitution.* Word substitution is very similar to syllable substitution. In this strategy, an entire word is confused with another similar- or identical-sounding word. For example:

How do you make gold soup? Put fourteen carrots in it. (Webb 1978)

Again, the confused word can appear in the question instead of the punchline:

What do you do if you find a blue banana? Try to cheer it up. (Webb 1978)

Note that in the above joke, “blue” has two meanings, but only one surface form. This demonstrates that a word can be confused with: an alternate meaning (“blue”, the color, with “blue”, the mood); a word spelled differently but sounding the same (“carats” with “carrots”); or a word that sounds slightly different, as in:

Where do elves go to get fit? Elf farms. (Webb 1978)

The confused word is often part of a common phrase. For example, in: What sits in a fruit bowl and shouts for help? A damson in distress (Webb 1978) the joke relies on the riddlee recognizing the punchline as a warped version of the phrase “damsel in distress”.

*Metathesis.* Metathesis is quite different from syllable or word substitution. Also known as spoonerism, it uses a reversal of sounds and words to suggest (wrongly) a similarity in meaning between two semantically-distinct phrases. For example, this joke:

What's the difference between a torn flag and a bent sixpence? One's a tattered banner and the other's a battered tanner.

implies that tattered banners and battered tanners actually have a lot in common, the only difference being the reversed initial letters.<sup>5</sup> Words, as well as letters, can be reversed, for example:

What is the difference between a donkey and a postage stamp? One you lick with a stick, the other you stick with a lick. (Pepicello and Green, 1984)

Again, there is the implied similarity between donkeys and stamps, just because some of the actions you can perform on them are phonologically similar.

### *Chosen class of riddles*

All three of the above types of pun are potentially tractable for detailed formalization and hence computer generation. We chose to concentrate on word-substitution puns simply because lists of phonologically identical words (homophones) are readily available, whereas the other two types require some kind of sub-word comparison.

In particular, the class of jokes on which we have concentrated all:

1. have the substituted word in the punchline of the joke, rather than the question, and
2. substitute a homophone for a word in a common noun phrase.

These restrictions are simply to reduce the scope of the research even further, so that the chosen subset of jokes (punning riddles with noun-

phrase punchlines) can be covered in a comprehensive, rigorous manner. The model and method used to generate such jokes, however, should be general enough that it can be expanded in a straightforward way to cover all word-substitution puns, and — eventually — sub-word puns and spoonerisms.

## **Symbolic descriptions**

### *Introduction*

Now that we have restricted the target set of jokes to simple question-answer puns with noun-phrase punchlines, it is necessary to describe the common features of their structure in such a way that they could be constructed by a computer. These features must be specified well enough that any piece of text that has them is a joke. It is not good enough to say that there are some jokes among the pieces of text that share these features. In keeping with the remarks made earlier about linguistic methodology, we are trying to devise rules which will define as exactly as possible our chosen class of jokes, without some further intervention by human “intuition” or “common sense”. Although we expect that one way of extending our theory would be to state evaluative rules (heuristics) which rated the quality of jokes, we have no such mechanisms at present.

On the other hand, not all jokes (nor even all question–answer punning riddles) need have the stated features — to completely specify everything that could make text funny is too ambitious a task at present.

### *A model of simple question–answer riddles*

As mentioned in the previous section, one common way of creating a pun is word–word or word–syllable substitution. That is, take a valid English word or phrase, and substitute into it a word that is phonologically similar to part of the word/phrase. The riddle then uses the resulting nonsense word or phrase as if it were a semantically sensible construction, just because it is phonologically similar to a real word or phrase. The effective meaning of this fake construction is a combination of the meanings of the two pieces of text used to build it. For example:

What do you give an elephant that's exhausted? Trunkquillizers. (Webb 1978)

In this joke, the word “trunk”, which is phonologically similar to the syllable “tranq”, is substituted into the valid English word “tranquillizer”. The resulting fake word “trunkquillizer” is given a meaning, referred to in the question part of the riddle, which is some combination of the meanings of “trunk” and “tranquillizer”:

trunkquillizer: A tranquillizer for elephants.

Note that this is not the only meaning for “trunkquillizer” that could produce valid jokes. For example:

What kind of medicine gives you a long nose? Trunkquillizers.

is a joke (if not a good one) based on a different definition for “trunkquillizer” — namely, “a medicine that gives you a trunk”. The fake meaning should combine notable semantic features of both of the valid words/phrases used to construct it, so that the riddle question will be a reasonable description of the nonsense concept.

In non-humorous communication, we can use the meaning of a real word/phrase to build a question that has the word/phrase as an answer:

What do you call someone who douses flames? A firefighter.

Similarly, riddles like the *trunkquillizer* example use the constructed meaning of a *fake* word/phrase to build a question that *would* have the word/phrase as an answer, if it really existed. This question becomes the first part of the riddle, and the fake word/phrase becomes the punchline.

The building-blocks we have identified so far, are:

1. A valid English word/phrase;
2. The meaning of the word/phrase;
3. A shorter word, phonologically similar to part of the word/phrase;
4. The meaning of the shorter word;
5. A fake word/phrase, made by substituting the shorter word into the word/phrase;
6. The meaning of the fake word/phrase, made by combining the meanings of the original word/phrase and the shorter word.

The relationships between these building blocks are shown in Figure 1.

At this point, it is important to distinguish between the mechanism for building the meaning of the fake word/phrase, and the mechanism that

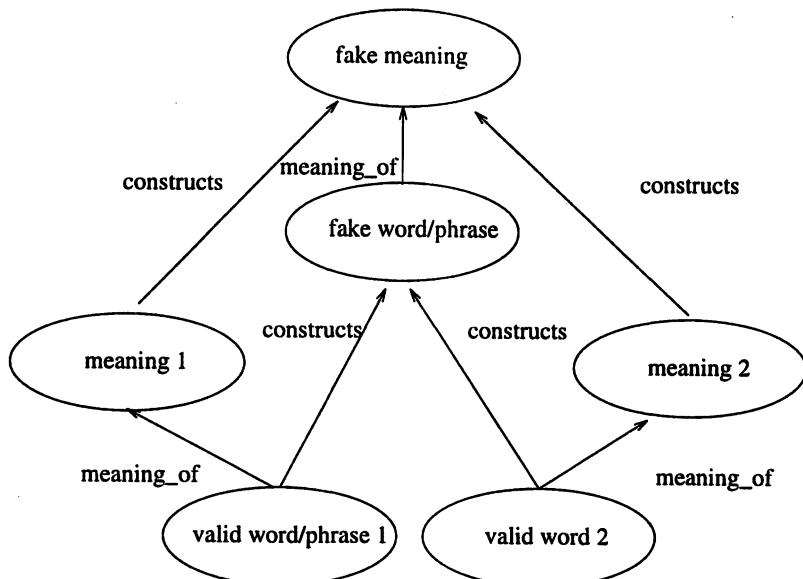


Figure 1. *The relationships between parts of a pun*

uses that meaning to build a question with the word/phrase as an answer. For example, the following questions use the same meaning for *trunkquilizer*, but refer to that meaning in different ways:

1. What do you use to sedate an elephant?
2. What do you call elephant sedatives?
3. What kind of medicine do you give to a stressed-out elephant?

On the other hand, these questions are all put together in the same way, but from different fake meanings:

1. What do you use to sedate an elephant?
2. What do you use to sedate a piece of luggage?
3. What do you use to medicate a nose?

We have adopted the term *schema* for the first sort of symbolic description (the underlying configuration of meanings and lexemes), and *template* for the second sort (textual rules or patterns used to construct an appropriate question-answer pair). Further details of these notions are given below, after we have summarized our lexical structures.

### *Lexicon*

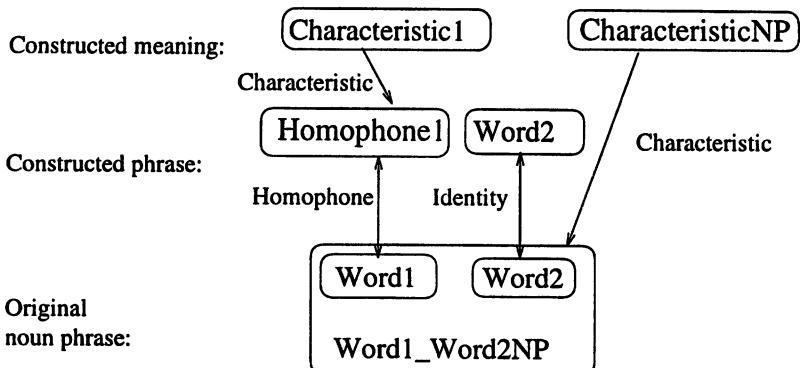
We have made some fairly conventional assumptions about the content and structure of the lexicon, as follows. There is a (finite) set of *lexemes*. A lexeme is an abstract entity, roughly corresponding to a meaning of a word or phrase. Each lexeme has exactly one entry in the lexicon, so if a word has two meanings, it will have two corresponding lexemes. Each lexeme may have some properties which are true of it, such as being a noun, and there are a number of possible relations which may hold between lexemes, such as synonym, homophone, and subclass. Each lexeme is also associated with a near-surface form which indicates (roughly) the written form of the word or phrase. We will deliberately blur the distinction between relations which are explicitly represented in the lexicon and other inter-lexeme relations which could be defined from the actual concrete entries. For example, one could define a relationship “superordinate-synonym” such that X is a superordinate-synonym of Y if there is a lexeme Z which is a superordinate of Y and Z is a synonym of X. We shall use the term *lexical relation* to denote any relation between lexemes, regardless of whether a given lexicon might store it explicitly.

The implemented computer program (described in the next section) represented the lexicon as fairly conventional data structures, but the details of those mechanisms are not part of the theory. In particular, the homophone relation was represented as a separate data structure, the *homophone base*, based on data collected from an independent source.

### *Schemata*

A schema stipulates a set of relationships which must hold between the lexemes used to build a joke. More specifically, a schema determines how real words/phrases are glued together to make a fake word/phrase, and which parts of the lexical entries for real words/phrases are used to construct the meaning of the fake word/phrase. Strictly speaking, the relationships are between sequences of lexemes, but in most cases the sequence will be of length one (a single lexeme); there are only a few examples so far where we need the flexibility to link to a longer sequence of lexemes.

