THE COMPUTATIONAL GENERATION OF A CLASS OF PUN

by

CHRISTOPHER VENOUR

A thesis submitted to the

Department of Computing and Information Science

in conformity with the requirements for

the degree of Master of Science

Queen's University

Kingston, Ontario, Canada

September, 1999

Abstract

This document examines what makes texts funny and explores how sophisticated a computer program and database would need to be in order to produce jokes of varying complexity. A precise model of a class of pun is then created and implemented into a program that can generate jokes. By doing so, we show that some forms of punning can be codified and thus puncture some of the mystique surrounding humour. Our analysis of humour also reveals some new fundamental building blocks that will be required for comprehensive language generation and understanding.

Acknowledgments

I would like to thank my supervisor Dr. Michael Levison for his guidance in the creation of this thesis. I would also like to thank my family who has always supported me in my schooling and in all things. Thank you also to my family at St. James for their support and interest in me. A special thanks to Mary whose gentle and kind ways are a blessing to me and all who know her. And finally I would like to thank my brother-in-law Michael without whom this thesis would not have been completed. I am very grateful to him for all his kindness and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Humour and Computational linguistics

This thesis describes methods for automatically generating a type of pun: the "homonym common phrase pun" or "HCPP". A lot of research into humour has explored "why we make jokes or what they mean" or "what aspects of jokes are common to particular cultures and what aspects transcend culture" [LL93]. In her dissertation Kim Binsted writes: "Although a great deal of work has been done on humour in psychology, literature and sociology, less has been done in linguistics, and little in AI or computational linguistics" [BR94, 12]. When linguists have considered the subject of humour, they often resort to informal taxonomy, classifying jokes in terms of the kind of linguistic play involved, but only in vague and general terms (see [Chi92], [Ho77], [Cu88]). Most of their research ignores or only superficially analyses the detailed mechanisms of humour. Computational linguists, however, must understand these mechanisms before any progress can be made in automatically generating or detecting humorous material. This thesis describes a precise model of the "HCPP" and implements a program to generate that class of puns. In doing so it confirms that some forms of humour can be codified and it reveals some new fundamental building blocks that will be required for comprehensive language generation and understanding.

1.2 Relevance of the research

The study of humour is an important research topic for a number of reasons. Natural language understanding and generation programs aim to model linguistic phenomena, and since humour is

a component of natural language, it too is an important area of research in the field of Artificial Intelligence (AI). Minksy argued that AI research aims to have a computer execute "a task which, if done by a human, requires intelligence to perform" [Min63]. Thus, joking, a quintessentially human act, lies within the domain of artificial intelligence.

The analysis of humour is also important because it can help guide the development of a lexical database which will ultimately need to be constructed for natural language understanding. For computers to achieve truly automated natural language understanding, they will probably require access to an enormously complex semantic network or database in which words and ideas have subtle links between them. The kind of links necessary and the complexity of this database have not been fully delineated. Jokes and quips are an example of natural language that relies heavily on subtle links, so this study will shed light on the giant network that will be required to generate and understand natural language.

Humour research also brings up the interesting question of how sophisticated a computer system needs to be in order to produce even the simplest joke. Levison and Lessard point out that linguistic humour is an "ideal testing ground for the points of contact between cognitive and linguistic knowledge" [LL95]. From our research, we have found that some good jokes can emerge from simple linguistic play, but that a complex system with a large knowledge base and some grasp of logic and common sense is necessary to generate consistently interesting jokes.

Humour is also worthy of study because it "provides us with valuable insights into the mechanisms which underlie 'normal' language production" [LL92, 175]. We will be concentrating on verbal puns which, for the most part, consist of two sentences that share certain semantic relations or links. Studying puns of this kind reveals the complexity of the process human beings take for granted when they generate a single coherent sentence or multiple sentences that relate to each other.

And finally, the study of what makes something funny is an interesting question in its own right. Attardo and Raskin write that "the text of a joke is always fully or in part compatible with two distinct scripts" [AR91, 308]. In straightforward 'just the facts' kind of communication, ambiguity is viewed as negative because it is an obstacle to comprehension. But other types of discourse such as metaphor, idioms, poetic language and humour often encourage ambiguity because interesting connections between normally dissociate concepts can be made. Studying this process is a worthy endeavor because poets and punsters perform a kind of cross-breeding of disparate ideas, giving rise to new, and sometimes interesting and entertaining ideas which invigorate and enrich the cultural imagination.

1.3 Goals

The goals of this thesis are to:

1. determine what kinds of links between words are necessary for generating HCPPs.
2. derive algorithms for generating these kinds of puns.
3. construct a small database from which that class of puns can be generated.
4. implement some of the algorithms using the natural language generator VINCI.
5. evaluate the performance of the program and suggest ways of filtering out bad results.
6. draw conclusions about this kind of humour and natural language generation in general.

In Chapter 2 we will discuss previous theories about humour and from them devise a practical model of humour. Chapter 3 will examine previous attempts at generating jokes automatically. Chapter 4 describes the algorithms designed for generating our chosen class of puns (goals 1 and 2 above) and the database they make use of. Chapter 5 describes how the algorithms are implemented in VINCI (goals 3 and 4). Chapter 6 will analyze the performance of our joke

generator and suggest ways of improving it (goal 5) while Chapter 7 will summarize what we have learned about humour and natural language generation (goal 6).

Chapter 2

Developing a Practical Model of Humour

In this chapter we discuss Attardo and Raskin's general theory of humour and extend it to support the goals of this research. Attardo and Raskin are two of the first researchers to begin to look at the mechanisms at work in humourous statements. Their work, which is more formal and detailed than previous research, offers important insights into what makes a text funny. Their theory is still not specific enough, however, to provide concrete guidance for developing a program that can generate jokes. In fact their model seems to suggest that a computer would need to possess a lot of cognitive power before it could generate jokes. Therefore we present in section 2.1.2 a general theory of humour that is inspired by Attardo and Raskin's model but is more detailed and computationally tractable than theirs.

2.1 Attardo and Raskin's theory of humour

Attardo and Raskin [AR91] analyzed a corpus of light-bulb jokes and postulated that there are six joke parameters by which all jokes (not just light-bulb jokes) can be classified. These six parameters are:

- Language
- Narrative strategy
- Target
- Situation
- Logical mechanism
- Script opposition

The **language** parameter of the joke describes "all the choices at the phonetic, phonologic, morphophonemic, morphologic, lexical, syntactic, semantic and pragmatic levels of language structure" [AR91, 298]. In short, this parameter specifies the way the joke is worded. The content

a	*How many <stereotyped group> 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.*	
b	*The number of <stereotyped group> needed to screw in a lightbulb? Five - one holds the bulb and four turn the table.*	same as (a) in all but the language parameter
c	*It takes five <stereotyped group> to screw in a light bulb: one to hold the light bulb and four to turn the table he's standing on.*	same as (a) but with a new narrative strategy
d	*How does a <stereotyped person> brush his teeth? He holds the brush and moves his head.*	
e	*How does a <stereotyped person> fan himself? He holds the fan and shakes his head*	same as (d) but with a new "situation" parameter.
f	*A T-shirt with the slogan "Gobi Desert Canoe Club".*	jokes (a) - (e) all use the same logical mechanism of role reversal. This joke however, uses a different logical mechanism called juxtaposition
g	*Customer: "I'd like to return this pair of shoes please". Clerk: "Did you wear them at all?" Customer: "Only once, while taking a bath".*	"garden path" logical mechanism.

Table 1: Jokes illustrating Attardo and Raskin's parameters.

(and some of the form) of the joke is determined by the five other parameters. This content can be expressed in different ways. For example, jokes (a) and (b) in Table 1 have all the other joke parameters in common but their language parameter is different (i.e. they are worded differently). The **narrative strategy** parameter describes the different forms a joke can take. For example a joke can be a riddle, conundrum, or expository text. An expository version of joke (a) or (b) would be (c). The **target** of a joke is a stereotyped group such as Poles, lawyers or blondes which acts as the butt of the joke. This parameter is optional because not all jokes ridicule. The event or **situation** in a joke describes an activity that is taking place "such as changing a light bulb,

crossing the road, playing golf" [RAR93, 125]. For example jokes (d) and (e) are similar in that they describe absurd ways of performing simple tasks. The only real difference between the two jokes is the situation or activity in which the absurdity is manifest. The **logical mechanism** parameter describes in what way logic is being undermined or played with in a joke. Jokes (a) - (e), for example, all employ the same logical mechanism of role reversal. The normally static things - the ground (on which the table rests), and the person's head do not usually move when performing the stated activities. An example of a different kind of logical mechanism is juxtaposition, which can be seen in joke (f) or the garden path phenomena in (g).

The last parameter in Attardo and Raskin's list is **script opposition**. They claim that "the text of a joke is always fully or in part compatible with two distinct scripts and ... the two scripts are opposed to each other in a special way" [AR91, 308]. Some of the ways scripts can be opposed are: real vs. unreal, good vs. bad, high status vs. low status or nondumb vs. dumb. For instance jokes (a) -(e) are examples of the nondumb vs. dumb opposition.

Kim Binsted suggests that Attardo and Raskin are mistaken in saying that two scripts in a joke are opposed. Often they are not opposites but just different [Bin96, 17]. See, for example, the following joke:

> *How many psychiatrists does it take to change a light bulb? Only one but the bulb has got to really want to change.*

There is no script opposition here - changing a lightbulb and a person changing are not opposites. But there is a script difference thanks to a play on the homonym "change" - one script is about psychiatry and the changes it can promote, the other is about changing lightbulbs (homonyms are words that have the same form (spoken or written or both) but differ in meaning. The homonym brings together two dissociate concepts and misleadingly suggests that there is some semantic similarity between them.

Attardo and Raskin arrange these parameters in the way shown in Figure 1. The diagram represents "a process in which the decisions and choices about the various traits and ingredients [of a joke] are made in a justified logical order" but what "it does not mean is that jokes are actually produced this way by the speakers" [AR91, 314]. Thus "the order of levels is totally devoid of any temporal value - a lower level is not a later level" [AR91, 327]. This kind of modeling may be familiar to students of linguistics. For example the model for generating a sentence "start(s) with the initial symbol S and pass(es) many underlying levels of decreasing depth and abstraction and experienc(es) complex transformations, all before reaching the surface form of the sentence" [AR91, 327]. But it "would be absurd to suggest that this is how a native speaker produces the same sentence" [AR91, 327]. Similarly, the model analyzing the components of a joke "do not correspond to the consecutive states of actual production" [AR91, 327].

Although they delve more deeply into the structure of jokes than many of their predecessors, Attardo and Raskin claim from the start that they have not developed "a model of joke production and that, therefore, production-related considerations do not and cannot inform the model" [AR91, 294]. Therefore they do not describe the sequence of steps a computer might be programmed to perform to generate a joke. More importantly, however, their joke parameters are not described in enough detail to offer concrete guidance in the development of a joke-generating program. The parameters are described at a high level and this seems to suggest that a computer would need to possess a lot of cognitive power - something which researchers in artificial intelligence have found extremely difficult to implement - in order to generate a joke. For example a computer would need to possess an extensive knowledge base to choose the appropriate props required for the situation, wield the power of logic and common-sense in order to subvert them ('know thy enemy') and create appropriate sentences given a chosen narrative strategy and the values of all the other parameters.

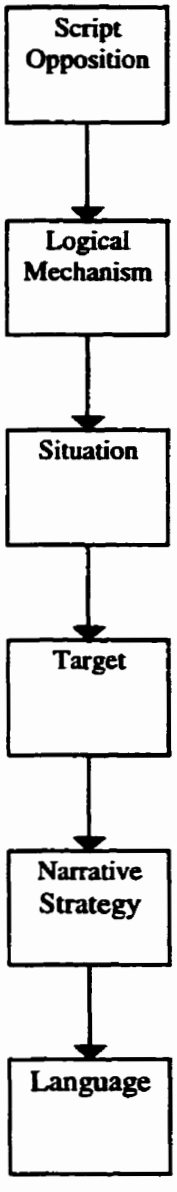


Figure 1. Attardo and Raskin's six joke parameters [AR91, 325].

Nonetheless, Attardo and Raskin's insights into humour are valuable and their model is a good starting point for constructing a more detailed model of humour in Section 2.3 and finally, the model of humour which we chose to implement (described in Section 2.4).

2.2 Dividing humour into two categories

Do we, as Attardo and Raskin's model suggests, really have to wait for an essentially intelligent machine to be built before humour can be generated? Probably not. The algorithmic modeling of humour becomes more feasible if we divide humour into two broad categories and choose the category that is more manageable - i.e. has less need of cognitive power.

2.2.1 Verbal vs. Situational humour

Two thousand years ago, Cicero wrote

> *For there are two types of wit, one employed upon facts, the other upon words.*
>
> (De Oratore, II, LIX, 239-40)

Today many linguists categorize humour in the same way, dividing it into situational humour and verbal humour (or prosaic vs. poetic [Ho77], contextual vs. linguistic [Bin96]). Verbal humour, is based on the following characteristics of utterances:

- phonological (the sound of words)
- morphological (how words are formed)
- syntactic (how words are put together to form sentences) [LL95]

An example of this genre is shown in Table 2(a). Unlike situational jokes, verbal humour depends heavily on form - on the lexical choices and syntax of the sentence - and often fails to be funny if synonyms or paraphrases of the punch line are used [LL95]. For example if we were to change the punch line to "Unfortunately none of the ten puns did", the paragraph is no longer humorous.

Situational humour on the other hand, consists of jokes in which no particular element in a sentence is crucial to the joke. An example is in Table 2(b). This joke can be expressed in a variety of ways - changing its syntax or form does not affect the humour of the utterance. Jokes of this kind are extremely hard to model and implement because encyclopedic knowledge, logic and common sense, the holy grails of artificial intelligence, are called upon to create and understand them.

This thesis will focus on the more realizable goal of generating instances of verbal humour. Verbal humour plays with linguistic devices and relations and hence is more susceptible to modeling than more cognitively complex situational humour. Investigating the generation of verbal

a	*A man entered a pun contest. He submitted ten puns in the hope that one would win. Unfortunately, no pun in ten did.*
b	*A poll was conducted in which 1000 women were asked if they would sleep with Bill Clinton. 35% of the women surveyed said yes, 40% said no and 25% said never again.*

Table 2: Examples of (a) verbal humour and (b) situational humour.

humour may also act as a useful entry point into identifying and solving some of the difficulties of generating situational humour.

2.3 The Ambiguity Theory of Humour

Attardo and Raskin claim that "the text of a joke" (presumably both verbal and situational jokes since they do not, as we have, distinguish between linguistically-based jokes and more cognitively complex ones) "is always fully or in part compatible with two distinct scripts" [AR91, 308]. Often these two (or more) scripts exist within an ambiguity (an ambiguity is an expression able to be interpreted in more than one way). This insight acts as the foundation for constructing a more detailed and practical model of humour than Attardo and Raskin's. Pepicello,

Greene and Binsted also assert the central role of ambiguity in humour and distinguish between "low-level" ambiguity and "high-level" ambiguity.

2.3.1 Low-level ambiguities.

Low-level ambiguities "allow several different senses for a word or words in a spoken or written text" [Bin96, 38]. This type of ambiguity predominates in verbal humour and functions at the level of words, substrings and phrases.

Word-level ambiguity

A text with word-level ambiguity contains a word with multiple meanings. The pronunciation or spelling (or both) of a word in a sentence suggests two or more meanings for that word. Since there are two parameters at work here (pronunciation and spelling), there are four different ways a word can be ambiguous.

1. word sense ambiguity: a word with one phonological form and one written form corresponds to two or more meanings. This kind of word is called a homograph. (Homographs are words that are pronounced the same and written in the same way but differ in meaning). Table 4(a) uses word sense ambiguity.

		sound	written	linguistic term	examples
1	word sense ambiguity	same	same	homograph	"bank"(financial institution), "bank"(river's edge).
2	spelling ambiguity	same	different	homophone	"hair" "hare"
3	pronunciation ambiguity	different	same	homograph	"wind"(which blows through the trees), "wind"(the action of winding).
4	two different words	different	different		"whistle" "kite"

Table 3: Word-level ambiguities.

2. spelling ambiguity: one word corresponds to two or more written words which are pronounced the same but have different meanings. This kind of word is called a homophone[1]. Table 4(b) uses spelling ambiguity.

3. pronunciation ambiguity: one written form corresponds to two or more phonological forms and meanings as shown in Table 4(c). Jokes using this kind of ambiguity contain homographs (words which are written in the same way and may or may not have the same pronunciation) and are visual rather than oral jokes because when the joke is said out loud, only one of the phonological forms is expressed. Perhaps for that reason, they are rare.

4. Two words are different in both pronunciation and spelling. These are not words upon which a pun would revolve.

Substring-level ambiguity

Two words that do not look or sound identical to each other can be similar enough that their substrings are suggestive of each other. Table 4(d) is an example. In this joke there

- is a substring similarity between spook/spec
- are clues in the surrounding sentences ("short-sighted" and "What do ... wear") that suggest the fake word is supposed to be evocative of the word "spectacles".

There are four different ways a substring of a word can resemble a word, as illustrated in Table

a	*Where do snowmen keep their money? In snow banks.*	pun with a homograph in it.
b	*What do you get when you cross a rabbit with a lawn sprinkler? Hare spray.*	pun with a homophone in it.
c	*They drove down the mountain switchbacks as the hurricane set in. It was a very windy road.*	pun with a homograph in it.

[1] Homonyms are a superset of homographs and homophones. In other words, homographs are homonyms and homophones are homonyms but not vice versa.

d	*"What do short-sighted ghosts wear"? "Spooktacles".*	pun with substring replacement in it.

Table 4: Examples of puns with word-level ambiguity.

5. All of these words can be used to create humorous statements. For instance, "'*I hate chemistry' Tom said acidly*" or "*What's a cat with eight lives? An octopus*".

	word	substring	word evoked	identical feature
a	acidly	acid	acid	root word (sound and sight)
b	octopus	pus	puss	sound
c	windy	wind	wind	sight
d	spectacles	spec	spook	consonants

Table 5: Examples of substring-level ambiguity.

Phrase-level ambiguity:

In phrase-level ambiguity, word boundaries are played with. For instance the last syllable of a word in a phrase and the first syllable of the word that follows it can form a word, or two words can join together to form a single word as in Table 6(a). Another kind of phrase-level ambiguity occurs when syllables of non-adjacent words are swapped (this is called metathesis) "to suggest (wrongly) a similarity in meaning between two semantically-distinct phrases" as in Table 6(b) or 6(c) or when one phrase is mistaken for another as in Table 2(a) [Bin96, 6].

Syntactic ambiguity

Another kind of low-level ambiguity is syntactic ambiguity. It describes the situation in which a sentence has multiple meanings because it can be parsed in more than one way. An example of a joke that uses syntactic ambiguity is Table 7(a). The ambiguity, of course, is

a	*Why didn't the frog sit on the toadstool. Because there wasn't much room (mushroom).*	two adjacent words forming a single word.
b	*What's the difference between a torn flag and a bent sixpence? One's a tattered banner and the other's a battered tanner.*	swapping of consonants in non-adjacent words (metathesis).
c	*I'd rather have a bottle in front of me than a frontal lobotomy.*	more complicated metathesis. The syllables in common to both phrases are: 'front' 'll' 'bot' 'me'. Those not shared are 'in' and 'of').

Table 6: Examples of jokes using phrase-level ambiguity.

a	*"Would you rather have an elephant kill you or a gorilla?" "I'd rather have the elephant kill the gorilla".*	syntactic ambiguity
b	*"How can you tell if an elephant has been in your fridge"? "Footprints in the butter".*	contextual ambiguity
c	*"Waiter, there's a fly in my soup!" "Don't shout so loud, sir, or everyone will want one".*	pragmatic ambiguity
d	*"Why do birds fly south in the winter?" "Because it's too far to walk"*	focus ambiguity

Table 7: Examples of syntactic ambiguity (a) and high-level ambiguity (b-d) [Bin96, 39].

whether the gorilla is (i) a subject or (ii) an object. Parsing (i) is more obvious than parsing (ii) because the question cleverly leads us into making a faulty assumption. Given that most of us are interested in self-preservation, it would be absurd to ask our preference between being killed or having an animal killed so we discount parse (ii).

Also, we are subtly fooled by the question because the elephant and the gorilla are both animals in the wild and so we unconsciously group them together as alternate subjects for the verb "kill". This kind of shrewd manipulation of knowledge and reasoning comes into play in situational humour. Therefore Kim Binsted oversimplifies her analysis of joke 7(a) when she implies that it is only syntactically complex. The ambiguity of this joke depends on not just syntax but reasoning and encyclopedic knowledge. In other words, the ambiguity in this joke is a hybrid of low-level and high-level ambiguity. High-level ambiguity, described in the next

section, plays with reason and knowledge. Thus it is important to keep in mind that many jokes that are classified as examples of verbal humour might also contain high-level ambiguities and jokes considered examples of situational humour might contain low-level ambiguities.

2.3.2 High-level ambiguity

Examples of situational humour exploit high-level ambiguity (although they often contain low-level ambiguities as well). These types of ambiguity are not linguistically based but instead involve the manipulation or play of logic, common sense and world knowledge[2]. Thus in situational humour, "the senses of the individual words are not in question, only the interpretation of the whole text" [Bin96, 38]. Because this kind of humour works by manipulating logic or world knowledge, there are as many kinds of situational jokes as there are logical tenets and facts about the world. It would be impossible to compile a complete list of the different kinds of ways situational jokes are ambiguous but we will describe three to give the reader an idea of this kind of ambiguity.

Contextual ambiguity

Binsted claims that contextual ambiguity is a "conscious manipulation of social decorum" [Bin96, 40] and provides item (b) in Table 7 as an example. The ambiguity (incongruity might be a better word in this case) which creates the humour in the text is that if an elephant had in fact been inside the fridge, it would have destroyed the fridge and would not have just delicately left footprints in the butter. Binsted states that "there is no linguistic ambiguity ... (in this joke) because neither the question nor the punchline have more than one interpretation" [Bin96, 40]. She chooses to ignore that "being in someone's fridge" is a common phrase which means having taken something out of the fridge and instead sees only the literal interpretation of someone

having bodily entered the fridge. Both interpretations are absurd - elephants don't appear in people's homes and take things from the fridge (but not as absurd as you might think because anthropomorphism is common in jokes). Nor can an elephant climb into someone's fridge and just leave footprints in the butter. But the joke does not really depend on the question being interpreted in two ways. It is better if both interpretations are discerned but this is not necessary.

Pragmatic ambiguity

Pragmatic knowledge "concerns how sentences are used in different situations and how use affects the interpretation of the sentence" [All95, 10]. Item (c) in Table 7 is an example of a joke that makes use of pragmatic ambiguity. In this joke, the act of shouting functions as a kind of homonym because two different emotions can be ascribed to it. In our culture, a shout or exclamation can result from anger or it can spring from happiness or excitement. But given the situation - the obviously negative experience of having a fly in one's soup - the emotion we would reasonably expect the shouting patron to be feeling is anger. The waiter's response to the patron's outburst is funny because he interprets or pretends to interpret the exclamation as a sign of gastronomic gusto rather than disgust. Thus the confusion in this joke is not based on the kind of linguistic trickery of low-level ambiguity but rather more cognitively complex knowledge of context and human psychology.