There are many different possible schemata.<sup>6</sup> For example, the schema depicted in Figure 2 constructs a fake phrase by substituting a homophone for the first word in a real phrase, then builds its meaning from

Figure 2. *The lotus schema*

the meaning of the homonym and the real phrase. Another schema might construct a fake phrase by substituting a homophone for the second word in a real phrase, then constructing its meaning from the first word in the phrase and the homophone. (This is, in fact, exactly what the *jumper* schema does.) Here we shall present schemata diagrammatically. The boxes enclose lexemes, or sequences of lexemes. An arrow from one box to another indicates that there is some relation between the lexemes in the two boxes, with double-headed arrows used for symmetric relationships. The labels on the arrows (links) are the names either of particular relations (for example, identity) or of some set of possible relations (for example, characteristic).

The schema shown in Figure 2 is uninstantiated: that is, the actual lexemes to use have not yet been specified. Moreover, in some places the exact choice of relations is still unspecified, with a set of possible relationships between the lexemes being stated. Instantiating a schema means choosing lexemes (or sequences of lexemes) to go in the schema, and specifying the exact relationships between those lexemes. (Slightly more formally, an uninstantiated schema is a set of variables and a set of constraints [links] between those variables, where constraints indicate relations that must hold between any values assigned to those variables.) In our initial implementation, we have found a need for only three types of link:

**Homophone:** The linked lexemes are homophones (whether spelled the same or not), as specified in the homophone base (see the Homonym Base subsection, below). This is a specific relation.

Table 1. Example of lexical relations

Relation	Value
class	vegetable
location	garden
action	grows
adjective	green
etc.	etc.

Identity: The linked lexemes are identical, and therefore have exactly the same entry in the lexicon. This is a specific relation.

Characteristic: There is some relationship between the lexemes, defined in terms of (but not necessarily explicitly represented in) their lexical entries. Hence this is a set of possible relations, defined in terms of possible lexical connections (see the Lexicon subsection, below).

For example, in the lexicon, the lexeme *spring-cabbage* might participate in relations as in Table 1.

If *spring-cabbage* were to be included in a schema, at one end of a characteristic link, the other end of the link could be associated with any of these values (vegetable, garden, etc.), depending on the exact label (class, location, etc.) chosen for the characteristic link. When the schema is fully instantiated, however, both the characteristic link and the lexemes it joins are specified.

The completely instantiated *lotus* schema in Figure 3 could (with an appropriate template — see the Templates subsection, below) be used to construct the joke:

What's green and bounces? A spring cabbage. (Webb 1978)

### *Templates*

A schemata (see above) defines a configuration of lexical items which can be linked, semantically, within a riddle. There must also be some mechanism which turns this configuration into a piece of text in ordinary language. In the narrow class of riddles we are considering, this text is a question and an answer, the latter consisting of a noun phrase. We have no proposals for sophisticated mechanisms which could carry out such a mapping. In our preliminary model, we have adopted a rather ad hoc

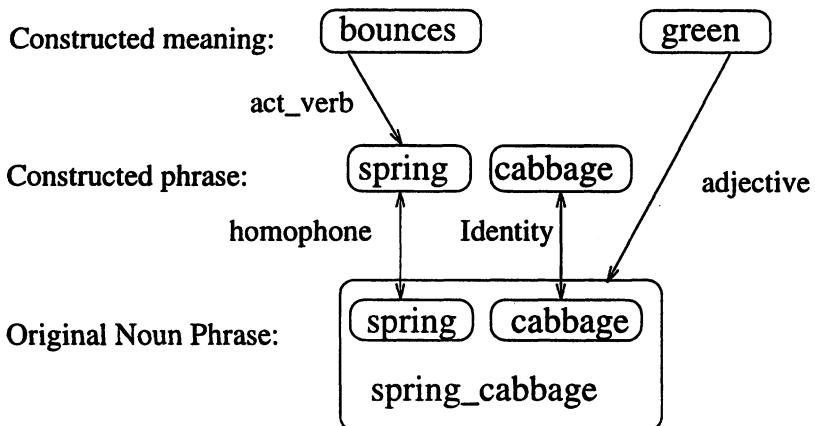


Figure 3. *A completely instantiated lotus schema*

method for transforming a related set of lexemes into a suitable question-answer pair. Essentially, we have a collection of stereotyped riddle forms, which can be thought of as fixed pieces of texts containing slots to be filled by words or phrases provided by the riddle schemata. These fixed text patterns, together with information about how to fill the slots, are referred to as templates.

More precisely, a template indicates the following information:

1. Any fixed segments of text that are to be used, such as ‘what do you call a \_\_\_?’ ‘what do you get when you cross \_\_\_ with \_\_\_?’ In our riddles, these are all in the “question” part.
2. Which lexemes from the schemata are to be used to fill which blank slots in the template.
3. Syntactic details of how these lexemes are to be expressed, for example as a singular noun phrase.

There is no essentially humorous knowledge encoded in the set of templates. They are a (rather crude) mechanism which takes a set of lexemes and formulates a question and an answer about them. The fact that there is anything amusing or bizarre about the question–answer pair is caused by the choice of lexemes related in particular ways, and this is the responsibility of the schema, not the template. If one supplied some lexemes not related in a humorous way, then a template should still produce a well-formed (but non-humorous) text. For example, given

suitable items, with appropriate lexical definitions,<sup>7</sup> some template might generate:

What kind of feline brings bad luck? A black cat.

There is an issue here which we believe needs much more exploration, namely the quantity of information which must or must not appear in the riddle question. For example, consider the joke:

What man claps at Christmas? Santaplause. (Webb 1978)

If the text-building mechanism provided too little information, this joke might become:

What man claps? Santaplause.

which fails as a joke. Similarly, if there was too much information, the text might be:

What bearded man in a red suit applauds at Christmas? Santaplause.

This is better, but still inferior to the original. Intuitively, it seems to be important to have the right balance of descriptive information. We have no theoretical solution to this problem. Instead we have built this into the humorous knowledge that our model has, by stipulating that certain schemata will work only with certain templates (or vice versa). Some schemata construct “obscure definitions” (definitions that do not strongly evoke the words in the constructed noun phrase), and so require templates that provide more information in the questions; whereas other schemata construct very clear definitions, and so require templates that do not “give the joke away”. For this reason, every schema has to be associated with a set of appropriate templates. To be more precise, part of the knowledge which defines possible riddles within our theory is a mapping which associates each instantiated schema with a set of appropriate templates for expressing it.<sup>8</sup>

### *Conclusion*

In this section, we have developed a simple model for question–answer punning riddles which use substitution as their main mechanism. A punning riddle is viewed as the surface manifestation of an instantiated

schema and an associated template (with its variable portions filled from the schema). A schema defines the constraints on lexemes which will produce possible riddles.

## The JAPE-1 computer program

### *Introduction*

We have implemented the model described above in a computer program called J<sub>A</sub>P<sup>E</sup>-1 (Joke Analysis<sup>9</sup> and Production Engine). J<sub>A</sub>P<sup>E</sup>-1 is significantly different from other attempts to computationally generate humor in three ways: its lexicon is humor-independent, in other words, the structures that generate the riddles are distinct from the semantic and syntactic data they manipulate; it generates riddles that are similar on a strategic and structural level, rather than merely in surface form; and, finally, J<sub>A</sub>P<sup>E</sup>-1 is an implementation of a model of riddles, so its success or failure is of theoretical, as well as practical, interest.

Although our model could be applied to punning riddles in general, we chose to concentrate on a subgroup. These riddles all:

1. Use word-word substitution as their main mechanism;
2. Substitute phonologically identical (as opposed to phonologically similar) words;
3. Substitute into a noun phrase; and,
4. Use the phrase thus constructed in the punchline (as opposed to the question) part of the riddle.

These restrictions are chosen largely to reduce the number of schemata and templates required, so that the important factors in J<sub>A</sub>P<sup>E</sup>-1's performance do not get lost in the noise. Also, the easy availability of homophone lists makes word-word substitution schemata easier to implement than those using word-syllable substitution or metathesis (both of which use sub-word divisions). Although these restrictions may seem excessive, they still describe approximately 100 of the jokes in *The Crack-a-Joke Book* (Webb 1978).

*Worked example.* In this section, as an illustrative example, we work through how J<sub>A</sub><sup>P</sup>E-1 reproduces one of *The Crack-a-Joke Book* riddles, namely:

What do you get when you cross a sheep and a kangaroo? A woolly jumper.

### *Top level*

J<sub>A</sub><sup>P</sup>E-1's main mechanism attempts to construct a punning riddle based on a common noun phrase. It has several distinct knowledge bases with which to accomplish this task:

1. a set of six schemata (see the Schemata subsection, below);
2. a set of fourteen templates (see the Templates subsection, below);
3. a lexicon, which contains humor-independent semantic and syntactic information about the words and noun phrases entered in it (see the Lexicon subsection, below);
4. a homophone base, which contains pairs of phonologically identical lexemes (see the Homophone Base subsection, below).
5. a post-production checker, which applies some simple heuristics to sift out some of the more obvious non-jokes (see the Post-Production Checking subsection, below)

As noted in the Symbolic Descriptions section above, the homophone base is abstractly part of the lexicon, since it merely embodies one more relationship between lexemes, but it was computationally convenient to separate it out.

At the top level, J<sub>A</sub><sup>P</sup>E-1 chooses an appropriate schema and an associated template, the choice of template is constrained by the choice of schema, and the choice of template in turn specifies the content of the underspecified links in the schema. It instantiates the schema using information from the lexicon and the homophone base, and fills the template, again using information from the lexicon. Filling a template produces a riddle in near-surface form, which is tested by the post-production checker.

It is not inherent in J<sub>A</sub><sup>P</sup>E-1's design that these stages of joke production happen in any particular order, although some choices do constrain others. In fact, in this implementation, the noun phrase is chosen first (see below for details of the algorithm). Also, J<sub>A</sub><sup>P</sup>E-1 optionally allows the noun phrase, the schema, and/or the template to be specified by the

user, so that the program may be given as much, or as little, guidance as wished.

### *The J<sub>A</sub><sup>P</sup>E-1 lexicon*

The lexicon contains the lexemes used in constructing the jokes, and syntactic and semantic information about them.

A lexeme is a symbol unique to one semantic interpretation of a word. Each entry in the lexicon has one, and only one, corresponding lexeme, which in turn is associated with a near-surface form. The near-surface form of a word can be associated with several different lexemes, and thus several lexicon entries.

A near-surface form is a piece of text (a word, phrase, sentence, or complete riddle) in grammatical, understandable English; however, it is not “pretty printed” (it may not have capitals at the beginning of sentences, etc.). In this implementation, a near-surface form is a list of lower-case words. This is for programming convenience only — once a riddle has been generated and checked, it is changed into surface form by the program.

The purpose of the lexicon is to store general (not humor-oriented) syntactic and semantic information. Although this lexicon was constructed specifically for J<sub>A</sub><sup>P</sup>E-1, the information contained in it is general and neutral — the joke-generating power lies elsewhere in the program, particularly in the schemata and their ties to suitable templates.

Although the lexicon stores syntactic information, the amount of syntax used by the rest of the program is minimal. Because the templates are based on common riddle forms, such as “What do you get when you cross \_\_\_ with \_\_\_ ?”, whole sentences need not be constructed “from scratch”. For this reason, the only necessary syntactic information has to do with the syntactic category, verb person, and determiner agreement. Also, the lexicon need only contain entries for nouns, verbs, adjectives, and common noun phrases — other types of word (conjunctions, determiners, etc) are built-in to the templates. Moreover, because the model implemented in J<sub>A</sub><sup>P</sup>E-1 is restricted to covering riddles with noun phrase punchlines, the schemata require semantic information only for nouns and adjectives.

Each lexeme can be considered to be a node in a network, linked to other lexemes in the network via the semantic slots in its lexical entry.

The values in these semantic slots should be other lexemes with entries in the lexicon. Syntactic slots, on the other hand, contain syntactic information, not lexemes (see Table 2 for more details on the available syntactic and semantic slots).

Although the values in the semantic slots should be other lexemes, in some cases (HAS, ACT\_OBJ, LOCATION, and USED\_TO\_OBJ) a semantic slot takes a near-surface form instead. This is because we are interested in J<sub>A</sub><sup>P</sup>E-1 as an implementation of a model of riddles, rather than as a generator of syntactically-complex sentences. In order to avoid complex (but uninteresting) syntactic generation, the values in some semantic slots are chunks of text (words put together grammatically in near-surface form) instead of lexemes, so that they can be put directly into a template without further syntactic manipulation. For example, the entry for the lexeme *lion* has, as its LOCATION slot value, the near-surface form of “in the jungle”, rather than the lexeme *jungle*.

*Worked example.* The phrase “woolly jumper” has the following entry, where italicized entries indicate lexemes:

```
lexeme = woolly_jumper
category = np
written_form = “woolly jumper”
comp_lex = woolly, jumper_1
vowel_start = no
countable = yes
class = sweater
inact_verb = wear
```

Someone else defining the word could have provided slightly different semantic slot values (a British word-definer would probably not have used the word “sweater”) or filled in more slots. However, any slot values which fit the descriptions in table 2 should produce recognizable jokes.

“Woolly”, an adjective, has the following entry:

```
lexeme = woolly
category = adj
written_form = “woolly”
vowel_start = no
describes_all = sheep
synonym = fuzzy
```

Again, another definer could make a slightly different entry, which would produce a different end joke.

Table 2. Lexicon slots

Slot	Used with	Allowed values
<b>Syntactic slots</b>		
Category	all entries	NP, noun, adj, verb
Written_form	all entries	The near-surface form of the lexeme. For nouns, this is taken by convention to be the singular form, and for verbs, the infinitive
Vowel_start	NP, noun, adj	yes or no (does the near-surface form of the lexeme start with a vowel?)
Second	verb	The near-surface, second-person form of the verb
Third	verb	The near-surface, third-person form of the verb
Comp_lex	NP	A list of the lexemes that make up the noun phrase
Countable	NP, noun	yes or no (is the noun or NP countable?)
<b>Semantic slots</b>		
Class	NP, noun	The immediate superclass of the lexeme (for <i>lemon</i> , <i>fruit</i> )
Spec_is	NP, noun	A lexeme that, when used to qualify the class lexeme, defines the entered lexeme reasonably precisely (for <i>lemon</i> , <i>citrus</i> )
Is	NP, noun	A lexeme that typically describes the entered lexeme (for <i>lemon</i> , <i>sour</i> )
Has	NP, noun	Part(s) of the thing to which the entered lexeme refers, in near-surface form. Should fill “It has ____.” (for <i>lemon</i> , “ <i>pips</i> ”)
Act_verb	NP, noun	A verb lexeme. Something the thing typically does (for <i>chef</i> , <i>cook</i> )
Act-obj	NP, noun	The near-surface form of the object of the ACT_VERB value (for <i>chef</i> , “ <i>food</i> ”)
Inact_verb	NP, noun	A verb lexeme. Something you typically do to the thing (for <i>horse</i> , <i>ride</i> )
Location	NP, noun	The near-surface form of its typical location (for <i>horse</i> , “in a pasture”)
Used_to	NP, noun	A lexeme. Something the thing is typically used to do (for <i>spatula</i> , <i>flip</i> )
Used_to_obj	NP, noun	The near-surface form of the object of the USED_TO value (for <i>spatula</i> , “ <i>pancakes</i> ”)
Synonym	NP, noun, adj	A lexeme of the same category as the entered lexeme, which has a similar entry — in particular, a lexeme’s synonym’s synonym is itself (for <i>pillow</i> , <i>cushion</i> )
Describes_all	noun, adj	A lexeme which refers to a thing or class of things which can (almost) always be described by the entered lexeme (for <i>slimy</i> , <i>worm</i> )

Finally, the word “jumper” has two entries, one for each meaning:

```
lexeme=jumper_1
category=noun
written_form="jumper"
vowel_start=no
countable=yes
class=clothing
spec_is=warm
synonym=sweater
```

```
lexeme=jumper_2
category=noun
written_form="jumper"
vowel_start=no
countable=yes
describes_all=kangaroo
act_verb=leap
```

Each of the lexeme values in the semantic slots of the above entries has an entry of its own.

### *Homophone base*

The homophone base is simply a list of homonym (different spelling) pairs and alternate-meaning (same spelling) pairs. These lexemes are available to be used as the ends of a homophone link, when instantiating a schema (see the Schemata subsections, above).

Homonyms are words that are phonologically identical, but have different spellings. Homonyms should also have different meanings; that is, spelling variants, such as “humour” and “humor”, are not considered to be homonyms. An alternate-meaning pair, on the other hand, is a pair of lexemes that have identical near-surface forms, but different semantic entries (sometimes known as homographs). For example, the lexeme *sole\_1*, which refers to a kind of fish, and *sole\_2*, an adjective synonymous with “only”, are alternate meanings; however, *shower\_1*, a light rain, and *shower\_2*, a bathroom device, are not, since they both refer to water falling from above. In J<sub>A</sub>E-1 the entries for the two lexemes must be completely different for them to be an alternate-meaning pair.

$J_A^P E-1$ 's homophones are from a list (Townsend and Antworth 1993) of homophones in American English, which has been shortened considerably for our purposes. Removed from the list were:

1. pairs including a proper noun (e.g. "Cain" and "cane")
2. pairs including an obscure word (e.g. "buccal" and "buckle")
3. pairs which are not homonyms in British English (e.g. "balm" and "bomb")
4. pairs which were alternate spellings of the same word (e.g. "humor" and "humour")
5. pairs including words which are neither adjectives nor nouns
6. pairs including abstract nouns
7. pairs whose meanings are often confused (e.g. "acclamation" and "acclimation")
8. pairs differing mainly in their syntactic category (e.g. "bare" as adjective and as verb)

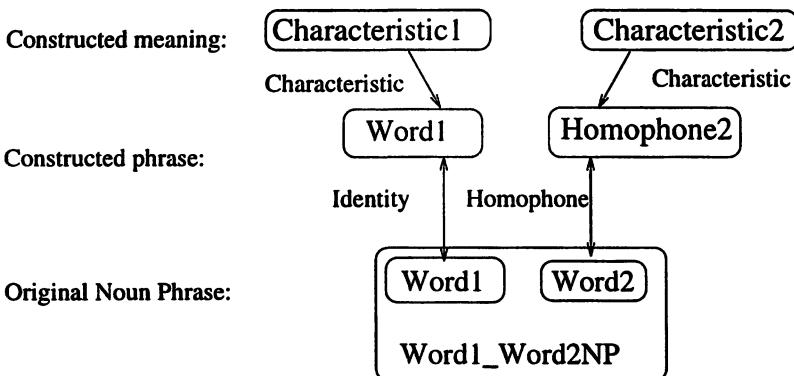
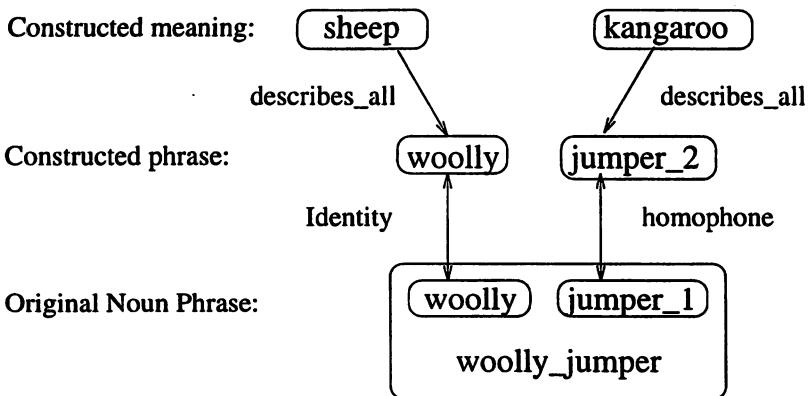
This version of  $J_A^P E-1$  treats homonym pairs and alternate-meaning pairs as equivalent, since they seem to play the same role in the type of riddle we are trying to reproduce. However, one can imagine types of jokes in which they are not equivalent, so they have been stored separately within the homophone base. We shall take the liberty of referring to both homonyms and alternate meanings as "homophones" from now on.