Focus ambiguity

The focus or point of the sentence is under-constrained in joke (d) of Table 7. Like joke (a), this pun is a hybrid of low-level and high-level ambiguity. The syntax of the sentence, which allows "why" to refer to different levels of causation, is crucial because it creates the necessary ambiguity in which cognitive knowledge is played with. The high-level ambiguity makes use of a

---

[2] World knowledge is the general knowledge about the structure of the world that language users must have

kind of meta world knowledge - knowledge about the state of an average human being's knowledge of the world and hence what a "reasonable" question might be. The "why fly" interpretation is absurd because the answer to it is so obvious. However, asking why birds fly to a certain area during the winter season would be a more challenging and viable question - the reason for their migration south is not so obvious. Table 8 summarizes the different ways that jokes make use of ambiguity.

low-level ambiguity:

words:

- the pronunciation and spelling of two words are identical.
- the pronunciation of two words is identical.
- the spelling of two words is identical.
- the words have a partial phonetic ressemblance. (i.e. paronymy).

substrings:

- the pronunciation and spelling of a word's substring are identical to some other word.
- the pronunciation of a word's substring and some other word is identical.
- the spelling of a word's substring and some other word is identical.
- neither the sound nor spelling of a word's substring and some other word are identical but the two words still resemble each other (paronymy).

phrases:

- words or syllables of words next to each other form a word.
- syllables of non-adjacent words are swapped to form new words.
- a word contains other words within it.
- a phrase contains a different phrase within it.

syntax:

- a sentence can be parsed in more than one way.

high-level ambiguity:

any aspect of logic, reasoning or normal experience is played with. Some examples are contextual ambiguity, pragmatic ambiguity, focus ambiguity.

Table 8: The different kinds of ambiguity that appear in jokes.

---

in order to, for example, maintain a conversation" [All95 10].

2.3.3 Are all jokes ambiguous?

Attardo and Raskin claim that "the text of a joke is always fully or in part compatible with two distinct scripts" [AR91, 308]. Often those two or more scripts coexist within an expression that is able to be interpreted in more than one way - i.e. within an ambiguity. Indeed all puns contain an ambiguity but not all jokes do. For example 2(b) does not contain a phrase or sentence which can be interpreted in multiple ways. But it does contain multiple scripts. One script consists of what we expect - a regular poll which ends up with "25% are not sure". The second script is what actually occurs. In other words, in these types of jokes, one script represents what the audience of the joke expects - given its knowledge of the world or its common sense or what it thinks should logically follow - and the other script represents what actually (and surprisingly) occurs.

Thus all jokes seem to contain multiple scripts but not all jokes are ambiguous. Ambiguity is only one way those two or more scripts can manifest themselves. This thesis concentrates on verbal humour and so will deal exclusively with texts containing an ambiguous phrase or sentence.

2.3.4 What kind of ambiguity is funny?

All puns contain a linguistic ambiguity but not all ambiguous texts are funny. For example, joke 9(a) is ambiguous - we do not know whether "Tom" or "John" is the antecedent of "He" - but it is not humorous. One of the reasons why the text is not funny is because the ambiguity in it needs to be resolved as in example 9(d). Simply resolving the ambiguity, however, does not render the passage funny, as example 9(b) makes clear. The ambiguity has to be resolved in a surprising way. Thus text 9(b) is not funny because it is obvious that Tom, the victim of a crime would be furious.

Even if the ambiguity is resolved in a surprising way, however, the text needs to somehow justify the second script and have it make some kind of sense. For example, in 9(c), the ambiguity is resolved in a surprising way - the thief rather than the victim is furious - but to most readers the text is probably more confusing or absurd rather than humorous because no reason is given for the thief's anger. The joke makes more sense if we offer some kind of rationalization for the surprising second script. Thus 9(d) is an improvement of 9(c) because it contains a rationalization for the surprising second script. The result is a funny comment about the thief's self-centredness and skewed moral view - the fury resulting from the theft does not emanate from the victim of this crime but from a petulant thief who is unhappy with the stolen item.

a	*John ate Tom's chocolate bar. He was furious.*	A text with unresolved ambiguity is not funny.
b	*John ate Tom's chocolate bar. He was furious because he wanted to eat it.*	The ambiguity is resolved but there is no surprise. The result is not funny.
c	*John ate Tom's chocolate bar. He was furious - John that is.*	
d	*John ate Tom's chocolate bar. He was furious - the bar was stale.*	The ambiguity is resolved in a surprising way and the result is funny.

Table 9: Ambiguous texts.

2.3.5 Script precedence vs. script equality

Two scripts are present in joke 9(d). "He" refers to Tom up until the sentence "The bar was stale" appears. At that point the second and more surprising script emerges in which "He" is "John". Attardo and Raskin claim that the punchline of a joke "triggers the switch from the one script to the other by making the hearer backtrack and realize that a different interpretation was possible from the very beginning" [AR91, 308]. In fact Giora claims that "the reader is made to cancel the first interpretation upon processing the second marked interpretation [Gio91, 470]. But not all jokes involve one script canceling another. For example

Where do you weigh a whale? At a whale-weigh station.

The pun "does not resolve to a single interpretation" [Bin96, 31]. "The railway station interpretation does not replace the one about weighing whales; instead, they seem to be combined into a somewhat nonsensical vision of a place where trains come *and* go and the weight of whales is determined" [Bin96, 31]. Both of these dynamics - where one script takes precedence over another and where multiple scripts coexist without one cancelling the others - will occur in the jokes created by our joke generator.

2.3.6 Where does the ambiguity occur?

Sometimes the ambiguous phrase or sentence of a joke occurs in the first part of the text and its second meaning is not discerned until the punchline reveals it (see jokes 7a and 7d). The surprise comes from having the second meaning (which is not as obvious as the first one for some reason) revealed in a subtle way. Surprisingly, the exact opposite dynamic can occur in which the ambiguity lies in the punchline. For example most Tom Swifties (see Table 11) follow this format in which the earlier part of the text reveals the duality of meaning of the punchline.

2.4 A Practical Model of Humour

This section describes a practical model of humour that borrows from Attardo and Raskin's theory and from the above analysis of the role of ambiguity. This model will be useful for describing all joke-generating systems discussed in this thesis.

Although Attardo and Raskin assert that they do not have a model for the generation of jokes, their theory leads naturally to the process shown in Figure 2 in which their parameters come into play.

2.4.1 Choose a Narrative Strategy

As Attardo and Raskin point out, the context in which a joke is being created must be taken into account. For instance it will determine which of the parameters is the first to be instantiated. For example, "one may witness a dumb act" and so the joke's first instantiated parameter would be Script Opposition. Or one might "hear another ground-figure reversal story (Logical Mechanism), find oneself in a car wash (Situation), hear a Polish joke (Target), participate in a riddle contest (Narrative Strategy), or hear a joke phrased in a certain way and be reminded of another one similar to it (Language) ... Virtually any combination of [the parameters] may be present in a situation and cause the production of a joke by calling for the appropriate choices pertaining to the absent [parameters]" [AR91, 326]. Thus the first step in the generation will vary

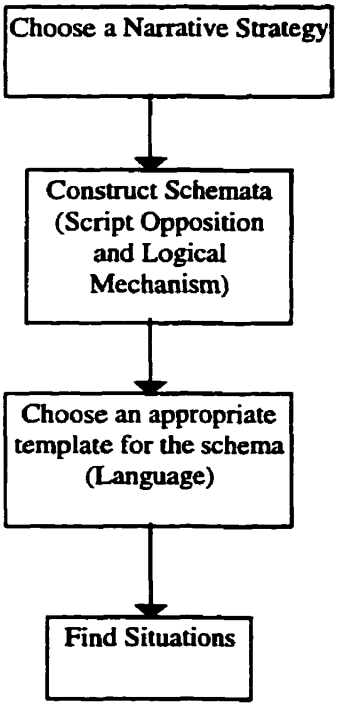


Figure 2: A sequence of steps for artificially generating jokes.

given the context in which a joke is constructed. We, along with previous researchers, have found that choosing the narrative strategy before all the other parameters simplifies the generation process because we have to select what kinds of humour are realistic to compute right

now given that current computers are not thinking machines. Also, this thesis is more interested in the logical mechanisms of jokes (what makes texts funny) than the different ways the ideas in a joke can be expressed. And finally, narrative strategies are often cultural (e.g. knock-knock or elephant jokes) and hence are inaccessible to machines so they are chosen at the beginning of the generation process.

2.4.2 Construct Schemata

Once a narrative strategy has been selected, the next step in our model for generating a joke is to construct a schema like the kind described by Binsted. Following Binsted, we use the idea of a

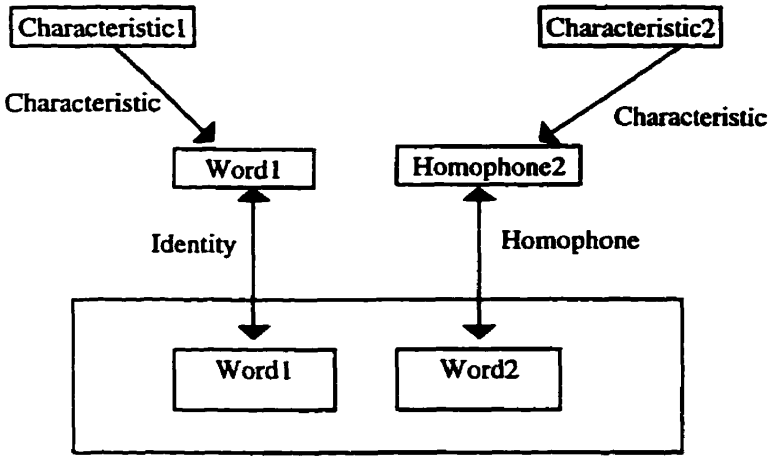


Figure 3: An uninstantiated schema

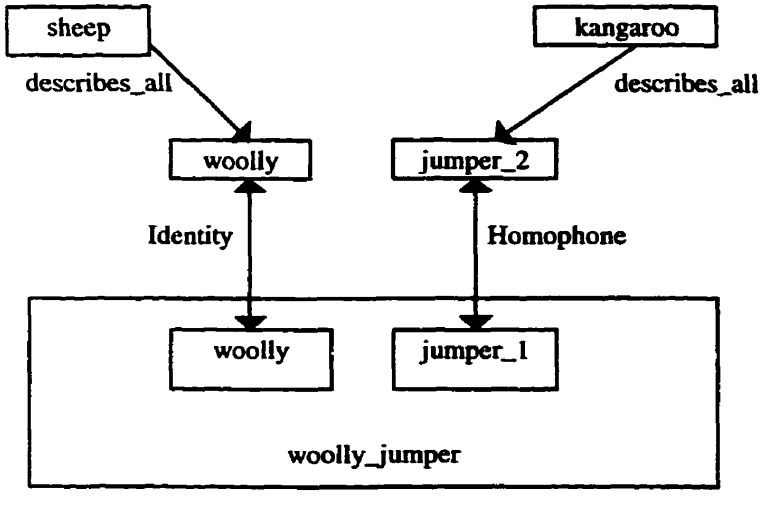


Figure 4: An instantiated schema which can yield the following riddle: *What do you get when you cross a sheep and a kangaroo? A woolly jumper.*

"schema" to represent the underlying mechanism of a riddle. Figure 3 shows an example schema which builds a riddle based on a noun phrase in which the second word has a homonym. Then a phrase is built based on this homonymous meaning and the original meaning of the phrase as in Table 10(a). Texts 10(b) and 10(c) are examples of jokes created by different schema.

Binsted's idea of a schema encapsulates Attardo and Raskin's notions of logical mechanism and script opposition (Section 2.1). The types of schemata which researchers like Lessard and Levison and Binsted develop, employ what Attardo and Raskin would consider "the most trivial logical mechanism, a kind of default" : "the juxtaposition of two different situations determined by the ambiguity or homonymy in [the] pun" [AR91, 306]. Similarly, the logical mechanism of this thesis's pun generator is also homonymy. The two (or more) meanings the homonym accidentally brings together are the opposing scripts.

a	*"What is green and bounces?" "A spring cabbage".*	The first word in a noun phrase is the homonym.
b	*"Where do hamburgers like to dance?" "At a meat ball".*	The second word in the noun phrase is the homonym. (a) and (b) have different schemata.
c	*"What's the difference between a pretty glove and a silent cat". "One's a cute mitten, the other's a mute kitten".*	A schema that uses metathesis as the basis for the pun.
d	*"What do you call a sheep that leaps?" "A woolly jumper".*	Puns (d) and (e) share the same template but use different schemata.
e	*"What do you call a hairy beast that swims?" "A weir-wolf".*	
f	*"What do you get when you cross a sheep and a kangaroo". "A woolly jumper".*	Puns (d) and (f) share the same schema but use different templates.

Table 10: Examples of Schemata and Templates [Bin96] The "schema" describes the essence of a pun - its precise lexical structure. The template is the frame in which this lexical structure is inserted.

2.4.3 Choose a Template

When a narrative strategy has been selected and a schema has been fully instantiated, a template is designed. Templates correspond to Attardo and Raskin's Narrative Strategy and Language parameters. A template consists of fixed text and blank spots or "slots" where words or phrases generated by the schema can be inserted [Bin96, 68]. For instance the template for joke 10(f) is

> **What do you get when you cross** [appropriate text fragment gleaned from schema] **with** [text fragment from schema]**?** [the constructed noun phrase].

The words in bold are the canned text and the brackets [ ] represent the slots. For example puns 10(d) and (e) have different schemata but share the same template ("**What do you call a ... that ...**").

A single schema can produce numerous puns if different templates are used. For example let us say we have the following entries in a lexicon (the notation here is from [BR94, 14]).

```
class jumper_2
*act_verb: leap
$describes_all: kangaroo

class woolly
*$describes_all: sheep
```

One template builds a riddle from the relations marked by * while another is required to construct a riddle from relations like those marked with the $, yielding 10(d) and 10(f) for example. The same schema is at work - a phrase is built based on "woolly" and a homonym of "jumper" - but different relations between lexemes were chosen and hence two different templates have to be used to express the joke. In some cases a lexicon will have the entries needed for some templates but not others. The joke generator developed for this thesis will use various templates.

2.4.4 Find Targets and Situation

The final stage in generating a joke is to insert what A&R call the "situation" into the template. We propose, following Lessard and Levison, and Binsted, that this final step can be automated. The computer will search a network of words or database in order to instantiate the schema. Some of the words gleaned from the network will represent the "situation" of the joke (the actions, participants, and objects described in the text). In the example of the instantiated schema of Figure 5, no target appears but the situation is the following:

actions: bouncing
objects: something green, a spring-cabbage.

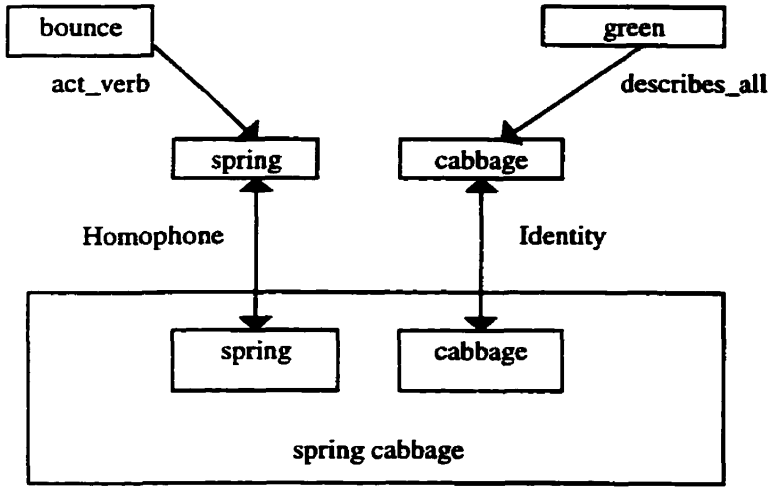


Figure 5: An instantiated schema which can yield the following riddle: *What is green and bounces? A spring cabbage.*

The joke-generators created by Levison and Lessard (sections 3.1 and 3.2), Binsted (section 3.3) and the one created and described by this thesis (chapter 4) - do not include a target parameter. The exclusion of the target parameter does not contradict Attardo and Raskin's theory of humour, however, because it is an optional parameter in their model (the only optional parameter in fact). One could argue that in the example in Figure 5, the target of the joke is the "spring-cabbage" but

Attardo and Raskin define the target as "an individual or group" possessing some kind of stereotype (for example "dumbness") for which they are mocked. Thus subjects of puns such as "spring-cabbages", which are neither human nor enjoy any kind of stigma we know of, will be regarded as situational props by our model.

Chapter 3

Previous Automated Joke Algorithms

Researchers have successfully created systems that generate certain kinds of simple verbal jokes. Lessard and Levison examined Tom Swifties and riddles and discovered that many of them "can be seen as the product of a clearly defined set of rules such that once the model is provided, an indefinitely large number of (them) may be created" [LL93]. They implemented these rules using VINCI, a natural language generator which they developed. Kim Binsted also analyzed riddles and modeled ones that "share deep traits" (34) and "semantic patterns" (35). She then wrote a program that generates a specific class of them. Sections 3.2 through 3.4 explain these models they constructed and how they implemented them.

3.1 Lessard and Levison's model of Tom Swifties

Levison and Lessard have modeled a class of Tom Swifties. Examples of this class of pun are in Table 11[LL97]. Every Tom Swifty has a central element that Lessard and Levison call the "pivot" around which the joke turns. For example in joke 11a, the pivot is "crabbily". Within that word is another word ("crab") called the base which is semantically related (in this case, by hyponymy) to an earlier word or phrase in the sentence called the target ("seafood"). From Table 11 (a-e) we see that the pivot of a Tom Swifty can be a verb, a noun phrase, an adverb or an adjective. A relationship of the base with the pivot and the pivot with the target are the minimal requirements for a Tom Swifty. Some examples have yet another relation however, between the pivot and an earlier part of the sentence such as the relation between the pivot "crabbily" and the phrase "I hate".

a	*"I hate seafood" Tom said crabbily.*	
b	*"We steal things together" Tom corroborated.*	the pivot is a verb.
c	*"I've only enough carpet for the hall and landing" said Tom with a blank stare.*	the pivot is a noun phrase (in a prepositional phrase).
d	*"I love the novels of D.H. Lawrence" said Tom chattily.*	the pivot is an adverb.
e	*"I dropped the toothpaste" said Tom, crestfallen.*	the pivot is an adjective.
f	*"I wish I were taller" Tom said longingly.*	synonyms: tall/long; synonyms: longing/wishing.
g	*"I wish I were taller" Tom said shortly.*	antonyms: tall/short.
h	*"I love seafood" Tom said shellfishly.*	hyponyms: shellfish is a subset of seafood. substring similarity: shellfish and selfish.
i	*"I hate pizza" Tom said crustily.*	meronyms: the relationship between parts and wholes.
j	*"I am getting drunk" said Tom wryly.*	instrument: a means of becoming drunk is rye.
k	*"I hate chemistry" Tom said acidly.*	domain: acid is an entity associated with chemistry.
l	*"There's too much tabasco in this chili" Tom said hotly.*	quality: tabasco sauce has the quality that it is hot.
m	*"It's a unit of electric current" said Tom amply.*	paraphrase: a unit of electric current is an amp.

Table 11: Examples of Tom Swifties.

Lessard and Levison's implementation of Tom Swifties can be described in terms of schemata and templates. Although they never refer to their algorithm in these terms, their design for generating jokes can be represented as the schema of Figure 6. The template will be "I ____ _____ " said Tom _____.

Lessard and Levison propose a hierarchy of Tom Swifties ranging from verbal humour to situational humour [LL97]. Table 12 shows some of the more complicated Tom Swifties.

3.1.1 Computational generation of Tom Swifties

This model was implemented by Lessard and Levison with their natural language generator system called VINCI [LL92a]. VINCI consists of

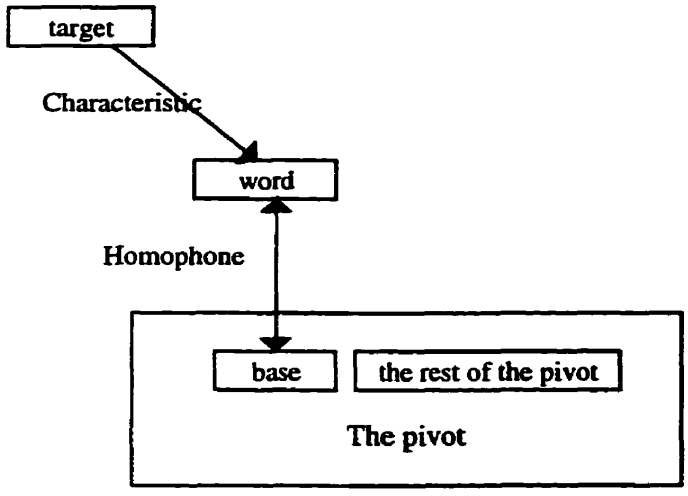


Figure 6: The schema for a simple Tom Swifty.

a	*"Yes my lobotomy was successful" said Tom absent-mindedly.*	the pivot contains multiple bases.
b	*"These are the propulsion systems used by NASA for the moonshots" said Tom apologetically (i.e. Apollo jet).*	the target is made up of multiple words.
c	*"Whenever I put on my scuba gear I get pins and needles" said Tom divertingly (diver-tingly).*	both the target and the pivot have multiple components
d	*"I had to fire my first mate when she got too heavy for the boat" said Tom excruciatingly (ex, crew, she ate).*	the pivot has multiple bases and the link between its components and earlier parts of the sentence are phonological (based on sound) rather than orthographic (written form).

Table 12: Situational Tom Swifties.

- a context free generator
- a mechanism which can transform, add, delete or move subtrees in the syntax tree
- a lexicon containing
  - words
  - the word categories they belong to (i.e. noun, adjective)
  - relations that hold between them such as synonym, hyponym.

In this way an item in the lexicon can point to other items in the lexicon and so a connected network of words can be created. Let's say our lexicon looks like this:

```
"crab"|N|edible...|hyperonym: "seafood"; paronym:"crabbily"/ADV|...
"seafood"|N|edible...
"crabbily"|ADV|attitude...|base:"crabby"/ADJ;...
```

Each lexical entry appears on a separate line and has fields delimited by vertical bars. Given this lexicon, we can express an algorithm as follows:

(1) choose a word (the base) that possesses both a hyperonym and an adverbial paronym in its lexical entry. (A hyperonym of a word x is the superset to which x belongs. For example, a hyperonym of "crab" is "seafood", of a "cat" is "animal").

(2) generate a sentence with a template that uses both the hyperonym and paronym. (Paronyms are words derived from the same root that differ in meaning. For example "man", "mannish", "crab", "crabbily").