*Worked example.* The two lexemes *jumper\_1* and *jumper\_2* are entered in the homophone base as an alternate-meaning pair, since the semantic parts of their entries (see the example lexicon entries, above) are completely different.

### *Schemata in $J_A^P E-1$*

The *jumper* schema<sup>10</sup> shown in Figure 4 is uninstantiated, in that the actual lexemes to be used to construct the joke have not yet been specified, and some of the relationships (the characteristic links) between the lexemes are still very general. The schema in Figure 5 is the same schema, completely instantiated.

These three links (characteristic, identity, homophone) are all that is required to specify a wide range of riddle schemata. However, if  $J_A^P E-1$  were to be expanded to handle other kinds of joke, more links would have to be added. For example, if  $J_A^P E-1$  had to produce spoonerisms, a *rhymes* link (at least) would be necessary.

Figure 4. *The uninstantiated jumper schema*Figure 5. *The instantiated jumper schema, using the noun phrase "woolly jumper" and the syn-syn template. Used to generate the joke: What do you get when you cross a sheep and a kangaroo? A woolly jumper*

Characteristic links indicate that the lexemes chosen for the schema must be linked via some relation based on the lexical entries. The simplest values for characteristic links, therefore, are the actual slot labels used in the lexicon:

- |               |                  |
|---------------|------------------|
| 1. inact_verb | 7. location      |
| 2. synonym    | 8. describes_all |
| 3. class      | 9. spec_is       |
| 4. is         | 10. has          |
| 5. used_to    | 11. used_to_obj  |
| 6. act_verb   | 12. act_obj      |

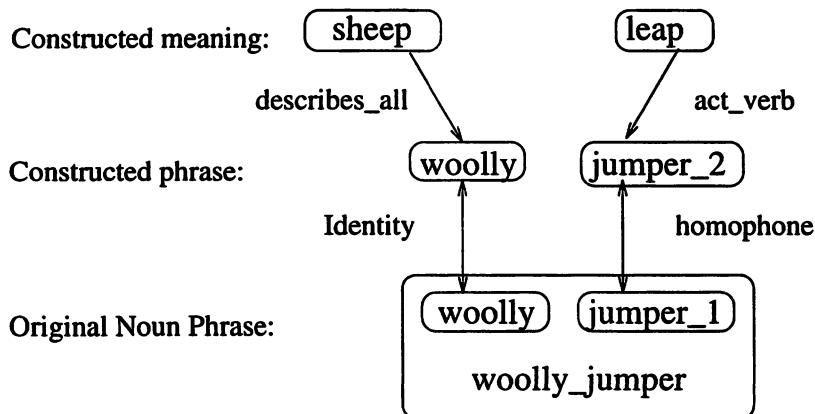
Characteristic links can also join a lexeme to a concatenation of lexemes. For example, one template uses a sequence of two semantic slot values, namely the *spec\_is* and *class* entries. This link would join the lexeme *jumper\_1* to the two lexemes *warm* and *clothing*. The information about what lexemes to concatenate is held in the template, and is tied to the exact choice of relation made for the characteristic link.

We have found it useful, in describing J<sub>A</sub><sup>P</sup>E-1's mechanisms, to make a distinction between two sorts of lexemes within a schema. There are *key lexemes*, which can be instantiated early on in the process because either they are part of the genuine noun phrase, or they are linked to such lexemes by well-specified links such as Identity. There are also *characteristic lexemes*, which are linked only by Characteristic links, and so cannot be chosen until the template is chosen.

The characteristic links in a schema are instantiated as soon as a template is chosen, and any lexemes still uninstantiated in the schema (the characteristic lexemes) are inserted as the template is filled. However, it is important to distinguish between the pieces of text generated by the template, and the lexemes in an instantiated schema, because it is the lexemes which are checked for suitability later (see the Post-Production Checking subsection, below).

*Worked example.* The example we are working through uses the *jumper* schema, as shown in Figure 4. In Figure 5, an instantiated *jumper* schema, the key lexemes have been slotted in appropriately. Also, both of the characteristic links have been instantiated to *describes\_all*, and the characteristic lexemes have been instantiated to *sheep* and *kangaroo*, from the lexical entries for *woolly* and *jumper*.

It is important to understand that this particular instantiation of the *jumper* schema is a result of two factors: the chosen noun phrase (*woolly-jumper*), and the chosen template (in this case, *syn\_syn*), described further in the subsection below on templates. The choice of template determines how the characteristic links are specified, and thus the values for the characteristic lexemes. For example, if a different template, *syn\_verb* (described in the next subsection), were chosen, the *jumper* schema would be instantiated in a slightly different way — one of the characteristic links would be instantiated to *act\_verb* instead of *describes\_all*, and thus that characteristic lexeme would be *leap* instead of *kangaroo* (see Figure 6).



**Figure 6.** The instantiated jumper schema, using the noun phrase “woolly jumper” and the syn-verb template. Used to generate the joke: What do you call a sheep that can leap? A woolly jumper

## *Templates in J<sub>A</sub><sup>P</sup>E-1*

As described in the Symbolic Descriptions section, a template constructs a suitable question-answer pair from the information contained in a schema. The question uses the "meaning" of some nonsense phrase to refer to that nonsense phrase in the answer, or punchline. Both the nonsense phrase and its "meaning" are built by the instantiation of a schema. Riddles tend to use one of a limited number of common forms; for example, questions of the form "What do you get when you cross \_\_\_\_ with \_\_\_\_ ?" come up quite often.  $J_A^P E-1$  exploits these common forms, in order to cut down on the syntactic information required to generate a grammatical surface-form riddle.  $J_A^P E-1$ 's templates have two important parts:

1. Specifications for the underspecified links in the schema;
  2. Near-surface form question-answer pairs containing blanks, which are to be filled by near-surface form text fragments. These text fragments are generated by the template, from the lexemes in the instantiated schema.

Also, where a particular choice of relation involves instantiating a variable to a particular sequence of lexemes, that will be indicated. For example, the *syn syn* template can be stated informally as:

Use relations from the set `{spec_is_class, describes_all, synonym}` to instantiate the characteristic links; for `spec_is_class`, form a sequence from the two lexemes that are linked to the key lexeme by `spec_is` and by `class`.

What do you get when you cross [text fragment generated from the first characteristic lexeme(s)] with [text fragment generated from the second characteristic lexeme(s)]?

[the near-surface form of the constructed noun phrase].

The `syn_verb` template can be summarized:

Instantiate first characteristic link to an element of `{spec_is_class, describes_all, synonym}`, for `spec_is_class`, form a sequence from the two lexemes that are linked to the key lexeme by `spec_is` and by `class`; instantiate second characteristic link to `act_verb`

What do you call [text fragment generated from the first characteristic lexeme(s)] that [text fragment generated from the second characteristic lexeme(s)]?

[the near-surface form of the constructed noun phrase].

In this implementation, the text fragments are generated using a simple Prolog definite-clause grammar (Bratko 1990), which simply puts the lexemes together in a syntactically correct near-surface form.

*Worked example.* If the `syn_syn` template is chosen, then the `jumper` schema is instantiated to the schema in Figure 5. Then, the information in that instantiated schema is given to the `syn_syn` template, described above, which generates the near-surface form of:

What do you get when you cross a sheep and a kangaroo? A woolly jumper.

Similarly, if `syn_verb` is the chosen template, the `jumper` schema is instantiated to the schema in Figure 6. The information in that instantiated schema is then used by the `syn_verb` template to generate the near-surface form of:

What do you call a sheep that can leap? A woolly jumper.

### *Post-production checking*

Although J<sub>A</sub>P<sub>E</sub>-1 generates pieces of text that are recognizably jokes, they are not all good jokes (see the Evaluation section, below, for a formative evaluation of J<sub>A</sub>P<sub>E</sub>-1's performance). It is clear that some kind of post-production heuristic checking is needed, to put the jokes in order of

quality, and perhaps remove those at the bottom of the pile. At present, only two such checks have been implemented. The first is that none of the lexemes used to build the question and punchline are accidentally identical; that is,  $J_A^P E-1$  checks that lexemes in an instantiated schema are identical only if linked by an identity link. This is to prevent “jokes” like the following:

What do you get when you cross a sheep and a jumper? A woolly jumper.

Such a “joke” might arise from loops in the lexicon — entries referring to each other in an unexpected way.

The second post-production check is that the lexemes used to build the nonsense noun phrase (the punchline) do not build a genuine common noun phrase. This is to prevent  $J_A^P E-1$  from accidentally coming up with sensible question–answer pairs, such as:

What do you call a cylindrical drink container? A Coke can.

It is difficult to imagine why  $J_A^P E-1$  would produce such a piece of text; however, it is clearly not a joke, and should be sifted out.

### *JAPE's basic algorithm*

In this design, there are several points at which a choice has to be made, a schema instantiated, or a template filled.  $J_A^P E-1$  tackles these tasks in the following order:

1. Choose a common noun phrase from the lexicon.
2. Choose an appropriate schema, into which the phrase can fit.
3. Fit the phrase's constituent lexemes into the schema.
4. Instantiate the key lexemes in the schema.
5. Choose an appropriate template, thus instantiating the characteristic links.
6. Generate the near-surface form of the riddle, instantiating characteristic lexemes in the process.
7. Check that the generated joke is neither repetitive, nor a sensible question–answer pair.

However, this exact ordering is a feature of the implementation, rather than the underlying model.

### **Factors likely to influence JAPE-1's performance**

In  $J_A^P E-1$ 's design as it stands, there are places open for the introduction of heuristics (rules of thumb) for improving joke quality, both during joke production and the post-production sifting of  $J_A^P E-1$ 's output. No such heuristics were introduced before testing, so that they would arise from an evaluation of  $J_A^P E-1$ 's unfiltered output, rather than pre-run guesswork. Nonetheless, it was possible, prior to the evaluation, to make certain predictions about  $J_A^P E-1$ 's performance which, if confirmed, could give rise to heuristics in later versions of the program. Some of the suggested heuristics would order the output of the program; others would trim the output, by reducing the number of jokes produced.

All of these conjectures were made after the lexical data was gathered, but before  $J_A^P E-1$  was run to produce the set of jokes to be judged (see the Evaluation section, below, for details of the methodology).