In our example, the base word that fits these criteria is "crab". It will act as the seed for generating a pun. In VINCI, the syntax for generating a simple Tom Swifty of this kind is given in Figure 7 [LL95]. A transformation (a rule which manipulates a syntax tree) called SWIFTY

```
{Syntax}

{Find a base noun which has both a hyperonym and a paronym in field 13}
BASE = N[edible]/13=hyperonym/13=paronym/ADV
{Generate a sentence from the base using first its hyperonym, then its paronym}
SWIFTY = TRANSFORMATION
N : PRON V[evaluation, edible.directobj] 1/@13:hyperonym
    V[said] N[Tom] 1/@13:paronym/ADV;

{Apply the transformation to the BASE}
ROOT = SWIFTY: BASE
```

Figure 7. A VINCI grammar for generating a simple Tom Swifty.

expands the syntax tree BASE (which is simply a word in this example) and produces:

(i) a pronoun "I"
(ii) a verb of evaluation which takes an edible object as a direct object, such as "hate" or "like"
(iii) the hyperonym of the base ("seafood")
(iv) the verb "said"
(v) the noun "Tom"
(vi) the adverbial paronym of the base ("crabbily").

This grammar produces Tom Swifties such as "I hate seafood" said Tom crabbily.

Items i, iv, v represent the template's canned text and items ii, iii and vi represent words generated by the schema.

3.2 Lessard and Levison's model of riddles

In addition to Tom Swifties, Lessard and Levison also modelled two classes of riddles that use spelling or word sense ambiguity. These riddles, like Tom Swifties, contain a "central point of the humorous structure" which they call the "pivot" of the joke [LL93]. This pivot is a pair of homonyms that are spelled differently. Lessard and Levison's model generates riddles such as those listed in Table 13. The first class of riddle, which they call a "syntagmatic homonym riddle" (examples a and b in Table 13), does not replace a word with its homonym but instead supplies both words in the punchline.

a	*"What are groups of sailors on an ocean pleasure trip?" "Cruise crews".*	syntagmatic homonym riddle that uses spelling ambiguity.
b	*"What does the man who looks at oceans do all day?" "Sees seas".*	syntagmatic homonym riddle that uses spelling ambiguity.
c	*"What has a mouth and does not speak?" "A river".*	paradigmatic homonym riddle that uses word-sense ambiguity.

Table 13: Syntagmatic and paradigmatic homonym riddles generated by Lessard and Levison.

The second class of riddles modelled by Levison and Lessard are what they call "paradigmatic homonym" riddles such as 13(c) is an example. The homonym in this kind of riddle has specific characteristics such as:

- it is part of something (an animate being's body)
- it has a role (eating or speaking)
- its second meaning is an entity which is part of something ("mouth of a river").

3.2.1 Computational generation of this class of riddles

Lessard and Levison used VINCI to generate both syntagmatic and paradigmatic homonyms. Consider, for example, the paradigmatic homonym riddles. In their table entry for that noun, the 13th field must contain a pointer to a holonym (a larger entity which encompasses some part) and a pointer to a typical role filled by the noun. In VINCI, the syntax for generating the paradigmatic homonym riddles is provided in Figure 8 [LL93].

```
ROOT= N[bodypart]/13=homn/13=rol
RIDDLE = TRANSFORMATION
N: FRAME[has] 1/@13:homn FRAME[cant] 1/@13:rol PUNCT[question];

SOLUTION = TRANSFORMATION
N: DET[indef] 1/@13:homn/@13:hol;

QUESTION = RIDDLE:ROOT
ANSWER = SOLUTION:ROOT
```

Figure 8: The VINCI grammar for generating a class of paradigmatic homonym riddles. The tree ROOT selects a noun like "mouth" which is a homonym and represents a body part. The transformation RIDDLE creates a FRAME or template of the form ("What has a" ...<homonym> "but can't" <role played by homonym>. The solution is an indefinite article followed by the holonym of the homonym. Thus riddles such as 13(c) can be generated from this syntax.

3.3 Binsted's model of a class of riddles

Kim Binsted also models simple riddles - claiming that "it would be over-ambitious to tackle sophisticated adult humour at this stage" [BR94, 4] - and implements them in a program. The riddles generated by her program have been deemed by human judges to be "of comparable quality to those in general circulation among school children" [BR94, 1].

She concentrates on jokes which use word-level ambiguity (2.3.1) and the strategy of metathesis. These puns are built on common noun-phrases and substitute a word in that noun phrase for a homonym. Her program makes use of schemata and templates (described in Section 2.4.2) and a lexicon.

3.3.1 Computational generation of this class of riddles

The lexicon

Binsted's lexicon contains semantic and syntactic information about words and noun phrases. (The common phrases Binsted's program makes use of are just noun phrases). Each entry has a unique identifying symbol or "lexeme" and a number of "slots" associated with it. For example a homonym like "habit" must have two lexemes or unique identifiers such as "habit_1", "habit_2" in order to distinguish the two meanings of the word. The slots contain syntactic information and semantic relations for words in the lexicon. Table 14 lists the semantic relations Binsted's generator requires to construct her class of riddles [Bin96, 76]. It is important to note that the information contained in the lexicon is "general and neutral - the joke-generating power lies elsewhere in the program, particularly in the schemata and templates, and the ties between them" [BR94, 13].

Each lexeme is a node in a network. The values in the semantic slots are often other lexemes but they sometimes contain chunks of text in near-surface form (a word, phrase, or sentence in grammatical, understandable English but is not in perfect surface form (e.g. it may not have

SLOT	Used With	Allowed Values
CLASS	np, noun	The immediate superclass of the lexeme. e.g. (lemon, fruit)
SPEC_IS	np, noun	The class to which the entered lexeme belongs. (The class defines the entered lexeme reasonably precisely). e.g. (lemon, citrus).
IS	np, noun	A lexeme that typically describes the entered lexeme. e.g. (lemon, sour).
HAS	np, noun	e.g. (lemon, pips)
ACT_VERB	np, noun	A verb lexeme. Something the thing typically does. e.g. (chef, cook)
ACT_OBJ	np, noun	The near-surface form of the object of the ACT_VERB value. e.g. (chef, food). (A chef cooks food)
INACT_VERB	np, noun	A verb lexeme. Something you typically do to the thing. e.g. (horse, ride)

LOCATION	np, noun	The near-surface form of its typical location. e.g. (horse, "in a pasture").
USED_TO	np, noun	A verb lexeme. Something the thing is typically used to do. e.g. (spatula, flip).
USED_TO_OBJ	np, noun	The near-surface form of the object of the USED_TO value. e.g. (spatula, pancakes). (A spatula flips pancakes).
SYNONYM	np, noun, adj	A lexeme of the same category as the entered lexeme. e.g. (pillow,cushion)
DESCRIBES_ALL	noun, adj	A lexeme which refers to a thing or class of things which can always be described by the entered lexeme. e.g. (slimy, worm).

Table 14: Lexical relations used to generate a class of riddles. Schemata make use of different relations so not all the slots need to be filled.

capitals at the beginning of sentences, etc)). For example, the lexeme "horse" in Table 14 has the phrase "in a pasture" for its LOCATION entry. Binsted uses phrases "so that they can be put directly into a template without further syntactic manipulation". She justifies this because her dissertation is about generating jokes, and not about "complex (but uninteresting) syntactic generation" [BR94, 13].

Thus Binsted uses schemata, templates and a lexicon to generate jokes. She claims, apparently unaware of Lessard and Levison's work, that her program differs significantly from other attempts to computationally generate humour in three ways:

1. Its lexicon is humour-independent. The algorithms are what create the joke, not the lexicon. In other words the joke is not "lexicalized" [LL93].

2. Another program[AR94] generates riddles that are similar merely in surface form. This program generates riddles that are similar on a "strategic and structural level" [BR94, 12].

3. It is an implementation based on a model of riddles. If the results are poor, the model is probably wrong.

Lessard and Levison have also done these things. This thesis will also make use of schemata and templates and a lexicon and will use Lessard and Levison's natural language generator VINCI to generate a class of jokes.

3.4 What components of the practical model (Section 2.4) are performed by the computer in previous joke-generators?

In the automatic generators of jokes that we discuss in chapters 3 and 4, the narrative strategy has been chosen by the creator of the generator. The appropriate schemata for a subset of that kind of joke have been programmed and appropriate templates for each schema have been written by humans. The computer then executes its algorithms and performs the last step of our practical model (Section 2.4) when it searches the database. Therefore the computer has been equipped to perform step 4 of the process represented in Figure 2 (Section 2.4). Future research might change the context in which a joke is to be created and thus have the machine do more. For example, let us say the computer is asked to make a joke about apples. The situation of the joke has been provided. More complicated kinds of logical mechanism and script opposition which do not rely on simple homonymy would have to be implemented.

Chapter 4

Modelling and Generating HCPPs

In this chapter we describe a detailed and formal linguistic model for a certain class of joke and demonstrate automated generation of jokes using it. This thesis concentrates on jokes which make use of low-level ambiguity, specifically word sense and spelling ambiguity. The puns use common phrases that contain a homonym. Puns with homographs are not modelled because they are not successful oral jokes. To keep the topic focused and manageable, the jokes do not make use of substring level ambiguity. Occasionally puns that contain high-level ambiguities result from lucky accidents. Jokes with high-level ambiguities are not modelled because, as discussed in Section 2.2.1 this would require a computer with common sense and world knowledge. Specifically, we have chosen to model puns which make use of idioms (e.g. "kick the habit", "pass the buck", "jump ship") or commonly connected words ("knead the dough", "serial killer", "tip the waiter") all of which we will refer to as "common phrases". In the future, such common phrases could be detected by a statistical program that surveys common literature. An example of this kind of pun is:

John ate a loonie. Now he's passing the buck.

4.1 Why use common phrases?

Puns can be written without common phrases but we make use of them because they simplify the generation of sentences[1]. When a common phrase is used, one of the meanings of the homonym is already "built into" i.e. captured and expressed by the cliched phrase. Thus only one sentence

articulating the other meaning of this word in the common phrase needs to be generated from scratch. Without a common phrase we would have to generate two entire sentences to express the meanings of both the word and its homonym.

4.2 Components of HCPPs

HCPPs can be formally defined in terms of a "target phrase", a "base" and a "pivot", in a manner very similar to Lessard and Levison's model for Tom Swifties. For example compare the following Tom Swifty and HCPP:

a. *"We've struck oil" Tom said crudely.*
b. *John is violent. He is raising (razing) cattle.*

The simplified schema models in Figure 9 reveal that both kinds of pun obey a similar "grammar". As we have seen from Lessard and Levison's work, simple Tom Swifties contain a word called the pivot (the word "crudely" in example i) and the pivot in turn contains a word within it ("crude"), the "base" which is a homonym. In the HCPP, the common phrase "raising cattle" can be considered the pivot. Like the Tom Swifty, this pivot unit contains a base - an element which enjoys any kind of relation (meronymy, hyponymy, antonymy etc.) with an earlier part of the sentence. The base's meaning differs from the pivot's and hence the pivot's meaning is challenged and subverted. In our example, the base is the word "raising" whose homophonous meaning "razing" (defined as "completely destroying") is reinforced by the target phrase "John is violent". (Note: the word "raising" actually has two homonyms: "raising" as in lifting up and "razing" as in destroying. If the other homonym were used, the pun "John is lifting up cows. He raises cattle" could also be constructed). Thus the Tom Swifty and HCPP schemas are similar but their narrative strategies and templates are very different.

---

[1] For example: "I'm bored" yawned the piece of wood with a hole in it. Three meanings of the homophone bored are used here: bored (unchallenged), board (piece of wood), bored(drilled).

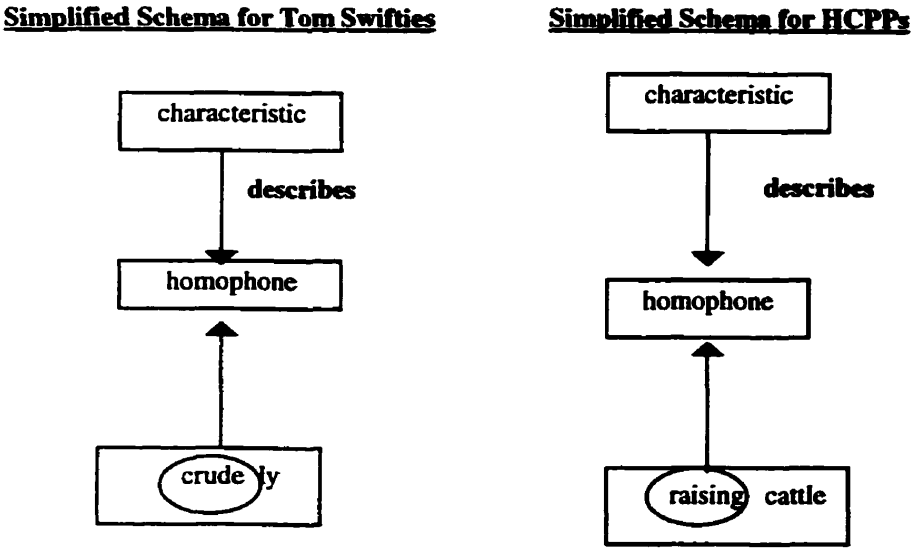


Figure 9: The similarity between Tom Swifties and HCPPs. The pivots appear in the boxes at the bottom, the bases in circles

4.3 Levels of complexity of HCPPs

The minimum requirement for generating homonym puns is a semantic relation between

a. multiple reverberations with parts or all of the common phrase:	*John stores his money in a mattress. He's making a down payment.*
b. reverberations involving parts of the common phrase and the meaning of the whole common phrase:	*John stores his money in a mattress so he has something to fall back on.*
c. reverberations with a substring or substrings of a word in the common phrase:	*John's girlfriend fell asleep while watching the Northern lights. He asked: "Does the aurora bore you Alice?"*
d. reverberations involving more than one common phrase per joke:	*"I can't believe a man of your calibre would do such a thing!" said Tom shooting off his mouth.*

Table 15: Some of the different kinds of bases in HCPPs.

a homonym of the base word and a word or phrase in the target sentence. Table 15 groups some possible HCPPs according to the kinds of relations that occur between the pivot and the target sentence.

4.4 Focusing on a subclass of homonym pun

This thesis will focus on puns that belong to categories 15(a) and (b). In order to render the algorithmic modelling clear and comprehensive, we further refine the focus to puns with the following characteristics:

- Most of the puns will consist of two sentences or a sentence and a sentence fragment. The pivot sentence will act as the punchline of the joke and so will always appear after the target sentence as in joke 2a. The sentences could be reversed to "John is passing the buck. He ate a loonie" but that would not be as satisfying. The sentence that ends with the common phrase resolves the built-up tension better.
- The pivot sentence/sentence fragment will contain a common phrase that in turn contains a homonym.
- The target sentence will always make some kind of reference to the homonym in the pivot sentence.
- "John" or "Joan" or a career person (such as a lawyer, nun, fisherman) or some kind of animal will be the subject of the first sentence and the pronoun for that entity ("He", "She") will be the subject of the second sentence.
- We will concentrate on common phrases with the syntaxes listed in Table 16.

The range of puns that meet these criteria is vast and spreads across both verbal and non-verbal types of humour. Table 17 illustrates this range by investigating part of the hierarchy of homonym puns that can emerge from a single common phrase "x kicks the habit".

Thus, jokes like a - d can be generated by our system but e and f cannot because this kind of complex cognition makes them too difficult. Although all of these puns would probably be classified as examples of verbal jokes because of their use of word-level ambiguity, puns e and f go beyond simple linguistic play and require reason and knowledge.

a	**Verb <optional article> noun** where the verb has a homonym that is also a verb.	**"tip the waiter", "jump ship"**
b	**verb phrase <optional article> noun** where the noun has a homonym that is also a noun.	**"kick the habit", "pour out his soul"**
c	**verb <optional article> noun** where the verb and the noun both have verb and noun homonyms respectively.	**"pass the buck", "kneads dough"**
d	**adjective noun** where noun has a homonym that is also a noun.	**"broken heart", "communist plot"**
e	**noun#1 noun#2** (compound noun) where noun#1 has a homonym which is also a noun.	**"soul mate", "night school"**
f	**adjective noun** where adjective has a noun homonym.	**"serial killer", "novel idea"**

Table 16: Some of the different syntaxes of the common phrases. Because we are performing linguistic transformations and generating sentences, it is useful to organize puns in terms of the syntax of the common phrase being used.

a	John attacks nun's clothes. He kicks the habit.	A simple translation pun that uses phrase type a. The direct object of the common phrase is a homonym so we find a synonym of its homophonous meaning and use a synonym or hyponym of the verb to create the target sentence.
b	John attacks nuns. He kicks the habit.	This is an improvement of (a). To generate this pun, however, the program would require a semantic net extensive enough to know that a habit is worn by a nun and that clothing (especially uniforms or clothing associated with a vocation) can be symbolic of a person.
c	The nun gave up smoking. She kicked the habit.	An example of an actual bad habit improves the pun by making it more subtle than pun (d). The improvement in (c) makes the pun harder to implement, however, because the database would need to have examples of bad habits. The juxtaposition of incongruities expressed in the pun - the idea of a nun smoking or having any kind of bad habit - is a comical one that adds to the humour of the joke but enabling the system to create this kind of opposition is very difficult - a sophisticated knowledge base and/or inferencing mechanism would be required. Interestingly, pun (c) was generated by our program, but the incongruity of a devoutly religious person smoking is an accidental bonus unplanned by the algorithm which yielded this joke. In other words, the incongruity in this pun will not necessarily appear in puns derived from other common phrases. In this example, it just so happens that the word "habit" and its homonym bring together two concepts (a nun and smoking) that are more opposed to each other than ones normally yielded by homonyms (for example, "air" and "heir").

d	The nun gave up a bad tendency. She kicked the habit.	
e	John used to attack nuns. But now he has kicked the habit.	This pun is ironic because John's giving up a violent tendency is described in violent terms and suggests that maybe John did not give up his violent predilections after all. This kind of irony requires complex cognition which computers currently do not possess.
f	The nun has given up her orders. She kicked the habit.	In this pun, the meaning of the common phrase resonates with the idea in the target sentence that the nun has given up something. In fact, because of this, it makes a disrespectful equation of the nun's devotion to God with an addictive tendency that should be shaken off. Giving up her vocation is equated with giving up some implicitly bad and trifling tendency. Also, the literal interpretation of "kick" and the homonym of "habit" (nun's clothes) taken together, form a symbol of the nun renouncing her religious orders. Literally "kicking" something often means showing that thing disrespect (unless it is a soccer ball) and so there is the suggestion here that the nun does not respect her former calling. The computer would therefore need to possess knowledge about the symbolism associated with an act such as kicking, and would have to be sophisticated enough to understand the following nuance: in this pun the habit no longer represents the nun as it did in example (b) but instead symbolizes her orders. This linking of things - habit and religious calling, habit and nun - are insights which the computer would have to possess. In other words, the computer would have to possess encyclopedic knowledge about the world.

Table 17: Part of a range of homophone puns that can emerge from a single common phrase "x kicks the habit".

4.5 Required Lexical Database Elements

A number of different algorithms have been written to generate HCPP's because common phrases with different syntaxes require different algorithms. All of these algorithms make use of a lexicon which contains certain semantic relations between nouns, noun phrases, verbs, verb phrases and adjectives. Table 18 lists the semantic relations the generator uses to construct HCPPs. A subclass of an HCPP will make use of a certain set of relations while another subclass will require a different set. Thus not all the lexical slots have to be filled for a particular common

SLOT	Used With	Allowed Values
class	np, noun	The immediate superclass to which the entered lexeme belongs. e.g. (**deer, animal**)
why_x	verb	The answer to the question "Why would x verb y?" and the answer only involves x. e.g. (**raze, "x is violent"**)
why_xy	verb	The answer to the question "Why would x verb y?" and the answer involves both x and y. e.g. (**raze, "x dislikes y"**)
homonym	noun, verb, adjective	e.g. (**heart, hart**), (**cure, cure**), (**serial, cereal**)
synonym	np, noun, vp, verb, adjective	e.g. (**cattle, cows**)
per_or_animal	noun, np, verb, vp	A person or animal associated with this lexeme. e.g. (**cattle, farmer**).
inact_verb	np, noun	A verb lexeme. Something you typically do to the thing. e.g. (**deer, hunt**), (**mess, made**)
approp_adj	(adjective, noun) pair	An analogous adjective for the given class. e.g. ((broken, animal), **injured**)
prep_assoc	np, noun	A preposition associated with the noun lexeme. e.g. (**tent, in**), (**wharf, on**)

Table 18: Lexical relations used to generate a class of riddles.

phrase (and the words that make it up) in order for a joke to be generated. Figure 10 demonstrates which relations are relevant to which syntactic categories of words.

4.6 The Schemata

Figures 11 through 21 demonstrate the various schemata for HCPPs. The syntax of the word phrase acted upon by the schema is shown in the topmost box. The "(h)" symbol appearing after a word category indicates that the word is a homonym. For example, Figure 11 shows Schema #1 which generates puns from verb-noun phrases in which the verb is a homonym.

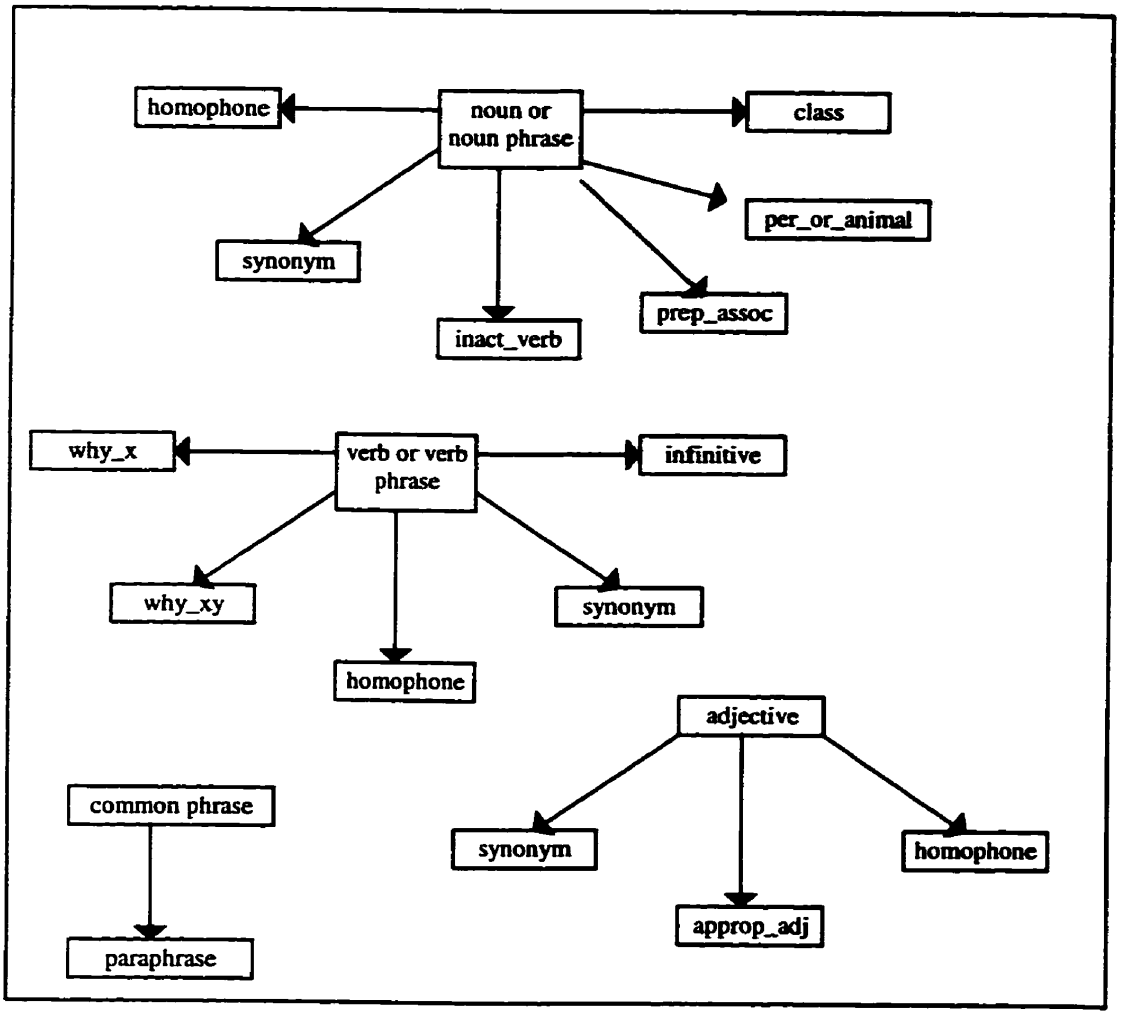
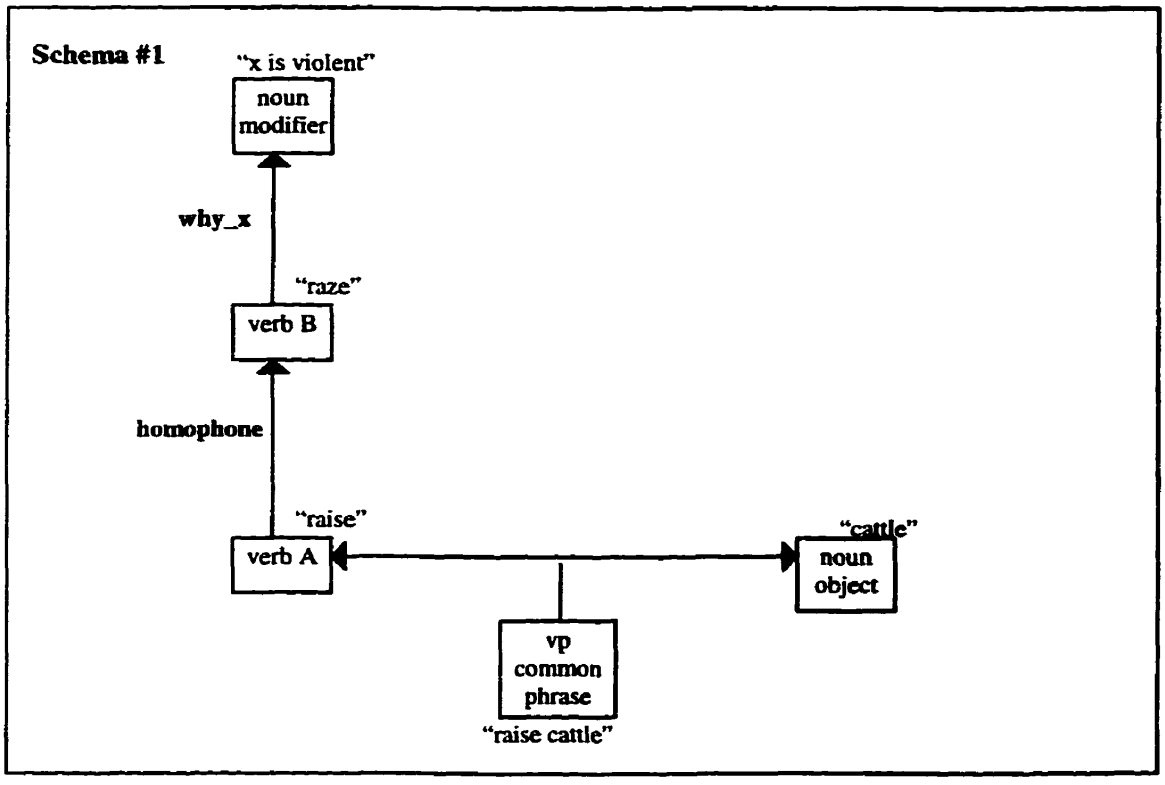


Figure 10: The relations required for different word or phrase categories. Most schemas make use of only a handful of relations. In other words, not all the relations are required to generate a particular kind of joke.

verb(h) - noun

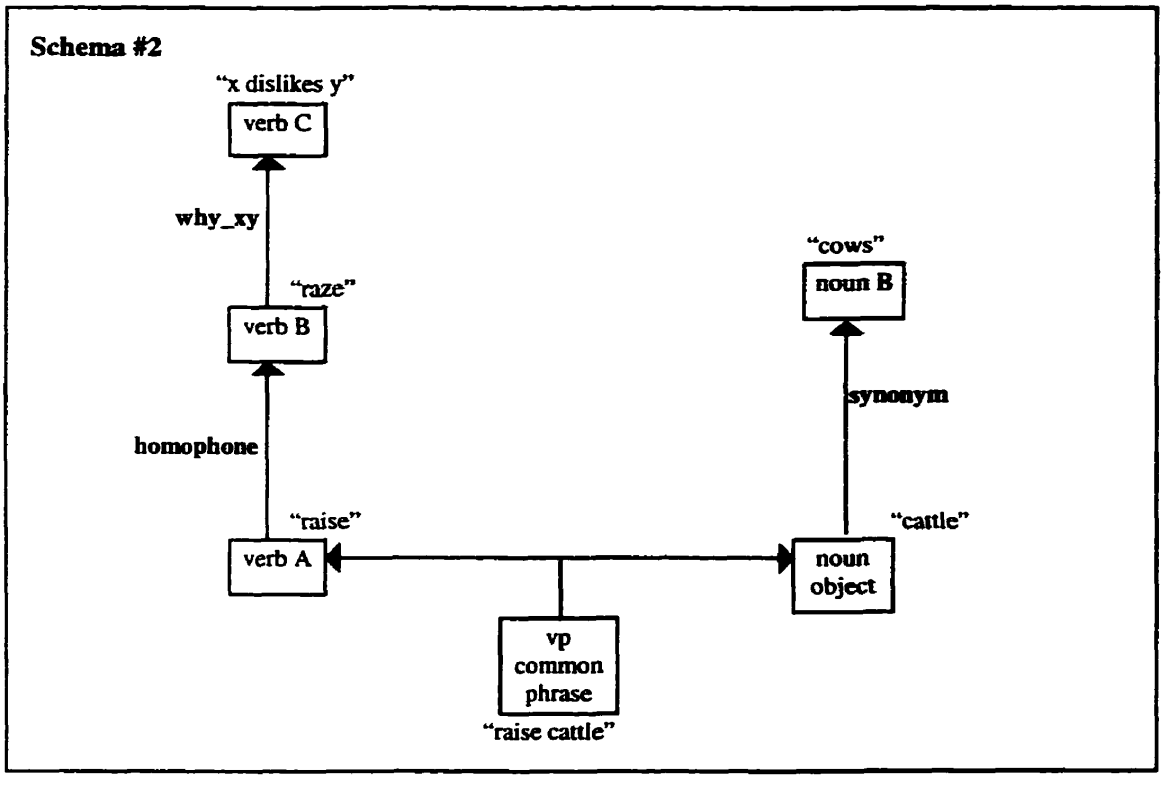


Template for Schema #1

"The " <nounA> "is" <noun modifier> "."
"He " <common phrase> "."

Figure 11: The why_x relation asks the question: "Why would x raze y". A possible answer for this relation might be "x is violent". The answer for this relation involves only a description of x and does not involve y. Examples of jokes produced by this schema is: "John is violent. He raises cattle". "John is compassionate. He cures the meat". "John is perverted. He flashes his lights".

verb(h) - noun



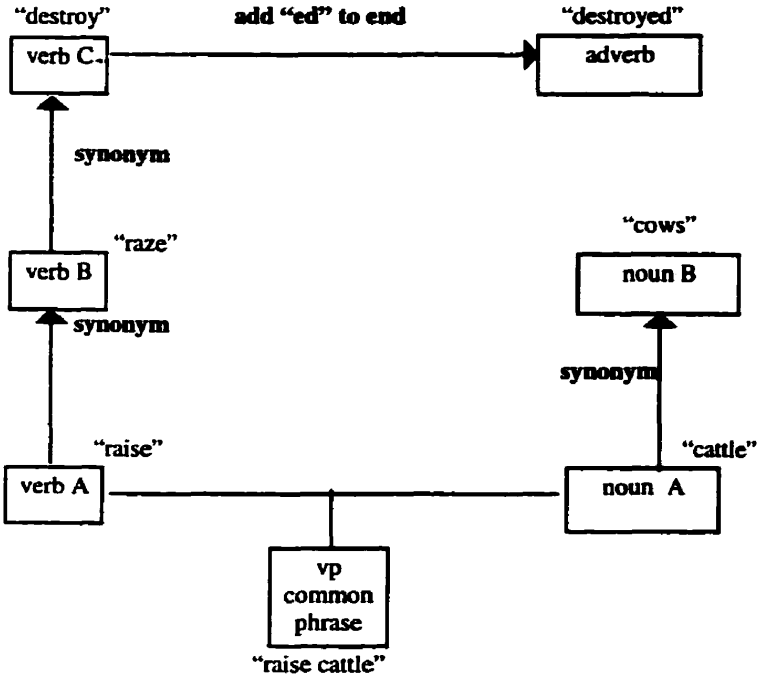
Template for Schema #2

"John " <nounA> <verb C> <noun B>"."
"He " <common phrase> "."

Figure 12: This schema is almost identical to schema #1 except the why_xy relation is used instead of the why_x relation and a synonym of the noun object is required. The kinds of jokes produced by this schema and template are: "John dislikes cows. He raises cattle". "John pities the animal flesh. He cures the meat". "John is attracted to electromagnetic radiation. He flashes his lights".

verb(h) - noun

Schema #3

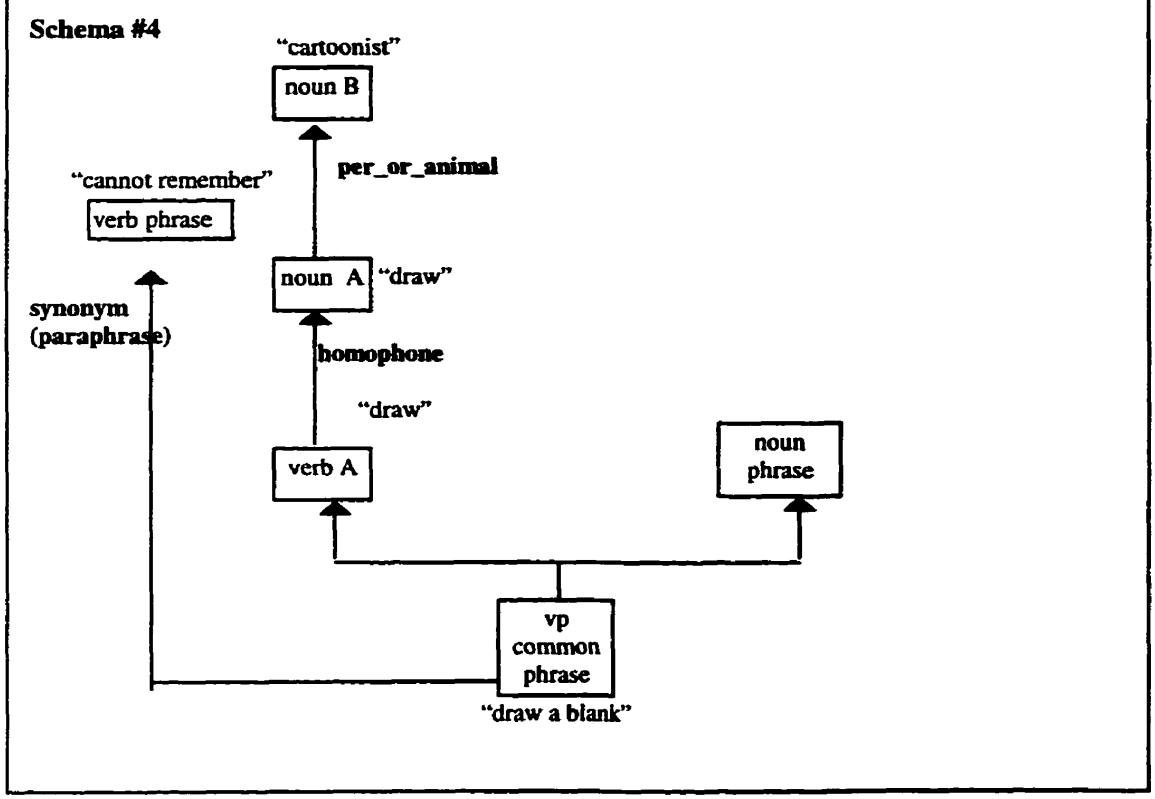


Template for Schema #3

"The " <noun B> <aux> <adverb >"."
"John" <common phrase> "."

Figure 13: The kinds of jokes produced by this schema and template are: "The cows are destroyed. John raises cattle" or "The animal flesh is healed. John cures the meat".

verb(h) - noun



Template for Schema #4

"The " <noun B> <verbphrase> "."
"She " <common phrase> "."

Figure 14: The common phrase has a verb in it which is a homophone. Some examples are: "The cartoonist cannot remember. She draws a blank". "The pervert is signaling. He flashes his lights "The cook overextended himself. He strained himself". Using the per_or_animal link with verbs is often effective because what a person does often says a lot about who that person is. Using per_or_animal with objects is probably less successful, however, because a person interacts with many objects in this world which do not have a lot to say about her. Of course some objects are related to a person, some are even symbolic - for example a doctor and her scalpel, a mail carrier and letters. But for the most part, it is probably safer to say that the verb being done by a subject in a sentence is more closely related to that subject than the object which the subject is interacting with.

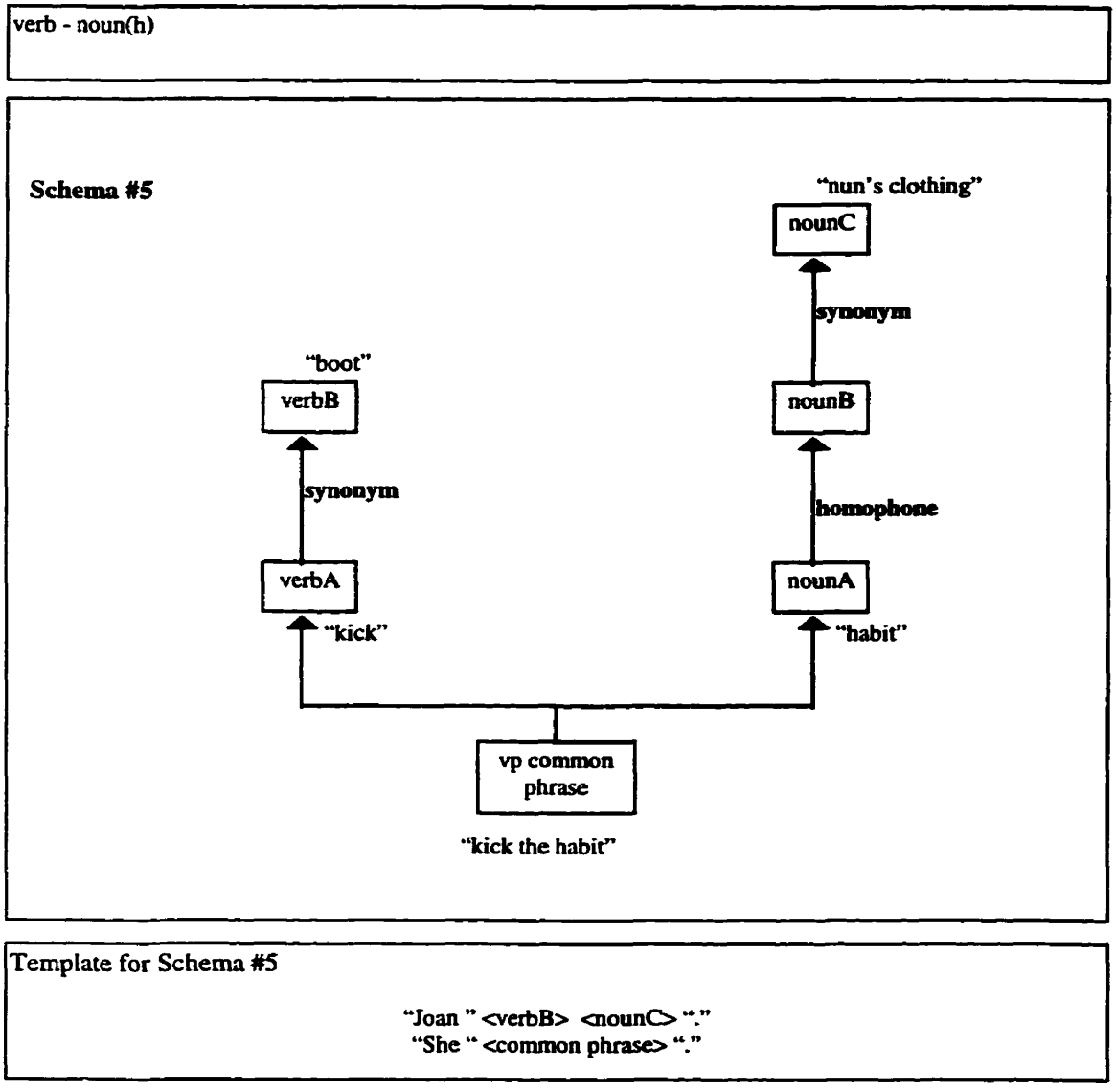
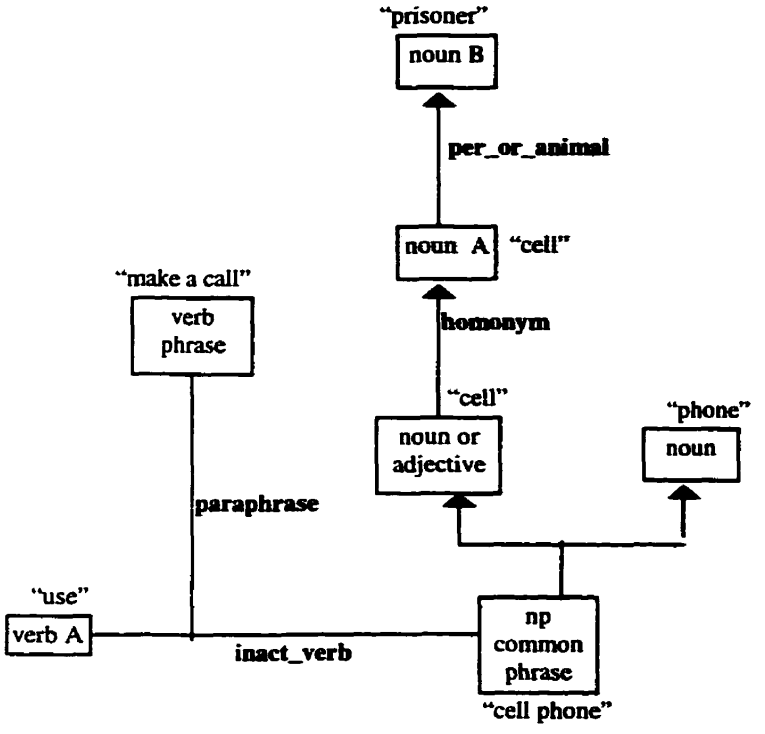


Figure 15: Simple translation puns. Some examples are: "Joan boots the nun's clothing. She kicks the habit". "Joan owns a signal. She has flare". "John created a hard candy. He made a mint". "Joan handed over a compartment. She gave berth".

1. noun(h)-noun
2. adj(h)-noun where hom(adj)->noun

Schema #6

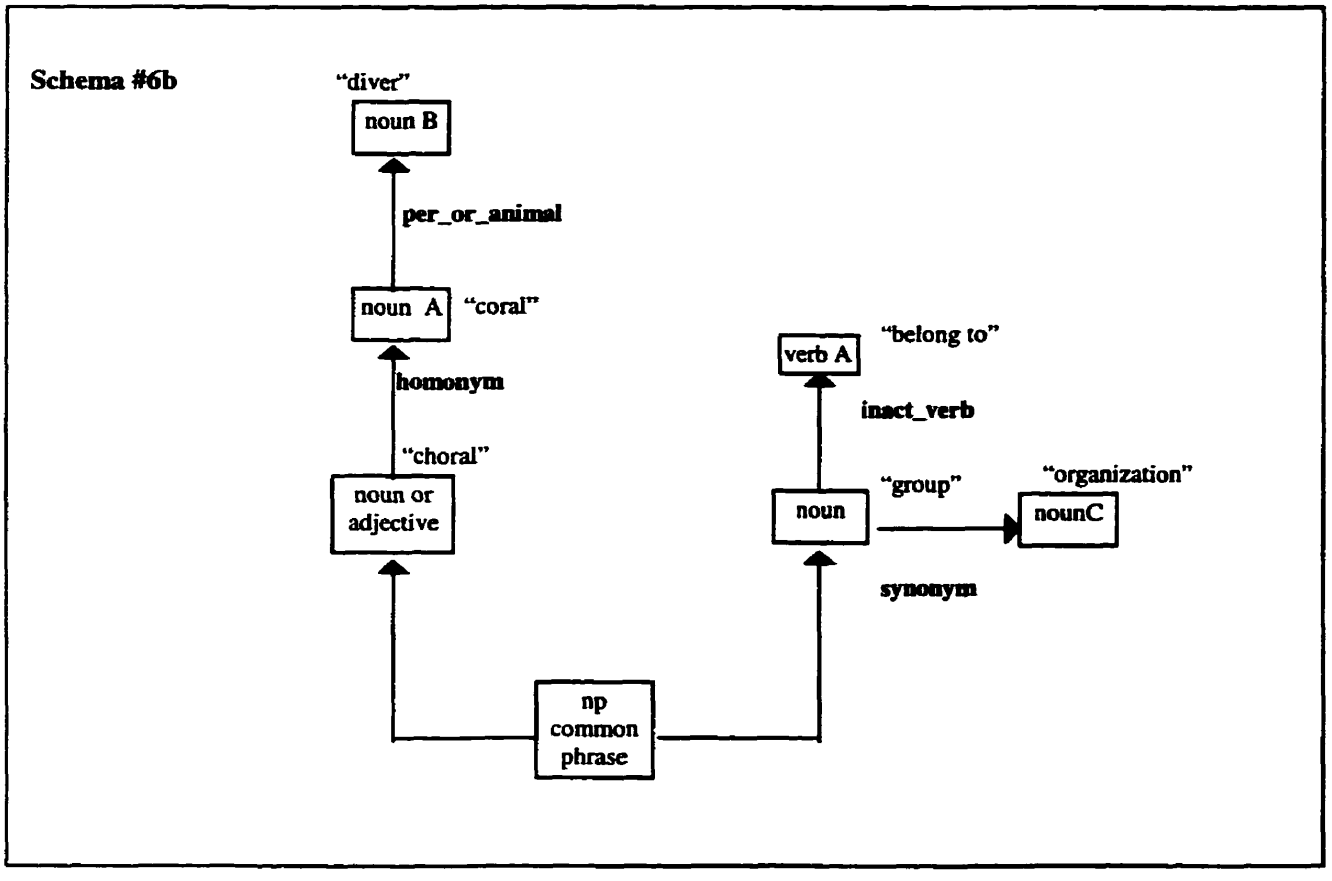


Template for Schema #6

"The " <noun B> <verb B> "a" <noun C>"."
"She " <verb A> <a> <common phrase> "."