#### *The lexicon and the homophone list*

It was anticipated that the quality and content of the lexicon would largely determine the quality of the jokes produced.

Each of the steps in  $J_A^P E-1$ 's joke production rely on the lexicon being clear, precise, accurate, and compatible with  $J_A^P E-1$ 's templates and schemata. For this reason, it was anticipated that the following requirements would be important:

1. Semantic information should be included in the lexicon only if it is typical of the word being entered. For example, dogs do sleep, but if the entry for "dog" includes that information, then  $J_A^P E-1$  is likely to ask questions like "What sleeps and ... ", expecting "dog" to be among the concepts evoked in the mind of the reader.
2. The information in the lexicon should be common knowledge. This applies both to the words chosen to be entered, and the entries they are given. It would be no use  $J_A^P E-1$  making a joke about a horse's frog, if most people do not know that "frog" can refer to part of a horse's foot, as well as to a green amphibian.
3. Jokes should use concrete words, such as "wooden", and "cat", rather than abstract words like "happiness", "attitude". This is because the constructed concept is more likely to be funny if it can be visualized. For example, the idea of a "toe truck" probably has more humor potential than that of an "optical illusion".

4. Jokes should avoid using very general words like “structure”, “substance”, “object” because such words have a huge number of possible instances. For example, although a Buddhist monk is indeed a person, the word “person”, without any other information, is unlikely to bring the image of a Buddhist monk to mind — so a joke depending on that evocation would probably fail.
5. Homophone pairs should not be easily confused, in other words the distinctness of their meanings should be common knowledge). For example, a pun based on the homophones “aural” and “oral” would probably not work very well, as these words are often misspelled and misused.

### *Choice of schemata*

Some schemata substitute only one homophone; others substitute homophones for both the words in the phrase. The more homophone links in the schema, the more information should be available for the template to use in the question part of the riddle. It was anticipated that any schema which did not provide enough information for the riddle to “solve” both homophone-substitutions, would therefore produce incomprehensible jokes.

On the other hand, a schema which both uses two homophone-substitutions, and provides the template with a great deal of information about the “meaning” of the nonsense phrase was expected to produce the best jokes. However, it would also produce the fewest jokes, for the same reason — few common noun phrases have constituent words which both have homophones, and few of our lexicon entries contain enough semantic information to fill this schema.

It was suspected that schemata which rely on good, clear lexicon entries for noun phrases would produce slightly worse riddles, as such entries are difficult to make (because, in general, noun phrases describe more complex concepts than individual words).

If a schema may create confusion between the meaning of the head noun of the punchline and the meaning of the whole punchline noun phrase, this may give poor results.

The above criteria could be used to impose an order on the schemata indicating which were most promising.

### *Choice of template*

The central part of the punchline, in all templates, is a noun phrase consisting of a noun and a modifier. Since the word order of the punchline

does contain some semantic information, namely, which of its words is the object and which word describes that object, it is important for the question to reflect that information. For example, if the punchline is to be “lemon aide”, then the question should describe a kind of assistant, rather than a kind of citrus fruit; that is,

What do you call a sour assistant? A lemon aide.

is a more coherent riddle than:

What do you call a helpful fruit? A lemon aide.

The characteristic order in the question need not be the same as that in the punchline, but it should describe a type of whatever the second word is. For example, the *use\_syn* template (“What do you use to \_\_\_ a \_\_\_?”) could be used in non-joking communication as follows:

What do you use to fight a fire? A fire fighter.

A coherent joke should have a similar structure, for example:

What do you use to wash a harness? A bridle shower.

rather than the reverse:

What do you use to steer drizzle? A bridle shower.

For this reason, it was anticipated that the templates which conformed to this pattern would be more successful than templates which use the lexemes in the opposite order.

We expected that another important factor in the success of a template would be the amount of information a question generated by the template would provide about the punchline. If it were to provide too much, the punchline would be given away; if it were to provide too little, the joke would be incomprehensible.

### *Schema-template pairs*

It was anticipated that certain schemata would work well with some templates, and not so well with others. It was difficult to predict which would be the successful pairs; however, we expected the distinction to be clear. In fact, we predicted that the elimination of certain schema-template pairings would improve the data considerably.

### *Post-production checking*

At the time of testing, the only post-production checks in J<sub>A</sub><sup>P</sup>E-1 were that the punchline of the riddle was not a real noun phrase, (not in the lexicon), and that none of the key lexemes used to build the riddle were accidentally identical (they were identical only if linked by an identity link). Other less obvious checks were not included in pre-evaluation J<sub>A</sub><sup>P</sup>E-1 in the hope that the evaluation would reveal their heuristic power so that they could be added later.

Some good heuristics were expected to be:

**Question length:** Questions that are too long seem unwieldy; questions that are too short don't provide enough information to suggest the punchline.

**Alliteration and rhyming:** Punchlines made up of alliterative or rhyming words are often more successful. Spoonerisms and related types of joke rely largely on this effect for their humor.

**“Funny letters”:** It is a commonly-held belief among stand-up comedians that certain letters (particularly “k”, “q”, “v”, “w”, and “z”) are inherently funny — of a pair of synonyms, the one containing the most such letters is funnier. This may be a myth; on the other hand, it may explain why one joke is a little bit funnier than another similar one.

**Subject matter:** Jokes that have “funny” subjects (sex, religion, politics, etc.) are generally funnier than similar jokes about more mundane issues.

**Accidental associations:** It was expected that some jokes produced by J<sub>A</sub><sup>P</sup>E-1 would be funny in unforeseen ways. For example, the question part of the riddle could be syntactically ambiguous, thus making the punchline a more humorous answer; or a semantic link not included in the lexicon could be suggested by an accidental juxtaposition of words in the riddle.

Obviously, some of these potential heuristics are easier to implement than others. For example, if accidental associations were easy to spot, they would not be accidental — they would be included in J<sub>A</sub><sup>P</sup>E-1's lexicon, schemata, or templates. Also, none of these post-production heuristics could absolutely determine whether or not a given joke is funny; however, they could be used to partially order J<sub>A</sub><sup>P</sup>E-1's output in terms of quality.

### **The evaluation procedure**

The purpose of this formative evaluation (Mark and Greer 1993) was primarily to point the way to improvements in the theory behind the

project, and directions for further research. It was not intended to be a rigorous examination of the “humor value” of J<sub>A</sub>P<sup>E</sup>-1’s output, for two reasons. Firstly, it would be a project in and of itself to design a rigorous measure of the humor content of text jokes. Secondly, J<sub>A</sub>P<sup>E</sup>-1 should be seen as an initial exploration, rather than a final product. Although a statistically correct test would be valuable, it would require a huge amount of data and a large number of volunteers — effort that would be wasted when some of the more obvious flaws in J<sub>A</sub>P<sup>E</sup>-1’s design were fixed.

There are two main stages to the evaluation: data acquisition and joke judging. During the data acquisition stage, volunteers unfamiliar with J<sub>A</sub>P<sup>E</sup>-1 were asked to make lexical entries for a set of words given to them. These definitions were then sifted by a “common knowledge judge”, and entered into J<sub>A</sub>P<sup>E</sup>-1’s lexicon. In the second stage, the jokes produced by J<sub>A</sub>P<sup>E</sup>-1 from these sets of words were then judged by a different group of volunteers. Their opinions were then compared to the hypotheses made above.

#### *Lexical data acquisition stage*

As our chosen class of riddles depended crucially on the use of homophones, it was essential that our lexicon contained a good range of homophone pairs. We therefore began by establishing a list of homophones, as described in the Homophone Base section above. Common noun phrases which contain at least one word on the homophone list were then added to the list of lexemes to be defined. Finally, the other words used in the common noun phrases were added to the list. Twenty-one noun phrases and fifty-nine words were then on the list, adding up to eighty lexemes needing to be defined.

Volunteers were each given a set of lexemes and instructions on how to define them in accordance with the specification of J<sub>A</sub>P<sup>E</sup>-1’s lexicon (see Table 2 earlier). In their instructions, it was emphasized that they should provide only typical, specific information about the lexemes they were to define.

After the lexical data had been collected, it was discovered that some of the volunteers had not followed the instructions, and had tried to fill every available slot in each entry, or left some words undefined. A certain amount of correction or filling out was therefore carried out according to a precise set of guidelines, by a further volunteer, not in any way

involved with J<sub>A</sub><sup>P</sup>E-1 (the “common-knowledge judge”). After the entries had been amended in this way, they were entered into J<sub>A</sub><sup>P</sup>E-1’s lexicon. J<sub>A</sub><sup>P</sup>E-1 then produced a set of 188 jokes in near-surface form.

### *Joke judging stage*

The jokes produced by J<sub>A</sub><sup>P</sup>E-1 were put into surface form and distributed in groups to “joke judges”, who rated jokes both absolutely (on a scale of 1 to 5) and relatively as to how good they were as jokes. They were also asked for qualitative information, such as how the jokes might be improved, and if they had heard any of the jokes before.

### *Problems with methodology*

As stated above, this is merely an informal exploration, rather than a rigorous evaluation. The following are some of the limitations of what was done.

A statistically sound analysis requires a lot of data. A lot of data, in this case, requires a lot of volunteers, and a lot of time to analyze the data. During this project, neither was available. Unfortunately, this dearth of data means that it is very difficult to draw conclusions from the data we do have. For example, if a certain schema-template pairing produces only one joke, due to the small size of the lexicon, it is impossible to decide if that joke (and its score) is typical of that pairing.

Another problem was the lack of a control group. We suspect that jokes of this genre are not very funny even when they are produced by humans; however, we do not know how human-produced jokes would fare if judged in the same way J<sub>A</sub><sup>P</sup>E-1’s jokes were, so it is difficult to make the comparison.

The volunteer joke judges were probably not a typical sample of the ordinary population. Due to the method of distribution of questionnaires (e-mail to friends and associates), all the volunteers were adults with regular access to a computer. Moreover, all the volunteers were either comedians or involved in artificial intelligence and so had expertise in one aspect or another of this project. This undoubtedly influenced their judgment.

Two factors were mentioned several times by volunteers as being reasons for giving low scores to jokes: “joke fatigue”, from reading too many similar jokes, and a lack of enthusiasm for this genre of joke.