Figure 16: The common phrase does not contain a verb. One of the words in it (an adjective or noun) is a homophone. Some examples are: "The prisoner makes a call. He uses a cell phone" or "The electrician reads a lot. She keeps up with current events". *If nounA is already a career person or an animal then the value for the relation per_or_animal is just a synonym of nounA.

1. noun(h)-noun
2. adj(h)-noun where hom(adj)->noun



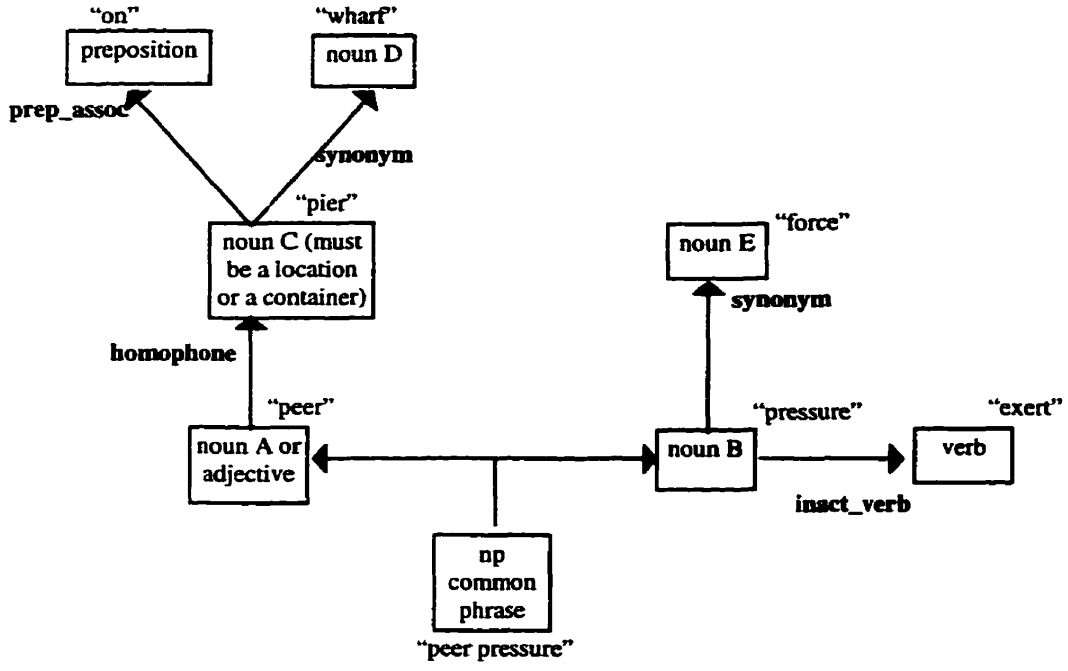
Template for Schema #6b

"The"<nounB> <verb A> {a/an} <nounC> "."
"A " <np common phrase> "."

Figure 17: The common phrase is a noun phrase (either an adj-noun or a compound noun). Either the adjective or the noun is a homophone. An example is: "The diver belongs to an organization. A choral group". If nounA is already a person or an animal then the value for the relation per_or_animal is just a synonym of nounA.

1. noun#1(h)-noun#2 where hom(noun#1) is a place or container.
2. adj(h)-noun where hom(adj)->noun and is a place or container.

Schema #7

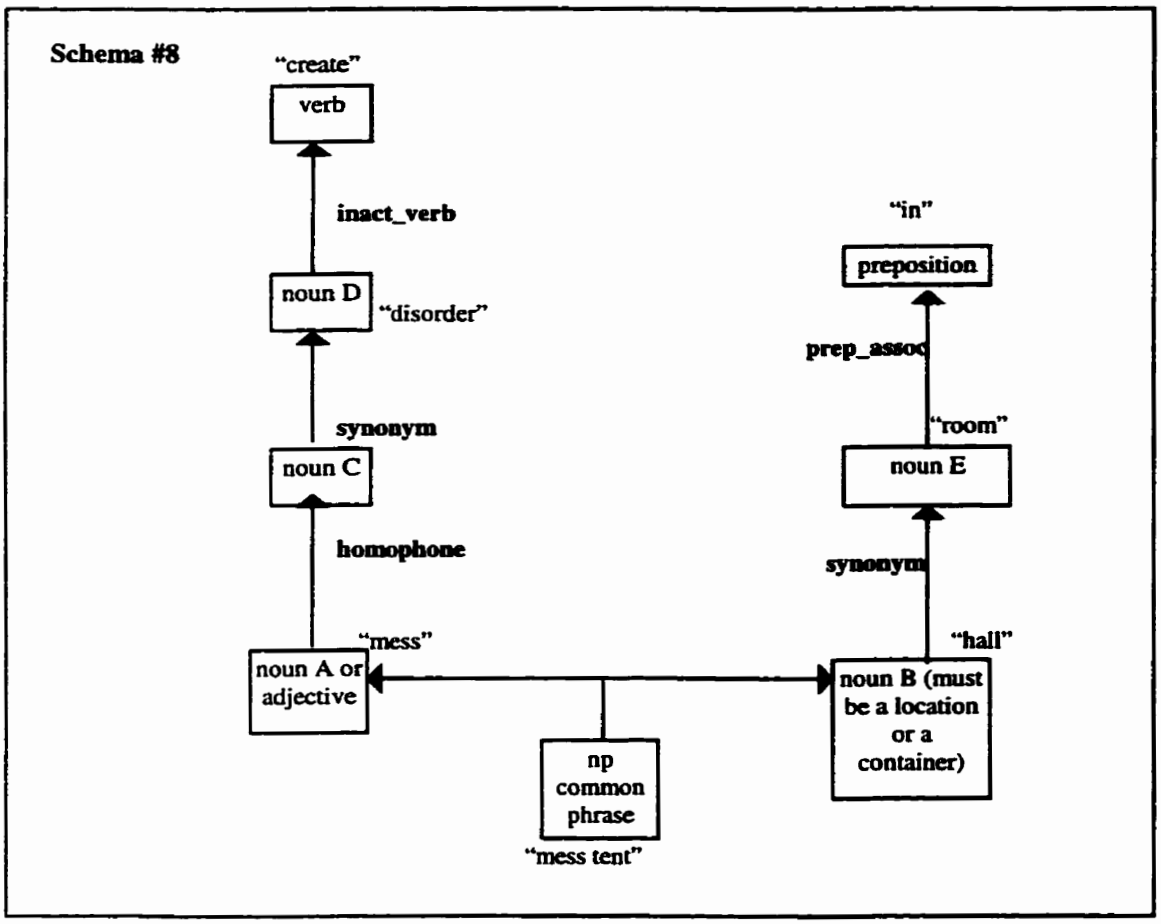


Template for Schema #7

"John" <verb> <noun E> <preposition> "the" <noun D>"."
"{A/An} <common phrase> "."

Figure 18: The kinds of jokes produced by this schema and template are: "John exerts force on the wharf. Peer pressure" or "John sees the person in a burial spot. A grave man".

1. noun#1(h)-noun#2 where noun#2 is a location or container.
2. adj(h)-noun where hom(adj)->noun and that noun is a location or container.

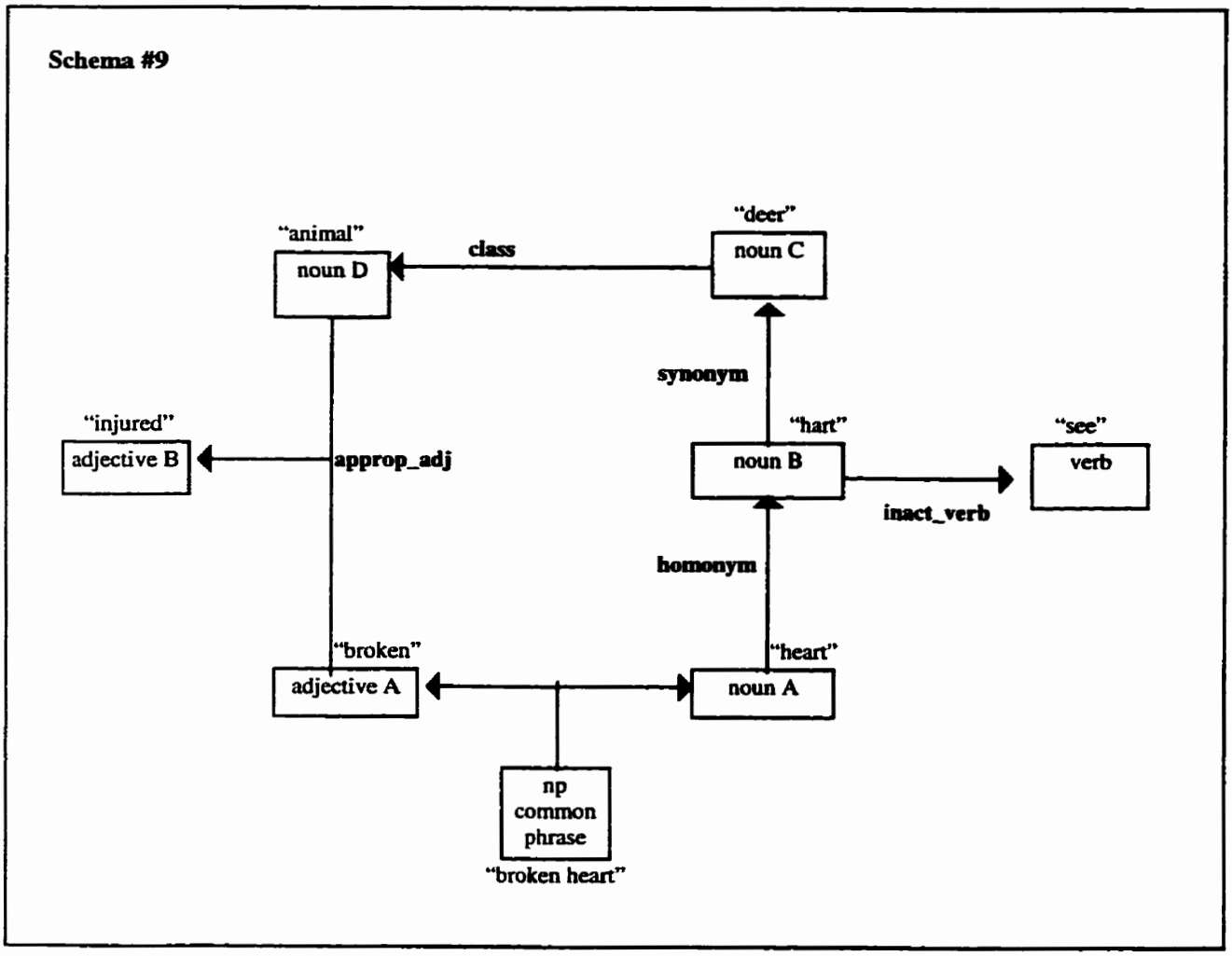


Template for Schema #8

"John" <verb> <noun D> <preposition> <noun E>"."
{A/An} <common phrase> "."

Figure 19: The kinds of jokes produced by this schema and template are: "John creates disorder in the room. A mess hall" or "John reads about a medieval warrior at the place of learning. A knight school" or "John combs the horse's hair on the pole. A main mast".

adj(h)-noun where hom(adj)->noun



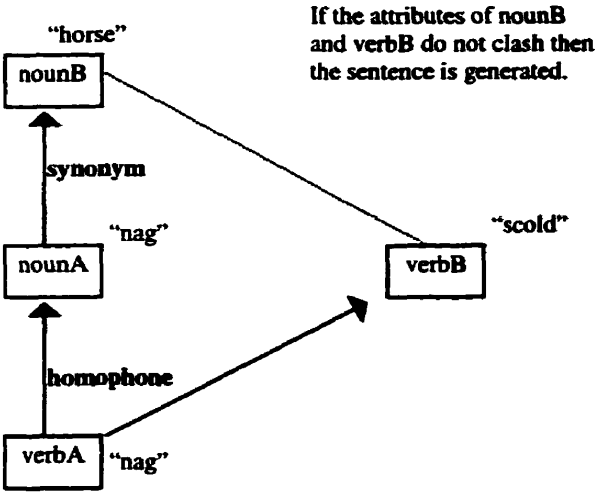
Template for Schema #9

"John" <verb> {"a/an"}<adjective B> <noun C>"."
{"It/He/She"}"is" {"a/an" } <common phrase> "."

Figure 20: The kinds of jokes produced by this schema and template are: "John sees an injured deer. It is a broken heart". If the adjective is appropriate as it is then a synonym of it is used. For example, "John is a warm descendant. He is a hot heir".

verb(h) -> noun

Schema #10



Template for Schema #10

"John" <verbB> "the" <nounB>.
"He" <verbA> "it" "."

Figure 21: The kinds of jokes produced by this schema and template are: "John scolds the horse. He nags it". "John sells the bird. He hawks it". Attributes such as animate, inanimate, tangible, intangible are given to nouns and verbs in order to procure some semantic coherence between them. For instance , with the given attributes, some absurdities such as "John drinks the distance. He laps it up" will not be allowed because "distance" is an intangible and the verb "drink" requires a tangible. But the list of attributes is incomplete so many semantic clashes can still occur.

Chapter 5

Implementation

This chapter describes a practical implementation of three of the schemata introduced in Chapter 4. A large portion of the work was the preparation of the lexical database from which the jokes were drawn. Section 5.1 describes that process and Section 5.2 reviews briefly how the schemata were implemented in VINCI.

5.1 Populating the Lexical Database

The whole procedure for populating the lexicon could be completely automated if a sophisticated semantic network of words and their relations were to exist - the kind of knowledge base which will ultimately need to be constructed for truly automated natural language generation and understanding. This knowledge base does not yet exist, however. Indeed one of the goals of this thesis is to discover some of the information which would need to be available to allow comprehensive artificial generation and understanding of natural language. The steps in the procedure (see Figure 25) have been done manually for this thesis because the needed digital lexical resources were not available.

A lexicon is "a set of records (lexical entries) which describe the words of the language, the word categories they belong to, and a variety of other ... information used in the generation process" [LL96]. Figure 21 demonstrates an example of a part of a lexicon and shows that each line in it holds information about a single word or phrase. The information includes: the lexeme of the word, its category, its attributes, and various relations.

5.1.1 The reserved fields

A VINCI lexicon consists of a collection of records, each of which is a sequence of fields. The first six fields of every lexical entry are reserved for certain kinds of information while subsequent fields can be defined in any way the user chooses. The six reserved fields are shown in Figure 22 and are listed here:

- Field 1 holds the lexeme of the word - i.e. a unique identifier for each word or phrase.
- Field 2 contains the category of the word - i.e. whether it is a noun, adjective, verb, determiner or common phrase.
- Field 3 holds attributes or characteristics of the word. For example, in Figure 2, "trunk" (as in car compartment) has the attribute 'location' and the class "Number" which contains the values "singular" and "plural". This class allows each lexical item to be either singular or plural - morphology rules then produce the appropriate forms, with or without "-s", as required.
- Field 4 can be used to indicate the frequency with which a user would like VINCI to include that word in a sentence. This feature was not necessary for the joke generator, however, so it was left blank.
- Field 5 holds information about how to make the word plural (if the word is a noun) or how to make it agree with third person subjects (if the word is a verb).
- Field 6 holds rules for a second morphology pass. This allows leaves in the tree to be compared to each other and to be transformed if need be. For example if the words "a" and "ox" are next to each other in a sentence, then "a" is changed to "an" because the word following it starts with a vowel.

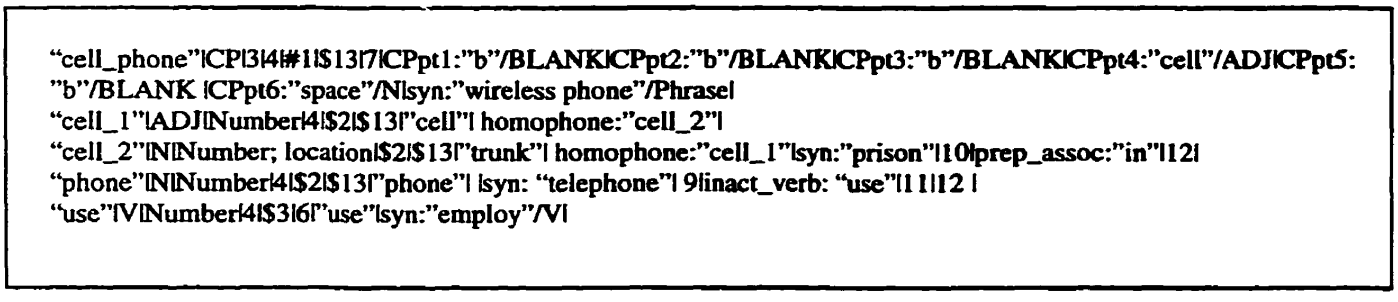


Figure 22: Example records from a VINCI lexicon. In this Figure, an integer indicating the field number has been placed in empty fields.

5.1.2 HCPP Lexical entries and their relations

Some of the kinds of words appearing in the lexicon are listed in Table 19. Certain relations to

nouns or adjectives that have homonyms which are locations	"carol", "birth"
nouns or adjectives that have homonyms which are not locations	"beach", "serial"
nouns that are not homonyms but are locations	"ship", "hill"
nouns that are neither homonyms nor locations	"singing", "godmother"
verbs that are homonyms	"raise", "flash"
verbs that are not homonyms	"visit", "give"
adjectives that are not homonyms.	"silly", "broken"

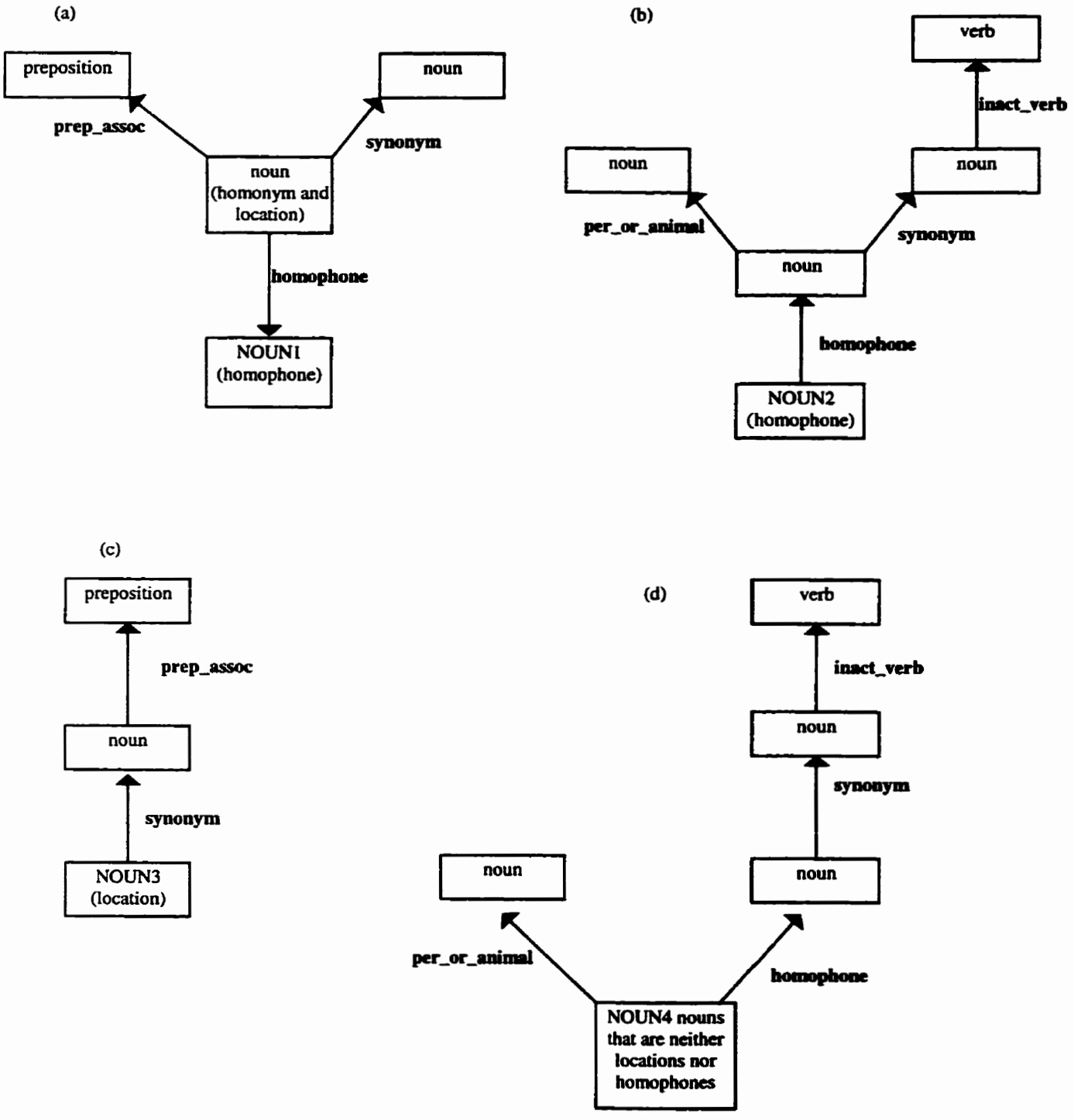
Table 19: The different kinds of words appearing in the lexicon.

field	allowed elements	comments
1	the lexeme.	
2	N|V|A|CP|PREP|ART|PHRASE|PUNCT|BLANK|PRON	Categories of words and phrases.
3	Location, Number, plur.	Attributes of the word.
4	relative frequency	Not used by our generator.
5	morphological rule for pass#1: $2, $3,	Rules that allow subject verb agreement and ensure that nouns have the proper singular and plural forms.
6	morphological rule for pass#2: $13	
7-n	anything	User-defined attributes.

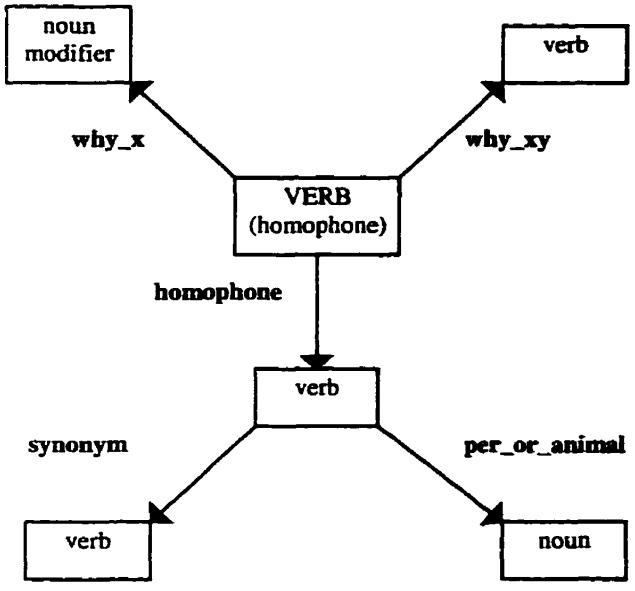
Table 20: A listing and description of the reserved fields in VINCI that are common to each entry in the lexicon.

these words are necessary for the generation of jokes (see section 4.7). Figure 23 demonstrates which relations are pertinent to the different kinds of words. These semantic relations will appear in certain fields in the lexicon. All words belonging to the same word category (for example all nouns) will have the same number and kinds of fields. These fields will be ordered in the same way but not all of them will be filled. For example both a noun that is a homonym and a noun that is not will have a specific field in their lexical entry reserved for homonym pointers. But

only the word that actually has a homonym will have a value for this field. The reason for creating the lexicon this way was to keep it simple and consistent.



(e)



(f)

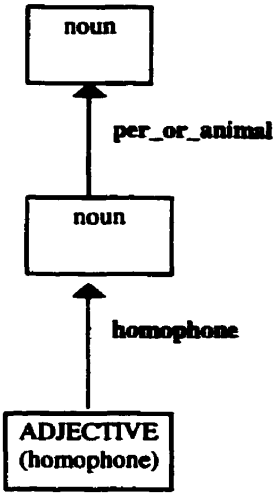


Figure 23: The different relations required for (a) nouns which are homophones and locations (b) homophone nouns (c) nouns that are not homophones but are locations (d) nouns that are neither locations nor homophones (e) verb homophones (f) adjective homophones.

Tables 21-23 show where the semantic relations to nouns, verbs and adjectives appear in the lexicon.

field	the type of information appearing in the field
7	the surface form of the lexeme (i.e. how it will actually appear in a sentence).
8	homonym
9	synonym phrase
10	per_or_animal
11	inact_verb
12	prep_assoc

Table 21: The lexical fields for nouns. Different schemata will make use of different fields. Questionnaires and dictionaries were handed out to volunteers and they supplied the various relations to the words making up the common phrases.

field	the type of information appearing in the field	comments
7	the surface form of the lexeme	
8	homonym	
9	syn	The synonym of the verb goes here. Entries in this field may be single verbs or they may be verb phrases consisting of a verb and a preposition. For example let us say a lexical entry is "peer". A synonym of this might be "look at". The whole phrase "look at" would appear here.
10	verb	Let us say there is a lexical entry for the verb phrase "look at". To conjugate the verb in this verb phrase, and have it agree with subjects, the verb needs to be isolated so it appears in this phrase. If the lexical entry is simply a verb, the verb appears here.
11	preposition {optional}	For the lexical entry "look at", the preposition "at" would appear in this field. If the lexical entry is simply a verb, then this field is blank.
12	per_or_animal	A person or animal associated with this lexeme. e.g. (**heal, doctor**).
13	why_x	example: "is caring"
14	blank	reserved in case why_x needed to be split up.
15	why_xy_verb	example: "want"
16	why_xy_rest	example: "to be better"

Table 22: The lexical fields for verbs.

field	the type of information appearing in the field
7	the surface form of the lexeme (i.e. how it will actually appear in a sentence).
8	homonym (it is a noun)

Table 23: The lexical fields for adjectives.

5.2 Building the Lexicon

When building the lexicon, the utmost care was taken to ensure that it be "general and neutral" i.e. not humour-specific [BR94, 13]. In other words we were careful that information was not put in the lexicon with particular jokes in mind. The following sections describe the steps in creating the lexicon and the whole procedure is summarized in Figure 25.

5.2.1 Generate the Initial List

First of all, 70 adjectives that have noun homonyms were taken from a list of homonyms on the web [Co99]. The list contained only homonyms that sound the same but are spelled differently (i.e. homophones). 7 obscure words were taken off the list, leaving 63 homonyms[1]. For the sake of variety, we also wanted adjective homonyms that sound the same and are spelled the same (i.e. homographs) so we consulted another source ([Fr66]). In 150 entries, however, only three adjectives that had noun homonyms were found: "gross", "major" and "minor". Nonetheless, these 3 were added to the list making a total of 66 adjective homonyms. From the same homonym list on the web[Co99], we then gathered 70 nouns that have noun homonyms. 9 of these were obscure and were taken off the list, leaving 61 noun homonyms[2].

The three homographs (for example "major" (the adjective) and "major" (the noun meaning "military person") have numbers ("_1" and "_2") appended to them so that they can be distinguished from each other. The resulting unique identifier (the lexeme) is then put at the head of a new line in the lexicon (field 1) because every word or phrase in the lexicon gets a single line in which information about it is recorded. Homophones (for example "hair" and "hare) do not require numbers added to them and hence are written as they appear in the homophone list. The category of the word is then recorded in field 2 and the class "Number" is put in field 3 (unless the homophone is a plural noun, in which case "plural" is written in field 3, indicating that the word does not have a singular form). The symbol for the morphology rule that transforms nouns into their proper singular or plural form is "$2" and is put in field 5 of each noun. Similarly, "$3", representing the morphology rule for verbs is placed in field 5 of each verb. A rule labeled "$13" for the second morphology pass is put into field 6 of the article "a". This rule changes the "a" to "an" if the word following it in a syntax tree starts with a vowel. Field 7 is

---

1The adjective homophones deemed obscure were "bundt", "faux", "grayed", "gnu", "plumb", "rood" and "mien".

filled with the near surface form of the homophone - i.e. the way the word will appear on the screen ("near" surface form because an "s" might be added to the string in field 7 if the word's plural or third person present form is required). The "homonym:" tag along with the word's homophone is then put in field 8. And the "syn:" tag along with a synonym of the word are put in field 9. These synonyms were provided by volunteers. Field 4 is filled with a blank because it relates to a feature of VINCI which is not required for our joke generator.

Thus, let us say the words "trunk" and "raise" (and their homonyms) were two of the words randomly chosen from the homonym list to appear in the lexicon. After their insertion, the lexicon would look like Figure 24.

```
"trunk_1"|N| | |#7|$13|"trunk"|homophone:"trunk_2"|syn: "luggage compartment of car"|
"trunk_2"|N|Number| |$2|$13|"trunk"|homophone:"trunk_1"|syn: "elephant's nose"|
"raise"|V|Number| |$3|$13|"tip"|homophone:"raze"|syn1: "bring"; syn2: "up"|
"raze"|V|Number| |$3|$13|"tip"|homophone: "raise"|syn1:"destroy"; syn2: ""|
```

Figure 24: the lexical entries for four homophones after step 1 has completed.

5.2.2 Remove unuseful homonyms and add common phrases to the lexicon.

The second step for creating the lexicon involved picking phrases from the Oxford English dictionary[3]. A word from the list of homonyms (5.2.1) was looked up in the dictionary and the

---

[2]The noun homophones deemed obscure were: "ewe", "eyelet", "gnus", "knaught", "lien", "mettle", "plait", "rheum", "whit".

[3] We tried using a program on the web called PhraseFinder[PF99] which takes a word as its input and outputs common phrases in which that word appears. For example if the word "fair" (a homophone) is input to PhraseFinder, the list of common phrases output is: "faint heart never won fair lady"; "fair and aboveboard"; "fair and square"; "fair crack of the whip"; "fair to middling". But this program is incomplete and failed to find many phrases that appear in the OED. Once its database base has expanded (users of the program can add to its database), however, it could be used for automating the creation of the lexicon.

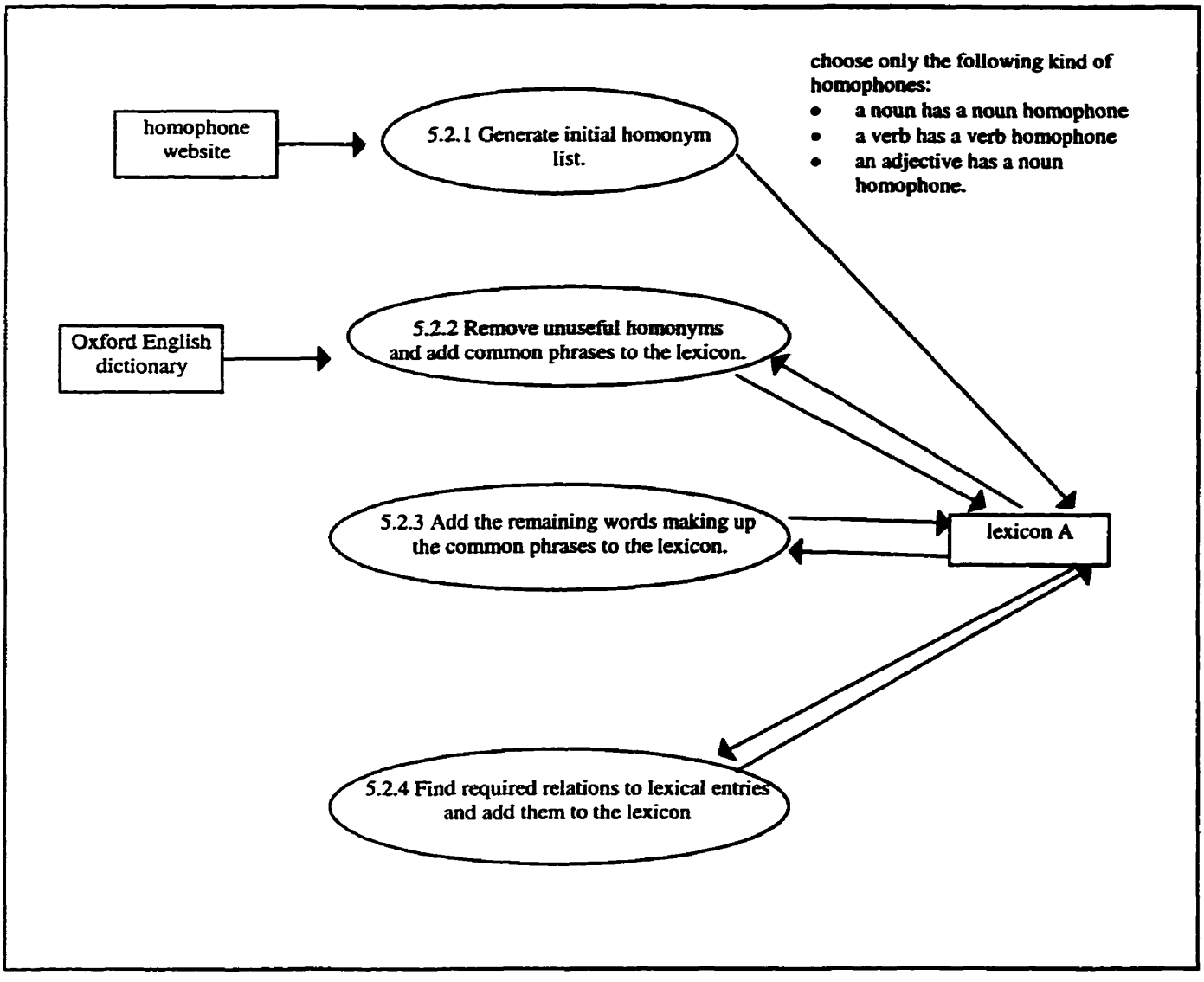


Figure 25: The first steps in creating the lexicon. A list of homophones is compiled from a listing on the web and is put into the lexicon. Then a list of common phrases is derived from those homophones using a dictionary. The resulting list of common phrases is added to lexicon A. The words appearing in the common phrases which do not have their own entries in the lexicon are then added to it.

first phrase to occur in the entry for that homonym which fit our syntactic criteria was picked. This was done for each of the homonyms. Care was taken to ensure that a phrase which ends up in the lexicon was not selected based on foreknowledge that it would yield good results. If a

common phrase with the proper syntax (described in section 4.4) could not be found for a word or for that word's homonym, both words were deleted from the lexicon. If multiple common phrases were found for a homonym, only the first one occurring in the dictionary was chosen[4]. When a common phrase meeting the syntactic criteria was discovered, the word and its homonym remained in the lexicon and the phrase and certain information about it were added. For example, let us say common phrases containing the word "trunk" (as in "luggage compartment of car") and the word "raise" were found. Figure 26 shows what our sample lexicon might look like once the second step has been completed.

From the 66 adjective homonyms, 41 common phrases were found but, for the sake of limiting the size of the lexicon (which is very time consuming to build by hand), only the first 20 were put in it. From the 61 noun homonyms, we stopped searching for common phrases when over 100 had been found. Upon examining this list we found that 16 of them were noun phrases in which one of the words was a location or container. Thus these 16 phrases were placed in the lexicon because schemata 7 and 8 could make use of them. This did not constitute a violation of our rule that the lexicon is to be built with no particular jokes in mind, however. If all 100 common phrases had been entered into the lexicon - as they would be for a system designed for comprehensive natural language generation - VINCI would have found the 16 relevant phrases anyway and ignored all the others when executing the algorithms for schemata 7 and 8. Thus by placing the 16 common phrases with noun locations in the lexicon, we were not restricting VINCI's search for appropriate noun phrases for schemata 7 and 8 but were simply saving ourselves from having to do a lot of unnecessary typing. Figure 26 shows that we have classified all common phrases to be of the same

type. In spite of their syntactic differences - "trunk space" is a noun phrase and "raise cattle"

---

[4] For example the homophone "bare" appeared in numerous common phrases such as "bare truth", "bare facts", "bare majority", "bare necessities" etc. but only "bare truth", the first phrase to occur in the dictionary's entry for this word, was placed in our lexicon.

```
"trunk_1"|N| | |#7|$13|"trunk"|homophone"trunk_2"|syn: "luggage compartment of car"|
"trunk_2"|N|Number| |$2|$13|"trunk"|homophone:"trunk_1"|syn: "elephant's nose"|
"raise"|V|Number| |$3|6|"tip"|homophone:"raze"|syn1: "bring"; syn2: "up"|
"raze"|V|Number| |$3 |6|"tip"|homophone: "raise"|syn1:"destroy"; syn2: ""|
"trunk_space"|CP|3|4|#1|$13|7|CPpt1:"b"/BLANK|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"
/BLANK|CPpt5:"trunk_1"/N|CPpt6:"space"/N|
"raise_cattle"|CP|3|4|#1|6|7|CPpt1:"raise"/V|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"/BLA
NK|CPpt5: "b"/BLANK|CPpt6:"cattle"/N|
```

Figure 26: Adding common phrases to the lexicon.

consists of a verb and a noun - both phrases are classified as CPs. The type CP is a kind of ordered superset which consists of the following components: verb, preposition, article, adjective, noun#1, noun#2. A particular phrase will contain all of these components but only a subset of them will have values. For example all the components of the phrase "trunk space" will be blank except for the last two: noun#1and noun#2. The phrase "raise cattle", however, has values for the verb and noun#2 components but the remaining ones are blank. Classifying all common phrases as the same type (in spite of their varying syntaxes) was not an attempt to model the way human beings regard common phrases: we are not suggesting that people think of common phrases in this way. Nor was grouping them under one label motivated by reasons of easing implementation. We were simply trying to make the point that any kind of common phrase containing a homonym can act as the seed of a joke - that all common phrases are identical in this respect. Table 24 summarizes the kinds of common phrases our joke generator acts upon.

5.2.3 Add the remaining words making up the common phrases to the lexicon.

So far, the only entries in the lexicon are homonym verbs, nouns and adjectives and the common phrases which contain them. (For example, see Figure 26). The next step in developing a lexicon

for our joke generator involves creating lexical entries for the words in the common phrases which do not yet have their own lines in the lexicon. These words will be nouns, adjectives, verbs and determiners. This step is important because in order for VINCI to make use of these

	verb	preposition[5]	article	adjective	noun	noun	type of expression	example
schema #1,2,3,4,5	yes	no	optional	no	no	yes	verb - noun phrase	"raise cattle" or "kick the habit"
schema #6,9	no	no	no	yes	no	yes	noun phrase	"broken heart"
schema #6,7,8	no	no	no	no	yes	yes	compound noun	"peer pressure"

Table 24: The types of phrases handled by the schemata.

words and others associated with them, they must have their own separate lexical entries. Thus in our running example, the words "cattle" and "space", along with information for their first nine fields are added to the lexicon in this step. Figure 27 demonstrates the result.

```
"trunk_1"|N|Number| |#7|$13|"trunk"|homophone"trunk_2"|syn: "luggage compartment of car"|
"trunk_2"|N|Number| |$2|$13|"trunk"|homophone:"trunk_1"|syn: "elephant's nose"|
"raise"|V|Number| |$3|6|"tip"|homophone:"raze"|syn1: "bring"; syn2: "up"|
"raze"|V|Number| |$3 |6|"tip"|homophone: "raise"|syn1:"destroy"; syn2: ""|
"trunk_space"|CP|3|4|#1|$13|7|CPpt1:"b"/BLANK|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"
/BLANK|CPpt5:"trunk_1"/N|CPpt6:"space"/N|
"raise_cattle"|CP|3|4|#1|6|7|CPpt1:"raise"/V|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"/BLA
NK|CPpt5: "b"/BLANK|CPpt6:"cattle"/N|
"space"|N|Number| |#7|$13|"space"| |syn: "room"|
"cattle"|N|plurl |#7|6|"cattle"| |syn:"cows|
```

Figure 27: The lexicon after step 5.2.3.

5.2.4 Find certain relations to lexical entries and add them to the lexicon

The relations for the various nouns, adjectives and verbs were collected by means of volunteers who answered a questionnaire. After this step, the lexicon contains all the information required for the purposes of our joke generator and looks like Figure 28.

5.3 Generating the Jokes

Figure 29 shows the implementation of algorithm #1 in VINCI code and the lexical entries required to generate a particular joke.

```
"trunk_1"|N|Number| |#7|$13|"trunk" |homophone"trunk_2"|syn: "luggage compartment of car"|
"trunk_2"|N|Number| |$2|$13|"trunk"|homophone:"trunk_1"|syn: "elephant's nose"|
"raise"|V|Number| |$3|6|"tip"|homophone:"raze"|syn1: "bring"; syn2: "up"|
"raze"|V|Number| |$3 |6|"tip"|homophone: "raise"|syn1:"destroy"; syn2: ""|
"trunk_space"|CP|3|4|#1|$13|7|CPpt1:"b"/BLANK|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"
/BLANK|CPpt5:"trunk_1"/N|CPpt6:"space"/N|
"raise_cattle"|CP|3|4|#1|6|7|CPpt1:"raise"/V|CPpt2:"b"/BLANK|CPpt3:"b"/BLANK|CPpt4:"b"/BLA
NK|CPpt5: "b"/BLANK|CPpt6:"cattle"/N|
"space"|N|sing| |#7|$13|"space"| |
"cattle"|N|plur| |#7|$13|"cattle"| |
```

Figure 28: A sample of what the lexicon might look like after steps 5.2.1-5.2.4 have been performed on two homophones: "trunk" and "raise".

```
BASE=CP                    %

ROOT = MAKEPIVOT: BASE     %

MAKEPIVOT = TRANSFORMATION
CP:
ART/"The"                                   {The}
1/@10:CPpt3/@9:per_or_animal                {farmer}
1/@8:CPpt1/@12:why_x                        {is violent}
PUNCT/"^."                                  {.}
PRON/"He"                                   {He}
1/@8:CPpt1[sing]                            {raises}
1/@9:CPpt2                                  {      }
1/@10:CPpt3                                 {cattle}
PUNCT/"^." ;                                {.}
%
```

Figure 29: Implementation of Schema #1 in VINCI.

[5] None of the phrases acted upon by our algorithms have a preposition in them but the category exists nonetheless because synonyms of these phrases might include them.

Chapter 6

Analysis of Results

6.1 Evaluating the output

Of the 11 schemata, three (6b, 7 and 8) were implemented. The lexicon contained 36 common phrases and 240 words (nouns, verbs, determiners, prepositions and adjectives) and from it 50 jokes were output: 34 from schema #6b, 7 from schema #7 and 9 from schema #8. Questionnaires asking people to evaluate the puns were distributed to 16 volunteers (a sample questionnaire appears in appendix D)[1]. The jokes were graded on a scale from 1 to 5[2]:

1: Not a joke. Does not make sense.
2: Recognizably a joke but a pathetic one.
3: OK. A joke you might tell a child.
4: Quite good.
5: Really good.

Table 25 shows examples of jokes that received these various scores. The average point score for all the jokes was 2.81. In other words the jokes were, on average, better than pathetic but, according to the volunteers, not quite good enough to be enjoyed by a child. This statistic obscures the fact, however, that quite a number of good jokes were generated. For example, Figure 30 shows that nearly half of the jokes (22 out of 50) scored between 3-5. And Figure 31 reveals that a significant number of votes of 4 and 5 were given. In retrospect, jokes might have received higher scores had they been heard

[1] One of the questionnaires was rejected because the person did not follow the instructions.
[2] This scale was created by [BR94].

rather than read by the volunteers. Performing an oral evaluation of the jokes may be a better idea for future experiments involving these kinds of puns.

	Score	Jokes
a	1-2	*The butcher commits a carelessness. A gross negligence.* *Joan visits a grave in the basement. A bier cellar*
b	2-3	*The diver joins a coalition. A coral society.* *A store-keeper boards a ship. A sale boat.*
c	3-4	*The sailor earns a diploma. A berth certificate.* *The juvenile studies a writer. A minor poet.*
d	4-5	*The pheasant breathes oxygen. Fowl air.* *The sailor bears a stress. Pier pressure.*

Table 25: Examples of jokes with different scores.

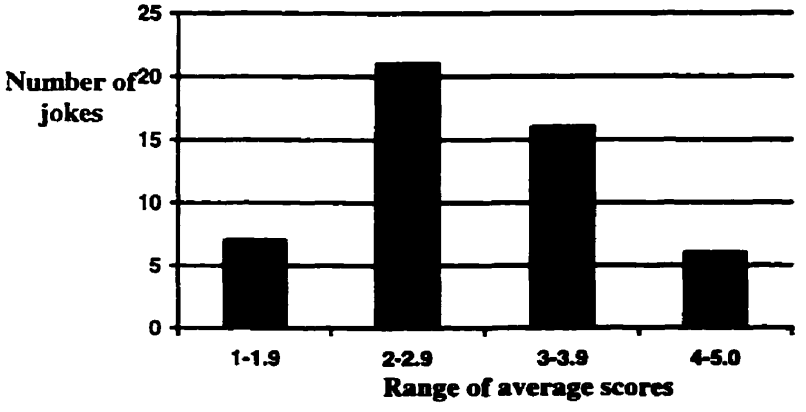


Figure 30: The average scores of the jokes. 7 jokes received a score from 0-1.9, 21 from 2-2.9, 16 from 3-3.9 and 6 from 4-5.