Ideally, with hindsight, the lexicon entries would have been taken from some standard lexicon. The resulting jokes would then have been mixed with similar jokes (from Webb 1978, for example), and then all the jokes would have been judged by a group of schoolchildren, who would be less likely to have heard the jokes before and more likely to appreciate them.

### **Analysis of results**

#### *Summary*

Most of the jokes were given a score of 1, glossed in the questionnaire as “a joke, but a pathetic one”, but there was a great deal of variation, with some jokes receiving maximum scores (“Really good”) and some being rated as off the scale (“Not a joke; doesn’t make any sense”).

On the other hand, our hypotheses were, for the most part, correct, and the results show that implementing even the simplest of these hypotheses would improve the resulting jokes significantly. Moreover, the top end of the output (those jokes that would survive the implementation of trimming heuristics) are definitely of *The Crack-a-Joke Book* quality, including:

What do you call a murderer that has fibre? A cereal killer.

What kind of pig can you ignore at a party? A wild bore.

What kind of emotion has bits? A love byte.

#### *Lexicon effects*

It is clear from the results (in particular, from the comments) that the quality of the lexicon greatly influenced the quality of the resulting jokes. Although we tried to ensure that the volunteer word definers understood and followed the lexicon requirements, and our “common knowledge judge” attempted to sift out inappropriate data, several “bad” entries (or parts thereof) survived to be used in joke construction. It is difficult to trace exactly what went wrong in a failed joke; however, the judge’s

comments suggest that many of the flaws arose from entries in the lexicon that did not meet the criteria described above:

1. Semantic information should be included in the lexicon only if it is typical of the word being entered. With reference to the rather poor joke:

What do you use to sniff a drilling tool? A wild bore.

one judge asked, “Does ‘to wild’ mean ‘to sniff’?” clearly trying to understand what sniffing has to do with the punchline. It came from the entry for “boar”: the definer wanted to express the idea that boars can be used to sniff for truffles. Unfortunately, sniffing as such is not typical of boars, and so does not bring them to mind at all.

2. The information in the lexicon should be common knowledge. This cropped up several times. Regarding these two jokes:

What do you use to clothe a coniferous tree? A fir coat.

What do you call a passenger ship you can drink? Ferry liquid.<sup>11</sup>

one judge remarked: “... the fir coat one is funny because most people would know what a coniferous tree is ... whereas the ferry liquid [joke] isn’t very obvious.”

A related problem was that the judges were from all over the world, and dialects differ. A Canadian judge said that there were “... too many ‘Britishisms’ not common to North American English, for example stag, hart, and queue.” A British judge pointed out an “Americanism”: “The bridal shower jokes rely on knowing that a wedding party is called a bridal shower in North America.”

3. Jokes should use concrete words ... rather than abstract words. The only abstract words included in this lexicon were “love” and “number”, neither of which caused any problems. In fact, the jokes they were used in got above average scores. The difficulty anticipated with abstract words was that, in general, they do not evoke strong images the way more concrete words do. Perhaps “love” and “number” are evocative enough to avoid this problem.
4. Jokes should avoid using very general words. The lexicon, once it had been checked and amended, did not contain any very general words.

### *Schemata effects*

As predicted, there was not a lot of difference in the performance of the three best schemata. For jokes using these schemata, the important factors affecting their scores were the choice of template and/or noun phrase used to build the joke. One of the other schemata did not produce very good results, as expected, because there is often some confusion over which words are referring to which concepts. This does not mean, however, that this schema should be discarded, but rather that the templates with which to pair it must be carefully chosen, so that the ‘meaning’ of the pun is clearly expressed.

The schema which made two substitutions and gave little information about the described concept did poorly, as expected. This suggests that a joke based on two homophone substitutions into a noun phrase needs more information to be provided in the question part of the joke, if it is to be comprehensible.

The other double-substitution schema, which we predicted would produce the fewest, but best, jokes, failed to produce any jokes at all. This is due to the fact that not only does the schema require a noun phrase into which two homophones can be substituted, but also its only compatible template requires several lexemes to be fully instantiated, and there was insufficient lexical data to achieve this.

### *Template effects*

The templates varied a great deal in the quality of jokes they produced.

The two best templates were (informally) “What do you get when you cross a ... with a ... ?” and “What do you call a ... ?”. This is, we suspect, because they provide the right amount of information in the question — not so much that the punchline is given away, and not so little that the joke is incomprehensible.

Certain templates fared much better than similar templates that used the lexemes provided by the schemata in the opposite order, as predicted, apart from the “what do you call ... that ... ?” template, in which the order seemed not to be significant.

A general comment that was made several times was that some of the questions were not “logically coherent” in some sense. This incoherence

was often the result of using one of the templates with bad word ordering. For example, the joke:

What kind of ship can clean dishes whilst caring for your hands? Ferry liquid.<sup>12</sup> received the comment “[This joke] lost points for saying ‘what kind of ship,’ and then having an answer that wasn’t a ship,” and a poor average score. The corresponding reversed template with better word ordering produced the following joke with the same schema and noun phrase:

What kind of detergent can cross water? Ferry liquid.

is still not brilliant, but it is more logically coherent, and received a medium score.

Only one template, (a variant of “What do you use to ... ...?”) was uniformly poor. This is largely because the order of the lexemes is of particular importance in questions of that form. The template which used the same textual material, but put the items in the reverse order did quite well.

### *Schema-template pairings*

If we were to eliminate the schemata and templates that always produced poor jokes, the jokes would undoubtedly improve; however, we would still be left with a lot of poor jokes. A large number of these can be removed by severing the link between certain schemata and templates, so that no jokes will be produced that use both. However, it is difficult to make a principled decision about which schema-template links to sever on the basis of such a small sample.

### *Quality of checking*

Although none of the joke ratings can be interpreted as supporting the hypotheses on post-production checking, several of the general comments do:

Question length: “[The criteria used to judge the jokes were] in order of importance: gut reaction, cleverness, delivery, rhythm. The rhythm and delivery [of these jokes] could be improved. They have to be ‘bang-bang’ jokes.”

Alliteration and rhyming: “Phrasing is important. Numbers aren’t funny, but ‘quirky quantifier’ is.” This comment was referring to the joke:

What do you call a quirky quantifier? An odd number.

“Funny letters”: “The grizzly bare joke [What do you call a ferocious nude? A grizzly bare] makes good use of the funny letter Z.”

Subject matter: “I don’t find ‘gay’ jokes funny.” This comment was referring to the joke:

What do you call a homosexual that cleans dishes whilst caring for your hands? Fairy liquid.

Accidental associations: “Keep the ones with the double meanings, for example [the one about] a gay being drunk.” This was with reference to the joke:

What do you call a homosexual you can drink? Fairy liquid.

The judge who made this comment gave the joke in question a high score and commented on the “obscene” image it brought to mind, while the other judge for this joke rated it “not a joke”.

### *Jokes heard before*

Perhaps one of the most heartening results of this evaluation is that several of the judges claimed to have heard similar or identical jokes before. This can be taken as evidence that J<sub>A</sub>P<sub>E</sub>-1 is producing some good examples of the punning riddle genre.

Four judges claimed to had heard these J<sub>A</sub>P<sub>E</sub>-1 jokes, all variations on the “cereal killer” pun:

What do you call a murderer that has fiber? A cereal killer.

What do you get when you cross grain with a murderer? A cereal killer.

What do you call breakfast food that murders people? A cereal killer.

Other J<sub>A</sub>P<sub>E</sub>-1 jokes that had been heard before included:

What kind of tree can you wear? A fir coat.

What kind of rain has presents? A bridal shower.

What do you call a good-looking taxi? A hansom cab.

What do you call a perforated relic? A holey grail.

What do you get when you cross a savage pig with a drilling tool? A wild bore.

Another interesting point was that, well after the evaluation had taken place, it was noticed that J<sub>A</sub><sup>P</sup>E-1 had produced two jokes very similar to ones in *The Crack-a-Joke Book*. J<sub>A</sub><sup>P</sup>E-1's jokes were:

What kind of pig can you ignore at a party? A wild bore.

and

What do you use to help a lemon?<sup>13</sup> Lemon aid.

*The Crack-a-Joke* versions were:

What runs around forests making other animals yawn? A wild bore.

and

What do you give a hurt lemon? Give it lemonade, of course.

### *Conclusions*

This evaluation has accomplished two things. It has shown that J<sub>A</sub><sup>P</sup>E-1 can produce pieces of text that are recognizably jokes (if not very good ones) from a relatively unbiased lexicon. More importantly, it has suggested some ways that J<sub>A</sub><sup>P</sup>E-1 could be improved:

1. The description of the lexicon could be made more precise, so that it is easier for people unfamiliar with J<sub>A</sub><sup>P</sup>E-1 to make appropriate entries. Moreover, multiple versions of an entry could be compared for “common knowledge”, and that common knowledge entered in the lexicon.
2. More slots could be added to the lexicon, allowing the person entering words to specify what a thing is made of, what it uses, and/or what it is part of.
3. New, more detailed templates could be made so that some of the more complex schemata would produce comprehensible jokes.
4. Templates which use the lexemes given to them in the “wrong” order (an order that suggests the words in the punchline should be reversed) could be removed.
5. The remaining templates could be adjusted so that they use the lexical data more gracefully, by providing the right amount of information in the question part of the riddle.

6. Schema-template links that give consistently poor results could be removed.

It is not easy to see how some of the other suggestions for improvement could be implemented in J<sub>A</sub>P<sup>E</sup>-1 as it stands. However, these ideas could be incorporated into later, more complex systems:

1. Improve the delivery and rhythm of the jokes.
2. Attempt to make the punchline alliterate or rhyme when possible.
3. Maximize the number of “funny letters”.
4. Stick to inherently funny subject matter.
5. Generate jokes, then test them for serendipitous associations.

If even the simplest of the trimming and ordering heuristics described above were implemented, J<sub>A</sub>P<sup>E</sup>-1’s output would be restricted to good-quality punning riddles.