6.2 Improving the joke generator

The different components of the generator - the lexicon, schemata and templates - affect the quality of the jokes output. The results of the experiment suggest some simple ways the lexicon and schemata could be changed to improve the jokes. For instance the lexicon should not contain obscure words. Although all words deemed obscure were

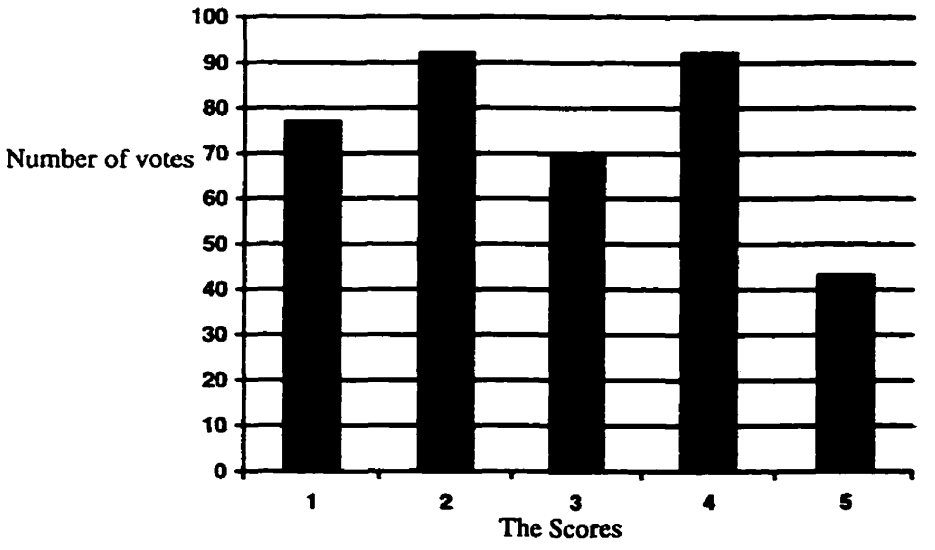


Figure 31: The number of votes per score.

removed from the lexicon during its creation (section 5.2.1), some words which were accepted proved to be unknown to many of the volunteers evaluating the puns. For example a number of the volunteers judging the jokes did not know the meaning of "buss", "bier", and "gross" (as in a

the schema	average score of a joke generated by the schema	average score of a joke generated by the schema if obscure words are not used by the generator
schema 6b	2.79	2.88
schema 7	2.94	unaffected
schema 8	2.80	3.09

Table 26: The average score of each schema and the average score if obscure words are filtered out.

qualifier of weight). If puns with these words were discounted, the average score for schema 6b rises from 2.79 to 2.88, the score for schema #7 is unaffected because it did not make use of any of these words, and schema #8's score rises significantly from 2.80 to 3.09 (schema #8's two worst rated puns are eliminated).

Another drawback with the lexicon was that some of the phrases in it were not really common phrases and they tended to produce inferiour puns. For example phrases such as "great hole", "main force", "lone parent", and "bare truth" are pairs of adjectives and nouns that can logically be grouped together, but they are not so intimately connected that they would be considered colloquial or idiomatic expressions. These phrases made it into the lexicon because when a homonym was looked up in the dictionary, the first phrase to occur in the entry for that homonym which fit our syntactic criteria was picked. We assumed that a phrase that occurred in an entry for a word would be idiomatic but this was not always the case.

It is important to use common phrases because a lot of the force of the HCPP joke depends on subverting the familiar: if the phrase is not really familiar, undermining it has little impact. Thus choosing phrases more carefully would improve the puns. A more specialized dictionary containing only phrases which are idiomatic would simplify the selection. Partridge's *Dictionary of Cliches*[Pa62], Henderson's *Dictionary of English Idioms*[He56] and *Wood's English Colloquial Idioms* [Wo69] were consulted but many of the expressions in them do not have the syntax required by our program. A program on the web called PhraseFinder [PF99] which takes a word as input and outputs phrases that make use of that word looks like a promising source for future research once more entries for it have been supplied. But it too missed many of the common phrases that a dictionary contains and so it was not used as a source. Therefore, if we were to conduct the experiment over again, the Oxford English dictionary would still act as the source for the phrases but an impartial human judge would examine the resulting list and be allowed to reject non idiomatic phrases. If this were done, significantly fewer bad puns would be generated.

Another way of improving the generator would be to have it reject phrases which are simply not good candidates for punning. The puns derived from these phrases are not funny, not so much because of a problem with individual schemata but because the general philosophy of using

homonymy as a logical mechanism for creating script opposition cannot always be relied upon. One is depending on the vagaries of chance - the accidental bringing together of two ideas and sometimes those ideas clash semantically. Therefore a filtering mechanism should be constructed which could discern semantic discord between words and reject them as candidates for puns. Implementing this filter would be straightforward in VINCI: words would be given attributes (such as "edible", "animate", "abstract" etc.) and the union of words with clashing attributes would be prohibited. Deciding on which attributes clash, however, is not so straightforward. For instance, Binsted argues that constructed phrases containing abstract nouns should be disallowed because they "do not evoke strong images the way more concrete words do" [BR94, 27]. It is true that some of these phrases seem impossibly hard to pun with: even a human being would have trouble finding a context in which one of our phrases "mane force" makes some kind of sense. But we do not agree that all phrases with abstract words should be rejected. Some of our best puns (see puns 8, 18, 39 and 47 in appendix E) contained a phrase with an abstract and concrete noun. (In fact the worst performing puns contained concrete compound noun phrases such as "grate hole", "base clef", "hole grain", "pail person" , "male bag" and "bass camp"). Thus a filtering method more sophisticated than the indiscriminate rejection of constructed phrases with abstract nouns seems warranted. Section 7.3 outlines a semantic network and attribute system which could be used to find points of similarity between words and if none were found, would allow us to more confidently assert that the words are indeed too dissimilar to be joined together in a pun.

Chapter 7

Conclusion

7.1 Summary

Attardo and Raskin's theory of humour was extended using concepts from Binsted. These insights helped us devise schemata for a particular kind of pun which we call the HCPP. A lexicon was then hand built with the following resources:

- a list of homonyms
- a dictionary
- volunteers

The schemata for generating the puns were implemented as grammars which were executed by VINCI, a natural language generator. Using these grammars and the lexicon, VINCI generated 50 puns (listed in appendix A) and volunteers were asked to judge their quality on a scale of 1 to 5. A significant number of them (22/50) received an average score from 3 to 5 and some slight changes mentioned in section 6.2 would improve the results further.

7.2 Possible extensions and improvements

Upon looking at the results, methods of adding variety to the kinds of puns output became apparent. For instance Figure 32 represents a schema based on schema 6b. The dotted line in Figure 32 marks the only difference to the schema and the template has been altered so that two sentences are output (rather than the sentence and sentence

fragment output by the original schema). In this new schema, the same connection exists between the pivot ("dire") and the target ("textile worker") but the pivot sentence's verb ("fulfills") and the second word in the common phrase ("need") relate to the adjective in the target sentence. These kinds of puns add variety to the generator because they have a different structure and rhythm and make use of different relations between words. Thus in addition to the kinds of puns output in our experiment, (27a) and (27b), (27c) and (27d) could be produced. Variety is important because people tire of hearing the same kind of pun with the same sentence structure. For this same reason, it would also be useful to implement the other schemata 1-5, 6a, 9, 10 described in section 4.7.

Another way of improving the output of the generator would be to find deeper and subtler connections between the pivot sentence and the target sentence. A method for doing this is discussed in the next section which outlines what the next generation of an HCPP joke-generator should look like.

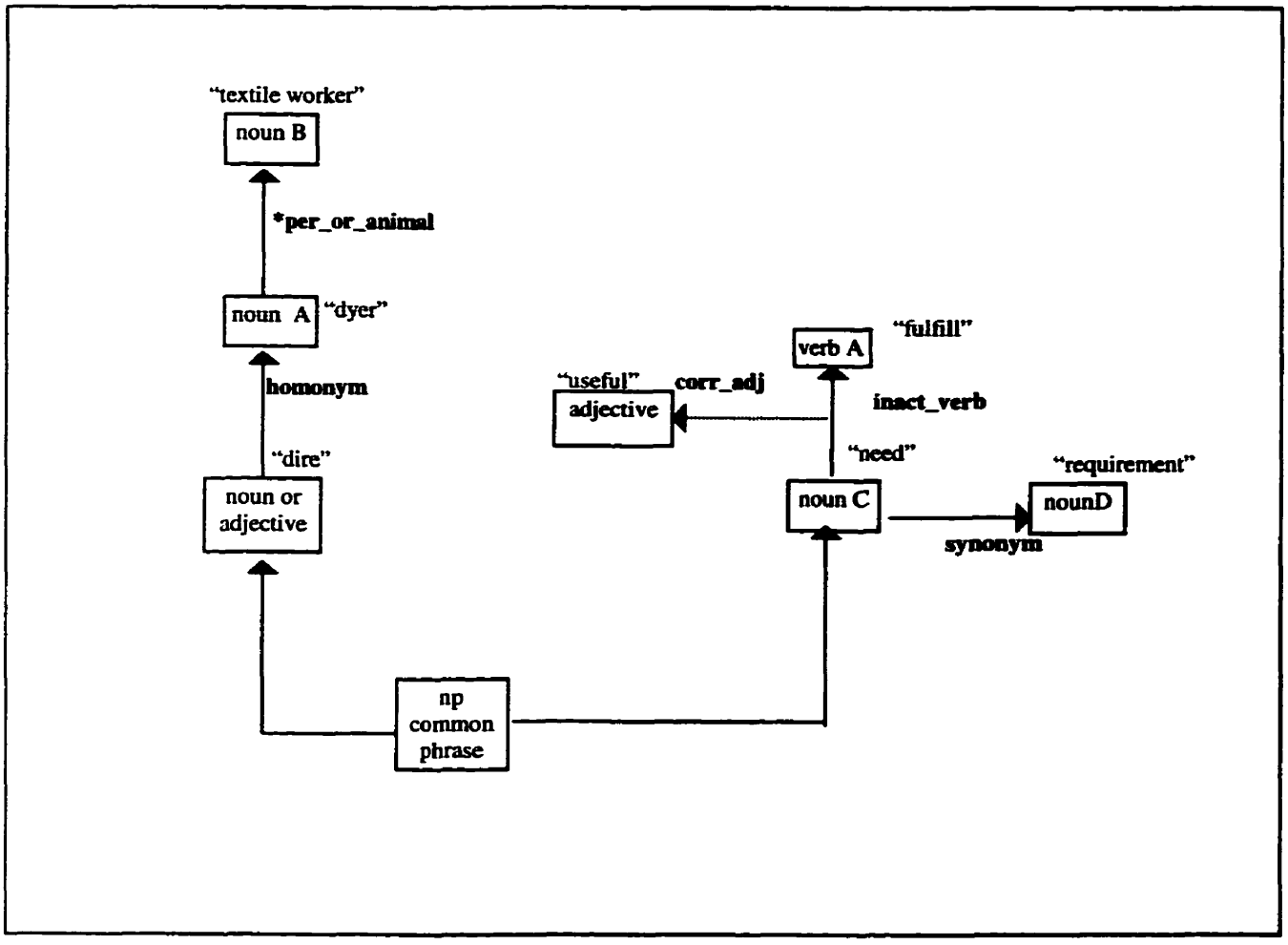
a	*The textile worker fulfills a requirement. A dyer need.*	A pun output by schema 6b.
b	*The grizzly speaks a verity. A bear truth.*	A pun output by schema 6b.
c	*The textile worker is useful. He fulfills a dyer need.*	A possible pun output by a new schema based on 6b.
d	*The grizzly is honest. He speaks the bear truth.*	A possible pun output by a new schema based on 6b.

Table 27: Puns output by 6b and by a variation of it.

7.3 The next generation

We envision that the next generation of the HCPP generator will find deeper, more varied and subtler connections between words in a more sophisticated semantic network. The schemata would investigate how words in the "fake" phrase (for example "dyer" and "need" in the constructed phrase "dyer need") might relate in their respective contexts.

1. noun(h)-noun
2. adj(h)-noun where hom(adj)->noun



Template for variation on Schema #6b

"The"<nounB> "is" <adjective> "."
"He" <verbA> {a/an} <nounA> <nounC>"."

Figure 32: A variation of schema 6b which aims to improve the rhythm of the resulting puns and to create a subtler context for the punchline. Possible puns output by this schema are: "The textile worker is useful. He fulfills a dyer need", "The grizzly is honest. He speaks the bear truth". "The knight is greedy. He acquires duel ownership", "The priest is responsible. He prevents an idol rumour".

The idea of primitive concepts plays a central role in the creation of this more sophisticated semantic network.

Primitive concepts

The notion of primitive concepts is important for artificial generation of humour and for computational linguistics in general. Most linguists agree that ideas are understood in terms of each other. A word's definition points to other words and these words' definitions in turn depend on the meaning of other words. In other words, words act like metaphors of each other. Some linguists argue, however, that all words cannot simply be pointers to other words because how then would any of them have meaning? They postulate that this regression does not proceed infinitely because there is a set of primitive concepts that are somehow "understood directly, without metaphor" (Lakoff, 56).

In *Semantic Structures* [Ja90], Ray Jackendoff draws an analogy between syntax and semantics to defend this idea that primitives form a foundation for human beings' understanding of words. He argues that just as an infinite number of sentences are formed from a finite set of words and rules for combining these words, so too an infinite number of concepts must be generated from a finite set of concepts and rules of combination. A person does not have an infinite list of every possible syntactic structure encoded in her brain yet she can with innate and learned rules she possesses, produce an infinite variety of sentence structures. Equally, Jackendoff argues that an infinite number of concepts exist which are not stored within the narrow compass of our finite brains but instead "must be generated on the basis of a finite set of principles of combination" [Ja90, 9].

A semantic network expressing ideas in terms of their primitive components would be very useful for natural language understanding and generation because deep connections between words would be captured - even between seemingly disparate words. For example, the kind of semantic network we visualize would possess links

between the following verbs: "run", "butter", "drive", "drink", "swim", "stop" and "pocket". On the surface, many of these actions seem quite unrelated but an idea uniting them all is the primitive concept of GO [Ja90]. It is obvious how the words "run", "drive" and "swim" express this idea but not so evident how the others do: "drink" contains the idea of a liquid GOing from a receptacle to a mouth, "pocket" means something GOing from somewhere into a pocket, and stop can be defined as not GO.

In regards to our pun generator, if ideas were expressed in terms of their primitive components, their commonalties and differences would be more clearly defined. And so finding the connection between two often quite different ideas brought together by the accident of homonymy would be facilitated.

a	*The textile worker fulfills a requirement. A dyer need.*	A pun output by the present generator.
b	*The factory hires more employees. There is a dyer need.*	A pun output by an improved generator.
c	*The factory fires employees. There is no dyer need.*	ditto above.
d	*Joan visits a grave in the basement. A bier cellar.*	A pun output by the present generator.
e	*The funeral home has a basement. It is a bier cellar.*	A pun output by an improved generator.

Table 28: Puns output by the present generator and an improved generator.

Take for example "dyer need", one of the fake phrases constructed by schema 6b in our experiment. The two ideas forced together here are "dyer" and "need" and a simple context merging them was created in pun (28a). A subtler and more sophisticated context could be found, however, with the proposed semantic network which, like our present lexicon, makes use of various relations and attributes but adds the feature of primitives. It is interesting to note that associating primitive concepts with a word could be accomplished using VINCI's attribute system. High-level attributes (such as

"employer" in Figure 33) could continue to exist but primitive concepts (such as "building" and "want" in Figure 33) would also be ascribed to words as attributes.

Figure 33 represents a sample of our proposed network and demonstrates one way the idea of "need" might manifest itself in the world of a "dyer". The network shows that the word "need" contains the primitive idea of "want" in it. The network also contains information that a dyer can be an "employee" who works for an "employer". Employees and employers have different relations to each other, one of which is that an employer "hires" employees. And one of the primitive ideas inherent in the idea of "hiring" is "wanting": an employer "hires" an employee because it "wants" that person or it "fires" an employee if it no longer "wants" that person. In this way a subtle and realistic relation between "need" and "dyer" would be found in the knowledge base and the resulting pun (28b or c) would be an improvement of (28a), the pun generated in our experiment. In the same way, 28(d) could be improved to 28(e) if a semantic network with primitive concepts were used (see Figure 34).

It would be difficult to determine how to carve up the world into categories of things and actions and decide which ideas are primitive[1]. And it would be an arduous task to decompose concepts into their primitive components but not impossible. In fact some thesauruses already have the kind of structure we envision for our semantic database: words are grouped together around a core idea (which could be thought of as a primitive idea). The only real difference between our network and this kind of thesaurus would be that the core ideas in our semantic

[1] For example at the beginning of his book Jackendoff chooses CAUSE to be a primitive but then argues later that this idea is not in fact primitive but should be decomposed further [Ja90, 131]. He states that a success parameter needs to be added in order to distinguish between cases when the application of some causative force is successful and when the result is undetermined. For example "Harry forced Sam to go away" (successful outcome) and "Harry pressured Sam to go away" (undetermined outcome). In this way a linguist's search for primitive concepts could be compared to the experience of generations of physical scientists who have explored the structure of the atom: their notions of what is indivisible are frequently challenged.

database would be at a more primitive level and so more words not normally associated with each other would be linked together via primitive concepts. For example the words "hire" and "need" would be connected in our network via the primitive concept "want" but are not associated with each other in a regular thesaurus. However some core ideas appearing in Roget's Thesaurus [Ch92] could, as they stand, be classified as primitive concepts. For example one of these ideas or categories is "impairment", a concept which pervades many different contexts. Some of the adjectives listed in the entry for this idea are:

1. injured, hurt, harmed
2. broken, chipped, shattered, in pieces, in shards
3. handicapped, maimed, limping
4. ragged, tattered, torn
5. depressed, frazzled, languishing, pining.

Within a general category, the thesaurus groups adjectives together into these kinds of semantically related clusters. Our network would contain these groups and attributes would be assigned to them to describe the kinds of nouns they can modify. For instance, lists 1,3, and 5 apply to animate objects (5 particularly to people), list 2 to breakable objects, and list 4 to things made out of material such as clothing. In this way deep and metaphorical connections between seemingly different ideas such as "broken" and " depressed" or "chipped" and "maimed" are captured and made available to our joke generator. The generator would find subtler connections between the pivot sentence/phrase and the target sentence, and better puns would be formed. For instance schema #9 (section 4.2) could be implemented and would yield jokes such as "John sees a depressed deer. It is a broken hart".

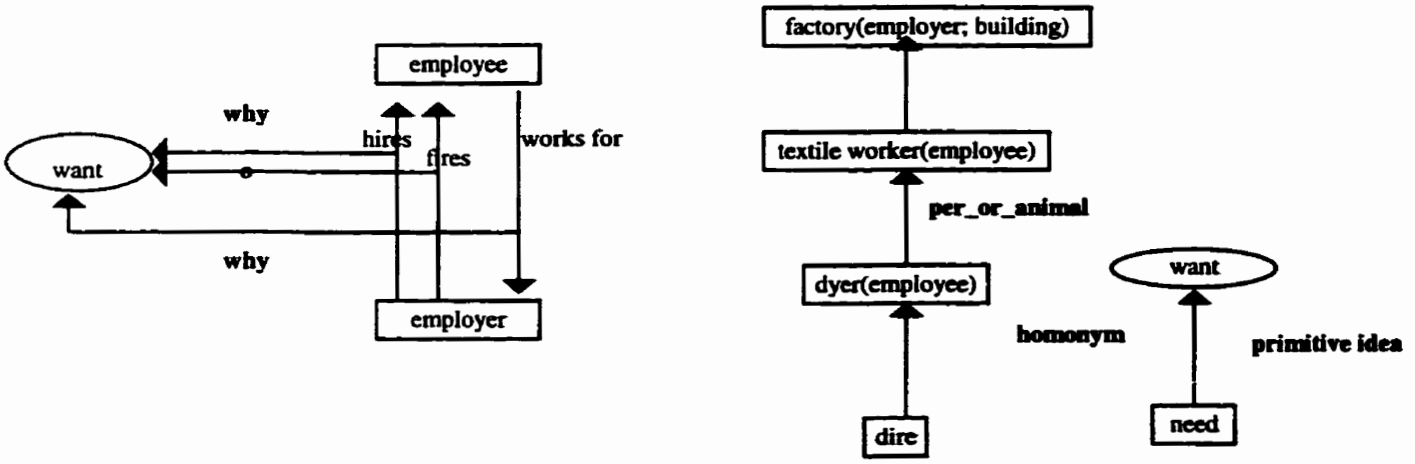


Figure 33: Part of the upgraded semantic network. The words in circles are primitive concepts. The resulting puns might be (25b) and (25c).

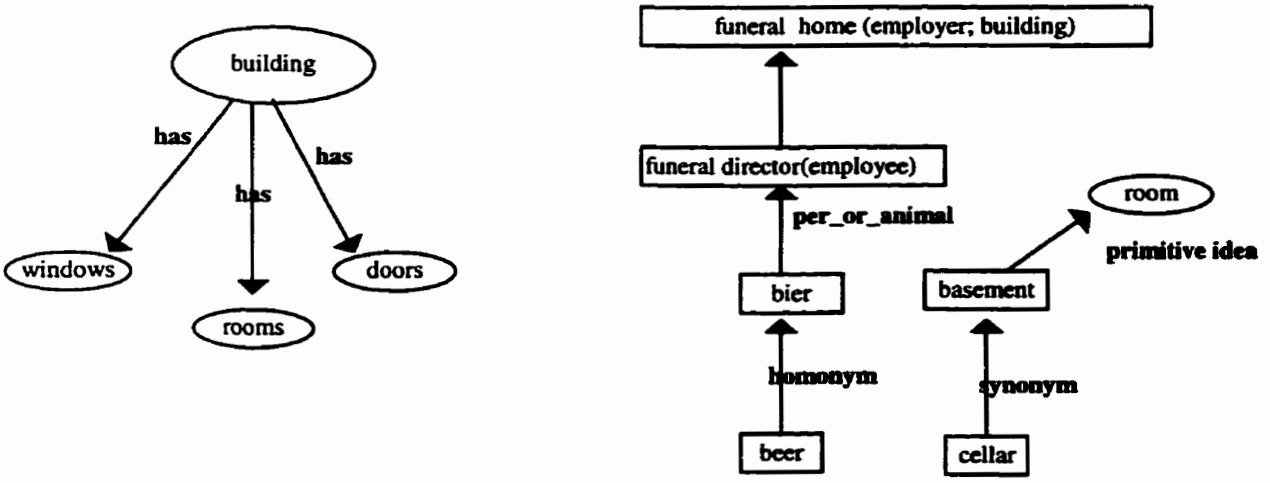


Figure 34: Part of the upgraded semantic network. A pun resulting from this might be 27(e).