## **Related work**

### *The General Theory of Verbal Humor*

Although many studies have been done on the language of humor (Chiari 1992) and (Booth 1974), few have looked at the linguistics of humor in any depth. This is not to denigrate the work that has been done in the field; however, in order for the computer generation of riddles to be a tractable problem for symbolic AI, a detailed, formal, linguistic model of (at least a subset of) humor is required at some point. The only model we found that even approaches the level of formality required is Salvatore Attardo and Victor Raskin’s General Theory of Verbal Humor (GTVH) as described and developed in Attardo and Raskin (1991), Attardo and Raskin (1994), and Ruch, Attardo and Raskin (1993).

The General Theory of Verbal Humor is an attempt by Attardo and Raskin to build a linguistically sound model of verbal humor.<sup>14</sup> By analyzing the similarities and differences of a set of variants on a light-bulb joke, Attardo and Raskin postulate six joke parameters, or knowledge resources (KR), which between them determine the final text form of the joke. These KRs are organized into a hierarchy.

*Script opposition.* The script opposition KR is based on Raskin’s earlier script-based semantic theory of humor (SSTH), which he summarizes as follows:

A chunk of structured semantic information, the script can be understood for the purposes of this article as an interpretation of the text of a joke. The main claim of STH is that the text of a joke is always fully or in part compatible with two distinct scripts and that the two scripts are opposed to each other in a special way. (Attardo and Raskin 1991)

The “special ways” in which scripts can be opposed are at various levels of abstraction, such as, real vs. unreal, good vs. bad, high stature vs. low stature, nondumb vs. dumb, etc.

Joke 1: How many Poles does it take to screw in a light bulb? Five. One to hold the bulb and four to turn the table he's standing on.

Joke 1 uses the nondumb vs. dumb opposition, since it is about applying a stupid method to a simple task which most people deal with in a simple, intelligent fashion.

*Logical mechanism.* This parameter determines the mechanism used to oppose the scripts. For example, joke 1 uses figure-ground reversal (from *Gestalt* psychology, according to Attardo and Raskin). Usually, when screwing in a light bulb, the room, the ladder or table, the person screwing in the bulb, etc. stay still, while the light bulb moves; joke 1 reverses that situation.

Holding the other parameters of joke 1 constant, but changing the logical mechanism to ‘false analogy’, we get:

How many Poles does it take to screw in a light bulb? Five. One to hold the light bulb and four to look for the right screwdriver.

Other such mechanisms include simple reversal, false priming, simple juxtaposition, and “the juxtaposition of two different situations determined by the ambiguity or homonymy in a pun” (Attardo and Raskin 1991).

Note that, in the “joke telling mode of communication”, the truth of statements and their consistency become less important. The pseudologic of the joke, therefore, need not be valid, just vaguely persuasive — persuasive enough that the listener will go along with the joke.

*Situation.* The situation of a joke is the set of details (time, place, objects, activity, etc.) which specify the joke. A given script opposition and logical mechanism can be applied to a number of different situations.

How many Poles does it take to wash a car? Two. One to hold the sponge and one to move the car back and forth.

This joke differs from joke 1 only in situation.

*Target.* The target of a joke, the person or stereotype the joke is aimed at, is the only optional parameter of the six. Many jokes have no identifiable target. The target of joke 1 is, clearly, Poles, but it could be changed to almost any other group which is stereotyped as “stupid” (blondes, Newfies, etc.).

*Narrative strategy.* This KR determines the form the joke will take, such as riddle, conundrum, expository text, etc. The more standard strategies have the advantage that the punchline pretty much automatically falls in the right place. Also, the choice of logical mechanism limits the range of narrative strategies available.

Joke 1 as expository text, rather than conundrum, might look like this:

It takes five Poles to screw in a light bulb: one to hold the light bulb and four to turn the table he’s standing on.

*Language.* This parameter specifies which paraphrasing of the joke is used (what the surface form of the joke is). It is constrained by all the other parameters. For example, although the language parameter determines the exact phrasing and placement of the punchline, all the other parameters (particularly narrative strategy and logical mechanism) have a lot of input into it as well.

#### *Relevance of GTVH to this project*

By providing a parameterized model of verbal humor in general, Attardo and Raskin have provided a rough structure which could, in part, guide the design of a humor producing program.

In particular, they note that holding some of their parameters constant produces a joke “template”.<sup>15</sup> If we choose a few of the better defined parameters to keep variable, and hold the rest constant, we should have a constrained model of (certain types of) humor which could, in theory at least, be used as the basis for a program design.

Unfortunately, their model is neither detailed nor formal enough to be implemented as it stands, even in a constrained, “template” form.

Some of their “knowledge resources,” in particular the script-opposition and logical mechanism KRs, require a near-complete understanding of the world (including the rules of physics, the operations of human society, and common-sense reasoning) in order to operate. Even their language KR “includes all the choices at the phonetic, phonologic, morphophonemic, morphologic, lexic, syntactic, semantic, and pragmatic levels of language structure that the speaker is still free to make” as well as “a few specifically humorous elements and relations” (Attardo and Raskin 1991). In order for computer implementation of this model to be feasible, it must be severely constrained, perhaps so much as to be unrecognizable.

Nevertheless, our model could be seen as a shell produced by constraining or holding constant some of Attardo and Raskin’s parameters. In terms of their six knowledge resources, our model could be described as follows:

*Language.* The surface form of the joke is determined by the template. The choice of template is constrained by the choice of schema, and the instantiation of the schema provides the template with information it needs to construct the surface form.

*Narrative Strategy.* The narrative strategy is fixed, since these jokes are all, by definition, question-answer riddles.

*Target.* There is no explicit target in any of the set of aimed-for riddles, although it is possible that some object of ridicule might be accidentally incorporated. If the lexicon were adjusted to include stereotypical (instead of typical) semantic links, then jokes with butts would probably arise more often.

*Logical Mechanism.* The logical mechanism is fixed for noun-phrase punning riddles: the juxtaposition of the meaning of a phrase, with the meaning of a word which puns part of the phrase. The different (uninstantiated) schemata could be seen as slight variations on this basic mechanism. Other riddle types would have different, but fixed, logical mechanisms. For example, a spoonerism riddle’s mechanism could be described as the juxtaposition of the meaning of a phrase with its spoonerism.

*Situation.* The situation is variable, though “choosing the situation” really means choosing words/concepts which are consistent with the logical mechanism (instantiating the schema).

*Script opposition.* In our class of riddles, there is no strong evidence for the two scripts being opposed — they need only be different. If there is any script opposition at all in punning riddles, it is fixed, real vs. unreal.

The model described in this paper, then, is consistent with Attardo and Raskin’s GTVH, although most of their parameters are either fixed or constrained. The substantive features of our model correspond to three GTVH parameters: an uninstantiated schema corresponds to a particular logical mechanism; instantiating the schema corresponds to choosing a situation; and filling a template produces the surface-form language of a joke. However, our model does not emphasize the same features of joke structure as Attardo and Raskin’s theory. In particular, Attardo and Raskin hold that script opposition is absolutely essential for a piece of text to be considered a joke; whereas it is not even represented in our model. Either script opposition is not actually very important in question–answer punning riddles, or one script opposition is implicit in and fixed for all such jokes.

### *Other joke generators*

We are aware of very few other attempts to generate jokes using a computer.

*The light bulb joke generator.* Attardo and Raskin have put together a simple joke generating program, LIBJOG (Light Bulb Joke Generator) (Attardo and Raskin 1994), mainly to show how poorly simple cut-and-paste methods work. The first version combines an entry for a commonly stereotyped group, for example:

- (i) (Poles ((activity1 hold the light bulb)
   
          (numberX 1)
   
          (activity2 turn the table he is standing on)
   
          (numberY 4)))

with a template (in their sense of this term) for a light bulb joke:

How many (group name) does it take to screw in a light bulb? (NumberX). One to (activity1) and (numberY) to (activity2). [Condition:  $X=1+Y$ .]

This results in, not surprisingly:

How many Poles does it take to screw in a light bulb? Five. One to hold the light bulb and four to turn the table he's standing on.

Although Attardo and Raskin claim that later versions of LIBJOG "introduced more templates, more fields, and looser (and richer) relations among them," they give no evidence of a significantly improved method. The joke-generating mechanism seems to remain the same: substitute the (humor-related) values in an entry for a stereotyped group, directly into a light-bulb joke template like the one above.

$J_A^P E-1$  differs from LIBJOG in several significant ways:

1.  $J_A^P E-1$  produces a wider range of riddles, which do not have the fixed surface format of light-bulb jokes.
2.  $J_A^P E-1$  is an implementation of a model of humor, albeit a very simple one, rather than a program that can produce jokes in an uninteresting way. Although  $J_A^P E-1$  uses a lexicon and templates, there is more to its method than simply pasting the two together.
3.  $J_A^P E-1$ 's lexicon is humor-independent, containing the kind of information one might expect to find in a dictionary (albeit in a different form); whereas LIBJOG's "lexicon" holds entries on commonly-stereotyped groups (Poles, blondes, etc.), with pieces of text describing stereotypical ways of screwing in a light bulb.
4.  $J_A^P E-1$  produces jokes which share deep traits, such as type of ambiguity and relationships between key words; whereas LIBJOG produces lightbulb jokes and only lightbulb jokes.

For the above reasons,  $J_A^P E-1$ 's successes and failures are of theoretical interest, and improvements in  $J_A^P E-1$  amount to refinements of a model of (a subset of) humor. Improvements in LIBJOG, on the other hand, can only be cosmetic, since it does not generate jokes in accordance with any particular theory.

*De Palma and Weiner.* Some of the issues involved in jokes such as:

What has a mouth and cannot eat?  
A river.

are discussed by (De Palma and Weiner 1992). That work is very close in spirit and intention to ours, with its concentration on formalizing the underlying semantic patterns in a very restricted subclass of punning riddle, but their chosen type of riddle is very different from ours. They have embodied the joke-creating knowledge in an algorithm which searches a knowledge base for suitably related but homophonous items, so it is not clear (from the relatively brief account) how they separate (if at all) knowledge about jokes from general knowledge.

They give no details of how much has been implemented and tested.

## **Future directions**

### *Overall evaluation of project*

We have shown that for one particular (non-trivial) subclass of question-answer punning riddles it is possible to formulate a precise and computationally tractable symbolic model which describes a wide range of jokes in a systematic way.

We have implemented a computer program ( $J_A^P E-1$ ) to produce jokes using this model, and have tested this using an unbiased, humor-independent lexicon whose contents were supplied by a set of volunteers.

We have informally evaluated the jokes generated in this way by having them assessed by a set of human judges.

The results showed that the current version of  $J_A^P E-1$  does generate good jokes (including several that had been heard and recorded before). Unfortunately, it also generates many bad jokes, and a few non-jokes. Trimming the program, by systematically removing poor lexical entries, templates, and schemata, would improve the output significantly; adding filtering heuristics would improve it even more, probably eliminating the non-jokes entirely. Also, in hindsight, we realized that human-made punning riddles should have been mixed with  $J_A^P E-1$ 's output to be judged, so that a fair comparison could have been made.

Although there is certainly room for improvement in  $J_A^P E-1$ 's performance, it does produce recognizable jokes in accordance with a model of punning riddles, which has not been done successfully by any other program we know of. In that, it is a success. Moreover, the model

underlying  $J_A^P E-1$  is general enough that it can be expanded straightforwardly to cover a wider variety of jokes, such as sub-word puns and spoonerisms.

The model that  $J_A^P E-1$  uses was influenced by Pepicello and Green's work on the language of riddles (Pepicello and Green 1984) and Attardo and Raskin's General Theory of Verbal Humor (Attardo and Raskin 1991), but it does not adhere strictly enough to either work to be considered evidence for or against their theories.

### *Possible extensions and improvements*

We have already discussed two broad areas of possible improvement that could be made to  $J_A^P E-1$ , namely trimming parts of the program and adding some filtering heuristics. In addition, the program could be extended to handle other joke types. At present,  $J_A^P E-1$  can generate only question–answer punning riddles with noun-phrase punchlines. One obvious direction of expansion would be to allow different kinds of punchline, for example:

Why is it dangerous to play cards in the jungle?  
Because there are so many cheetahs. (Webb 1978)

Although such jokes could be modeled with the same kind of schema that we already have, the templates and the lexicon would have to be much more complex syntactically, in order to generate appropriate question–answer pairs.

Another possible direction of expansion would be to attempt to generate simple spoonerisms and sub-word puns. Again, these could be modeled with schemata similar to the present ones, albeit with different link types (e.g. *rhymes\_with* instead of *homophone\_of*, for spoonerisms). The templates would have to be different from, but not more complex than, the templates  $J_A^P E-1$  currently uses.

Even if all the extensions described above are carried out,  $J_A^P E-1$  will still only be able to generate phonologically ambiguous question–answer riddles, which are just a small subset of humorous texts.

*University of Edinburgh*  
*g.d.ritchie@ed.ac.uk*

## Notes

KB is grateful for financial help from Canada Student Loans, the Overseas Research Students Scheme, and the St Andrew's Society of Washington, DC. We would like to thank Salvatore Attardo for his comments and for giving us access to his unpublished work.

1. A talk on this work was given at the International Conference on Humor and Laughter, Luxembourg, 1993, and a short version appeared in (Binsted and Ritchie 1994).
2. This subsection is essentially a precis of chapters two and three of (Pepicello and Green 1984), and the examples are theirs.
3. This kind of ambiguity is more commonly known as “lexical ambiguity”, but we retain the terminology from (Pepicello and Green 1984).
4. Note that other works may use the word “pun” differently.
5. A “sixpence” is an obsolete British coin, and “tanner” was a slang name for that coin. We apologize for the age and cultural bias of some of our examples.
6. The various schemata acquired obscure labels, with little mnemonic value, in the course of the project; these have no particular significance.
7. Culture-specific of course — to some people, black cats are lucky.
8. This association is, strictly speaking, between templates and an instantiated schema, not between templates and the skeletal schema which defines the possible patterns of lexemes. This is because the precise choice of relations for the vaguely specified links (“characteristic” links) will affect the appropriateness of a template. Conversely, one could say that the choice of template influences the choice of particular relation for the links (this is in fact how it is implemented).
9. “Analysis” in the sense that it is an implementation of a model —  $J_A^P E-1$  does not actively parse or analyze previously made jokes in any way.
10. Although it is not accidental that this schema’s name is one of the words in the punchline of the worked example, it is just a mnemonic. The jumper schema can, of course, generate other jokes, as we later demonstrate.
11. “Fairy Liquid” is a well-known brand of detergent in Britain.
12. Hand care is an advertized feature of “Fairy Liquid”.
13. This riddle came up during  $J_A^P E-1$ ’s development, not during the evaluation.
14. This section is essentially a precis of (Attardo and Raskin 1991). Their examples are used.
15. Attardo and Raskin use this term in a different way from the particular definition used in our model of riddles.

## Appendix

### *Some sample output*

These are some of the  $J_A^P E-1$ -generated jokes that were given good ratings by the human judges.

What kind of device can you get on at the airport? An aeroplane hanger.  
 What do you get when you cross a sad emotion with a stag? A broken hart.  
 What kind of contest can you drive on? A duel carriageway.  
 What do you call a coniferous mink? A fir coat.  
 What do you use to clothe a coniferous tree? A fir coat.  
 What do you get when you cross a coniferous tree with a jacket? A fir coat.  
 What do you get when you cross a passenger ship with dishwashing detergent?  
 Ferry liquid.  
 What do you call a passenger ship that cleans dishes whilst caring for your hands?  
 Ferry liquid.  
 What do you call dishwashing detergent that crosses water? Ferry liquid.  
 What do you use to wash up a homosexual? Fairy liquid.  
 What do you call a passenger ship you can drink? Ferry liquid.  
 What do you call a homosexual you can drink? Fairy liquid.  
 What do you call a ferocious nude? A grizzly bare.  
 What do you get when you cross a ferocious bear with a nude? A grizzly bare.  
 What kind of emotion has bits? A love byte.  
 What do you call a reverent quantifier? An awed number.  
 What do you call a quirky quantifier? An odd number.  
 What do you call a reverent numeral? An awed number.  
 What do you call a worshipper that is used to count? An awed number.  
 What kind of line has sixteen balls? A pool queue.  
 What do you use to prompt billiards? A pool cue.  
 What kind of pig can you ignore at a party? A wild bore.  
 What do you call a savage drill? A wild bore.  
 What do you call a flat fish that inherits everything? A sole heir.  
 What do you call a flat fish that inherits your money? A sole heir.  
 What kind of murderer has fiber? A cereal killer.  
 What do you get when you cross grain with a murderer? A cereal killer.  
 What do you call a breakfast murderer? A cereal killer.  
 What do you call grain that murders people? A cereal killer.  
 What do you get when you cross breakfast food with a murderer? A cereal killer.

## References

- Attardo, Salvatore and Victor Raskin  
 1991 Script theory revis(it)ed: joke similarity and joke representation model.  
*HUMOR: International Journal of Humor Research*, 4, 293–347.
- Attardo, Salvatore and Victor Raskin  
 1994 Non-literalness and non-bona-fide in language: An approach to formal and computational treatments of humor. *Pragmatics and Cognition*, 2, 31–69.
- Binsted, Kim and Graeme Ritchie  
 1994 An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)* Seattle, USA.  
 Available as Research Paper 690, Department of Artificial Intelligence,  
 University of Edinburgh.

- Booth, Wayne  
 1974 *A Rhetoric of Irony*. Chicago: University of Chicago Press.
- Bratko, Ivan  
 1990 *Prolog Programming for Artificial Intelligence*. Wokingham: Addison-Wesley.
- Buchanan, Bruce G.  
 1988 Artificial intelligence as an experimental science. In Fetzer, J. (Ed.), *Aspects of Artificial Intelligence*. Dordrecht: Kluwer. 209–250.
- Chiari, Delia  
 1992 *The Language of Jokes: Analysing Verbal Play*. London: Routledge.
- Chomsky, Noam  
 1957 *Syntactic Structures*. The Hague: Mouton.
- Chomsky, Noam  
 1965 *Aspects of the Theory of Syntax*. Cambridge: MIT Press.
- De Palma, Paul and E. Judith Weiner  
 1992 Riddles: accessibility and knowledge representation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92)*, Vol. 4, 1121–1125.
- Derks, Peter  
 1984 The use of humor rating scales. In *Proceedings of the Fourth International Conference on Humor* Williamsburg, VA. Department of Psychology, College of William and Mary.
- Derks, Peter L., Dana L. Lewis and Ralph O. White Jr.  
 1981 A comparison of ratio and category scales of humour. *Canadian Journal of Behavioural Science. Revue canadienne des sciences du comportement*, 13, 226–237.
- Mark, Mary A. and Jim E. Greer  
 1993 Evaluating methodologies for intelligent tutoring systems. *AI and Education: Special Issue on Evaluation*, 4, 129–153.
- Minsky, Marvin  
 1963 Steps towards artificial intelligence. In Feigenbaum, E. and J. Feldman (Eds.), *Computers and Thought*. New York: McGraw-Hill, 406–450.
- Newell, Allen and Herbert Simon  
 1976 Computer science as an empirical inquiry: symbols and search. *Communications of the ACM*, 19, 113–126.
- Pepicello, W. J. and Thomas A. Green  
 1984 *The Language of Riddles*. Columbus: Ohio State University.
- Ritchie, Graeme  
 1994 Learning from AM. In Johnson, J., S. McKee, and A. Vella (eds.), *Artificial Intelligence in Mathematics*, Oxford: Oxford University Press, 55–66.
- Ruch, Willibald, Paul McGhee and F.-J. Hehl  
 1990 Age differences in the enjoyment of incongruity-resolution and nonsense humor during adulthood. *Psychology and Aging*, 5, 348–355.
- Ruch, Willibald, Salvatore Attardo and Victor Raskin  
 1993 Toward an empirical verification of the general theory of verbal humor. *Humor: International Journal of Humor Research*, 6(2), 123–136.
- Townsend, William and Evan Antworth  
 1993 Handbook of Homophones (online version). 1993. Available by FTP as follows — machine name: *svr-ftp.eng.cam.ac.uk*, directory: *Comp.speech/data*, file name: *homophones-1.01.txt*.
- Webb, Kaye (Ed.)  
 1978 *The Crack-a-Joke Book*. London: Puffin.