7.4 Conclusion

The goals outlined in section 1 have been accomplished. Specifically we found some of the links that a lexical database will need to possess to generate a certain kind of pun and

we built a small database to hold these relations (goals 1 and 3). We created 10 schemata and templates for generating this type of pun and implemented three of them in VINCI, a natural language generator (goals 2 and 4). From these three implemented algorithms, 50 puns were computationally generated and then evaluated by a number of volunteers. Obvious ways of improving the output were suggested in chapter 6 and an extensive improvement was discussed in section 7.3.

We analyzed humour in detail because our goal was to discover formulas for creating puns and to implement them into a program which could generate jokes. We were successful in showing that recognizable jokes can be generated by a computer - that they obey a type of grammar - and so we punctured some of the mystique surrounding humour.

We discovered certain relations between words that are useful for joke generation (for example: per_or_animal, synonym, prep_assoc, homonym, act_verb etc.) and thus for natural language generation in general. In other words we have shown some of the semantic links that will have to appear in the enormously complex semantic network that we think needs to be constructed for truly automated generation and understanding of natural language.

Building HCPPs involved trying to combine two often very disparate ideas into a single sentence. Attempting to do this revealed the complexity of creating coherent sentences. We found that attributes are an effective way of enforcing semantic coherence and we introduced the idea of primitive ideas which can be regarded as special kinds of attributes. A semantic network containing primitive concepts and words' relations to them would act as a powerful resource for generating more sophisticated jokes and for natural language generation in general.

Bibliography

[All95] James Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., California, 1995.

[AR91] Salvatore Attardo and Victor Raskin. Script theory revis(it)ed: joke similarity and joke representation model. *Humor*, 4(3):293-347, 1991.

[Bin96] Kim Binsted. *Machine humour: An implemented model of puns*. Dissertation, University of Edinburgh, 1996.

[BR94] Kim Binsted and Graeme Ritchie. A symbolic description of punning riddles and its computer implementation. Research Paper 688, University of Edinburgh, Edinburgh, Scotland, 1994.

[BR96] Kim Binsted and Graeme Ritchie. Speculations on story puns. *Proceedings of the International Workshop on Computational Humour*, Enschede, Netherlands.

[Ch92] Ed. Robert L. Chapman. *Roget's International Thesaurus*. Harper Perennial, New York, 1992.

[Chi92] Delia Chiaro. *The language of jokes: analysing verbal play*. Routledge, London, 1992.

[Cu88] Ed. Jonathan Culler. *On Puns*. Basil Blackwell Ltd., Oxford, 1988.

[Co99] Alan Cooper. Alan Cooper's Homonyms. *As of August 1999, available at http://www.cooper.com/alan/homonym.html*

[Fr66] Ed. Julian H. Franklyn. *Which Witch? Being a grouping of phonetically compatible words*. Hamish Hamilton, London, 1966.

[Gio91] R. Giora. On the cognitive aspects of the joke. *Journal of Pragmatics*, 16:465-485.

[He56] B. Henderson. *Dictionary of English Idioms*. James Blackwood&Co., London, 1962.

[Ho77] C.F. Hockett. *The View from Language*. University of Georgia Press, Athens, 1977.

[Ja90] Ray Jackendoff. *Semantic Structures*. MIT Press, Cambridge Mass., 1990.

[La80] George Lakoff. *Metaphors We Live By*. University of Chicago, Chicago, 1980.

[Len90] Douglas B Lenat. *Building Large Knowledge-Based Systems*.Addison-Wesley Publishing Company, Inc., Reading, 1990.

[LL97] G. Lessard and M. Levison. Rule-governed Wordplay and Creativity. Computational

models of Creative Cognition Conference. Dublin, 1997.

[LL96] M. Levison and G. Lessard. VINCI Project: Natural Language Generation Environment Documentation. Queen's University, 1996.

[LL95] G. Lessard and M. Levison. Linguistic and Cognitive Underpinnings of Verbal Humour. International Cognitive Linguistics Association conference, Albuquerque, NM, 1995

[LL93] G. Lessard and M. Levison. Computational Models of Riddling Strategies. ACH/ALLC93 Conference Abstracts, pp. 120-122, Georgetown, 1993.

[LL92a] M. Levison and G. Lessard. A System for Natural Language Generation. *Computers and the Humanities*, 26:43-58, 1992.

[LL92b] G. Lessard and M. Levison. Computational Modelling of Linguistic Humour: Tom Swifties, ALLC/ACH92 Conference Abstracts, pp. 175-178, Oxford, 1992.

[Min63] M. Minsky. Steps Towards Artificial Intelligence. E. Feigenbaum and J Feldman, editors, *Computers and Thought*, pages 406-450. McGraw-Hill, 1963.

[OD92] William O'Grady and Michael Dobrovsky. *Contemporary Linguistic Analysis: An Introduction*. Copp Clark Pitman Ltd., Toronto, 1992.

[Pa62] Eric Partridge. *Dictionary of Cliches*. Routledge&Kegan Paul Ltd., London, 1962.

[PF99] Phrase Finder *As of August 1999, available at http://www.shu.ac.uk/web-admin/phrases/list.html.*

[PG84] W.J. Pepicello and Thomas A. Green. *The Language of Riddles*. Ohio State University, 1984.

[RAR93] Wilhelm Ruch, Salvatore Attardo and Victor Raskin. Toward an empirical verification of the General Theory of Verbal Humour. *Humor*, 6(2):123-136, 1993.

[Ras85] Victor Raskin. *The Semantic Mechanisms of Humour*. Dordrecht Reidel, 1985.

[Ri83] Elaine Rich. *Artificial Intelligence*. McGraw-Hill, 1983.

[Ur80] Lawrence Urdang. *Picturesque Expressions*. Gale Research Company, Detroit Mich., 1980.

[Th95] Ed. Della Thompson. *The Concise Oxford Dictionary of Current English*. Clarendon Press, Oxford, 1995.

[Wo69] J. Wood. *English Colloquial Idioms*. MacMillan and Co. Ltd., 1969.

Appendix A - The lexicon, morphology rules, attributes and terminals.

The lexicon

```
"John"|PROP|sing||#1||||"He"/PRON|
"He"|PRON|sing||#1|||
"Joan"|PROP|sing||#1||||"She"/PRON|
"She"|PRON|sing||#1|||
"The"|ART|||#1|
"the"|ART|||#1|
"a"|ART||||$13|"a"|
"A"|ART||||$13|"A"|
"^."|PUNCT|||#1|
"on"|PREP|||#1|
"in"|PREP|||#1|
"at"|PREP|||#1|
"It"|N|||#1|
"is"|V|||#1|
"to"|PREP|||#1|

"barren land"|CP|||#1||||||CPpt4:"barren"/ADJ||CPpt6:"land"/N|
"bare truth"|CP|||#1||||||CPpt4:"bare"/ADJ||CPpt6:"truth"/N|
"serial
killer"|CP|||#1||||||CPpt4:"serial"/ADJ||CPpt6:"killer"/N|
"choral
society"|CP|||#1||||||CPpt4:"choral"/ADJ||CPpt6:"society"/N|
"dire need"|CP|||#1||||||CPpt4:"dire"/ADJ||CPpt6:"need"/N|
"dual
ownership"|CP|||#1||||||CPpt4:"dual"/ADJ||CPpt6:"ownership"/N|
"foul air"|CP|||#1||||||CPpt4:"foul"/ADJ||CPpt6:"air"/N|
"gross
negligence"|CP|||#1||||||CPpt4:"gross1"/ADJ||CPpt6:"negligence"/N
|
"great hole"|CP|||#1||||||CPpt4:"great"/ADJ||CPpt6:"hole"/N|
"hoarse voice"|CP|||#1||||||CPpt4:"hoarse"/ADJ||CPpt6:"voice"/N|
"whole grain"|CP|||#1||||||CPpt4:"whole"/ADJ||CPpt6:"grain"/N|
"hostile
enemy"|CP|||#1||||||CPpt4:"hostile"/ADJ||CPpt6:"enemy"/N|
"idle rumour"|CP|||#1||||||CPpt4:"idle"/ADJ||CPpt6:"rumour"/N|
"lone parent"|CP|||#1||||||CPpt4:"lone"/ADJ||CPpt6:"parent"/N|
"main force"|CP|||#1||||||CPpt4:"main"/ADJ||CPpt6:"force"/N|
"major
surgery"|CP|||#1||||||CPpt4:"major1"/ADJ||CPpt6:"surgery"/N|
"minor poet"|CP|||#1||||||CPpt4:"minor1"/ADJ||CPpt6:"poet"/N|
"naval ship"|CP|||#1||||||CPpt4:"naval"/ADJ||CPpt6:"ship"/N|
"pale person"|CP|||#1||||||CPpt4:"pale"/ADJ||CPpt6:"person"/N|
```

```
"bass clef"|CP|||#1|||||||CPpt5:"bass"/N|CPpt6:"clef"/N|
"beech tree"|CP|||#1|||||||CPpt5:"beech"/N |CPpt6:"tree"/N|
"birth certificate"|CP|||#1|||||||CPpt5:"birth"/N
|CPpt6:"certificate"/N|
"carol singing"|CP|||#1|||||||CPpt5:"carol"/N |CPpt6:"singing"/N|
"fairy godmother"|CP|||#1|||||||CPpt5:"fairy"/N
|CPpt6:"godmother"/N|
"peer pressure"|CP|||#1|||||||CPpt5:"peer"/N |CPpt6:"pressure"/N|
"air bag"|CP|||#1|||||||CPpt5:"air"/N |CPpt6:"bag"/N|
"ant hill"|CP|||#1|||||||CPpt5:"ant"/N |CPpt6:"hill"/N|
"base camp"|CP|||#1|||||||| CPpt5:"base"/N |CPpt6:"camp"/N|
"nightclub"|CP|||#1|||||||CPpt5:"night"/N |CPpt6:"club"/N|
"mail bag"|CP|||#1|||||||CPpt5:"mail"/N |CPpt6:"bag"/N|
"beer cellar"|CP|||#1|||||||CPpt5:"beer"/N |CPpt6:"cellar"/N|
"dockyard"|CP|||#1|||||||CPpt5:"dock"/N |CPpt6:"yard"/N|
"sailboat"|CP|||#1|||||||CPpt5:"sail"/N|CPpt6:"boat"/N|
"bus shelter"|CP|||#1|||||||CPpt5:"bus"/N |CPpt6:"shelter"/N|


"acquire"|V|Number||$3||"acquire"||syn:"take"/V|
"air"|N|plur, have||#7||"air"|homonym:
"heir"/N|syn:"oxygen"/N||inact_verb:"breathe"/V|
"ant"|N|Number||$2|||homonym: "aunt"/N|
"arrive at"|V|Number||$2||"arrive
at"||syn:"reach"/V|"arrive"|"at"|
"aunt"|N|Number||$2||"aunt"|homonym:
"ant"/N|syn:"relation"/N|per_or_animal:
"relation"/N|inact_verb:"visit"/V|
"bag"|N|Number,
have,location||$2||"bag"||syn:"sack"/N||inact_verb:
"carry"/V|prep_assoc:"in"/PREP|
"banker"|N|Number||$2||"banker"|
"bare"|ADJ|||#1|||homonym: "bear"/N|
"barren"|ADJ|||#1|||homonym: "baron"/N|
"baron"|N|Number||$2||"baron"|homonym:
"barren"/ADJ|syn:"lord"/N|per_or_animal: "lord"/N|
"baseball player"|N|Number||$10|||||"baseball"|"player"|
"base"|N|Number, location||$2||"base"|homonym:
"bass"/N|syn:"station"/N|per_or_animal: "baseball
player"/N||prep_assoc: "on"/PREP|
"basement"|N|Number, location||$2||"basement"|
"bass"|N|Number||$5||"bass"|homonym:
"base"/N|syn:"vocalist"/N|per_or_animal:"vocalist"/N||inact_verb:
"hear"/V|
"bazaar"|N|Number||$2||"bazaar"|
"beech"|N|Number||$4|||homonym: "beach"/N|
"beach"|N|Number, location||$4||"beach"|homonym:
"beech"/N|syn:"sandbank"/N|per_or_animal:
"lifeguard"/N||prep_assoc: "on"/PREP|
"bear"|N|Number||$2||"bear"|homonym:
"bare"|syn:"grizzly"|per_or_animal: "grizzly"|
"bear"|V|Number||$3||"bear"|
"beer"|N|Number||$2|||homonym: "bier"/N|
```

```
"bier"|N|Number, location||$2||"bier"|homonym:
"beer"/N|syn:"grave"/N|per_or_animal: "grave-
digger"/N|inact_verb:"visit"/V|prep_assoc:"in"/PREP|
"birth"|N|Number||$2|||homonym: "berth"/N|
"berth"|N|Number,
location||$4||"berth"|homonym:"birth"/N|syn:"bunk"/N|per_or_anima
l:"sailor"/N||prep_assoc:"in"/PREP|
"board"|V|Number||$3||"board"||syn:"get in"/V|
"boardwalk"|N|Number, location||$2||"boardwalk"|
"boat"|N|Number, have,
location||$2||"boat"||syn:"ship"/N||inact_verb:
"board"/V|prep_assoc:"on"/PREP|
"booth"|N|Number, location||$2||"booth"|
"breathe"|V|Number||$3||"breathe"||syn:"inhale"/V|
"bunk"|N|Number, location||$2||"bunk"|
"burrow"|N|Number||$2||"burrow"|
"bus"|N|Number||$4|||homonym: "buss"/N|
"buss"|N|Number||$4||"buss"|homonym:
"bus"/N|syn:"peck"/N|per_or_animal:
"lover"/N|inact_verb:"give"/V|
"butcher"|N|Number||$2||"butcher"|
"camp"|N|Number, have,
location||$2||"camp"||syn:"settlement"/N||inact_verb: "arrive
at"/V|prep_assoc:"at"/PREP|
"capture"|V|Number||$3||"capture"||syn:"catch"/V|
"carol"|N|Number||$2|||homonym: "carrel"/N|
"carrel"|N|Number, location||$4||"carrel"|homonym:
"carol"/N|syn:"booth"/N|per_or_animal: "student"/N||prep_assoc:
"in"/PREP|
"carelessness"|N|Number||$2||"carelessness"|
"carry"|V|Number||$6||"carry"||syn:"hold"/V|
"cellar"|N|Number,
have,location||$2||"cellar"||syn:"basement"/N||inact_verb:
"enter"/V|prep_assoc:"in"/PREP|
"certificate"|N|Number,
have||$2||"certificate"||syn:"diploma"/N||inact_verb: "earn"/V|
"coral"|N|Number||$2||"coral"|homonym:
"choral"/ADJ|syn:|per_or_animal: "diver"/N|
"cereal"|N|Number||$2||"cereal"|homonym:
"serial"/ADJ|syn:|per_or_animal: "housewife"/N|
"charm"|V|Number||$3||"charm"|
"chimney sweep"|N|Number||$10|||||"chimney"|"sweep"|
"choral"|ADJ|||#1|||homonym: "coral"/N|
"cleaner"|N|Number||$2||"cleaner"|
"clef"|N|Number||$2||"clef"||syn:"musical sign"/N||inact_verb:
"look at"/V|
"climb"|V|Number||$3||"climb"||syn:"scale"/V|
"cultivate"|V|Number||$3||"cultivate"||syn:"develop"/V|
"coalition"|N|Number||$2||"coalition"|
"commit"|V|Number||$3||"commit"||syn:"perform"/V|
"decrease"|V|Number||$3||"decrease"||syn:"alleviate"/V|
"dock"|N|Number||$2|||homonym: "doc"/N|
```

```
"doc"|N|Number||$2||"doc"|homonym:
"dock"/N|syn:"surgeon"/N|per_or_animal:"surgeon"/N|inact_verb:"vi
sit"/V|
"dig"|V|Number||$3||"dig"||syn:"excavate"/V|
"diploma"|N|Number||$2||"diploma"|
"dire"|ADJ|||#1|||homonym: "dyer"/N|
"disco"|N|Number, location||$2||"disco"|
"diver"|N|Number||$2||"diver"|
"dual"|ADJ|||#1|||homonym: "duel"/N|
"duel"|N|Number||$2||"duel"|homonym:
"dual"/ADJ|syn:""|per_or_animal: "knight"/N|
"dyer"|N|Number||$2||"dyer"|homonym: "dire"/ADJ|syn:"textile
worker"/N|per_or_animal:"textile worker"/N|
"earn"|V|Number||$3||"earn"||syn:"acquire"/V|
"enemy"|N|Number, have||$7||"enemy"||syn:"foe"/N||inact_verb:
"hate"/V|
"energy"|N|plur||$7||"energy"|
"enter"|V|Number||$2||"enter"||syn:"go in"/V|
"erect"|V|Number||$2||"erect"||syn:"build"/V|
"exotic dancer"|N|Number||$10|||||"exotic"|"dancer"|
"father"|N|Number||$2||"father"|
"fairy"|N|Number||$4|||homonym: "ferry"|
"ferry"|N|Number,location||$4||"ferry"|homonym:"fairy"/N|syn:"boa
t"/N|per_or_animal:"sailor"/N||prep_assoc: "on"/PREP|
"first born"|N|Number||$10|||||"first"|"born"|
"foe"|N|Number||$2||"foe"|
"force"|N|Number||$2||"force"||syn:"energy"/N||inact_verb:
"harness"/V|
"foul"|ADJ|||#1|||homonym: "fowl"/N|
"fowl"|N|Number||$2||"fowl"|homonym:
"foul"/ADJ|syn:"pheasant"/N|per_or_animal: "pheasant"/N|
"fulfil"|V|Number||$3||"fulfil"||syn:"satisfy"/V|
"garden"|N|Number, location||$2||"garden"|
"general"|N|Number||$2||"general"|
"give"|V|Number||$3||"give"|
"go to"|V|Number||#1||||syn:"visit"|"go"|"to"|
"godmother"|N|Number,
have||$2||"godmother"||syn:"relation"/N||inact_verb: "visit"/V|
"gossip"|N|plur||#7||"gossip"|
"grain"|N|Number, have||$2||"grain"||syn:"seed"/N||inact_verb:
"plant"/V|
"grate"|N|Number||$2||"grate"|homonym:
"great"/ADJ|syn:""|per_or_animal: "chimney sweep"/N|
"grave"|N|Number, location||$2||"grave"|
"grave-digger"|N|Number||$10|||||"grave"|"digger"|
"great"|ADJ|||#1|||homonym: "grate"/N|
"grizzly"|N|Number||$7||"grizzly"|
"gross1"|ADJ|||#7||"gross"|homonym: "gross2"/N|
"gross2"|N|Number||$7||"gross"|homonym:
"gross1"/ADJ|syn:""|per_or_animal: "butcher"/N|
"harness"|V|Number||$4||"harness"||syn:"use"/V|
"hate"|V|Number||$3||"hate"||syn:"despise"/V|
"have"|V|Number||$9||"have"|
"hear"|V|Number||$3||"hear"||syn:"listen to"/V|
```

```
"heir"|N|Number||$4||"heir"|homonym: "air"/N|syn:"first
born"/N|per_or_animal: "first born"/N|inact_verb:"marry"/V|
"hero"|N|Number||$5||"hero"|
"hill"|N|Number,
have,location||$2||"hill"||syn:"mound"/N||inact_verb:
"climb"/V|prep_assoc:"on"/PREP|
"hoarse"|ADJ|||#1|||homonym: "horse"/N|
"hole"|N|Number, location||$2||"hole"||syn:"burrow"/N|
per_or_animal: "miner"/N|inact_verb: "dig"/V|
"home"|N|Number, location||$2||"home"|
"horse"|N|Number||$2||"horse"|homonym:
"hoarse"/ADJ|syn:""|per_or_animal: "pony"/N|
"hostile"|ADJ|||#1|||homonym: "hostel"/N|
"hostel"|N|Number||$2||"hostel"|homonym:
"hostile"/ADJ|syn:""|per_or_animal: "social worker"/N|
"housewife"|N|Number||$2||"housewife"|
"idle"|ADJ|||#1|||homonym: "idol"/N|
"idol"|N|Number||$2||"idol"|homonym:
"idle"/ADJ|syn:""|per_or_animal: "priest"/N|
"individual"|N|Number||$2||"individual"|
"join"|V|Number||$3||"join"||syn: "enter"/V|
"juvenile"|N|Number||$2||"juvenile"|
"killer"|N|Number||$2||"killer"||syn:"murderer"/N||inact_verb:
"capture"/V|
"kiss"|V|Number||$4||"kiss"|
"knight"|N|Number||$2||"knight"|homonym:"night"/N|syn:"hero"/N|pe
r_or_animal:"hero"/N|inact_verb:"kiss"/V|
"land"|N|Number, location,
have||$2||"land"||syn:"region"/N||inact_verb: "cultivate"/V|
"lifeguard"|N|Number||$2||"lifeguard"|
"lion"|N|Number||$2||"lion"|
"listen to"|V|Number||$8||"listen
to"||syn:"obey"/V|"listen"|"to"|
"lone"|ADJ|||#1|||homonym: "loan"/N|
"loan"|N|Number||$2||"loan"|homonym:
"lone"/ADJ|syn:""|per_or_animal: "banker"/N|
"look at"|V|Number||$8||"look at"||syn:"read"/V|"look"|"at"|
"lord"|N|Number||$2||"lord"|
"love"|V|Number||$3||"love"||syn:"cherish"/V|
"lover"|N|Number||$2||"lover"|
"main"|ADJ|||#1|||homonym: "mane"/N|
"major1"|ADJ|||$2||"major"|homonym: "major2"/N|
"major2"|N|Number||$2||"major"|homonym:
"major1"/ADJ|syn:"general"/N|per_or_animal: "general"/N|
"mail"|N|plur||#1|||homonym: "male"/N|
"male"|N|Number||$2||"male"|homonym:
"mail"/N|syn:"man"/N|per_or_animal: "man"/N|inact_verb:"charm"/V|
"man"|N|Number||$2||"man"|
"mane"|N|Number||$2||"mane"|homonym:
"main"/ADJ|syn:""|per_or_animal: "lion"/N|
"marry"|V|Number||$6||"marry"|
"miner"|N|Number||$2||"miner"|
"minor1"|ADJ|||#7||"minor"|homonym: "minor2"/N|
```

```
"minor2"|N|Number||$2||"minor"|homonym:
"minor1"/ADJ|syn:"juvenile"/N|per_or_animal: "juvenile"/N|
"mound"|N|Number, location||$2||"mound"|
"murderer"|N|Number||$2||"murderer"|
"musical sign"|N|Number||$10|||||"musical"|"sign"|
"naval"|ADJ|||#1|||homonym: "navel"/N|
"navel"|N|Number||$2||"navel"|homonym:
"barren"/ADJ|syn:""|per_or_animal: "exotic dancer"/N|
"need"|N|Number,
have||$3||"need"||syn:"requirement"/N||inact_verb: "fulfil"/V|
"negligence"|N|Number||$2||"negligence"||syn:"carelessness"/N||in
act_verb: "commit"/V|
"night"|N|Number||$2|||homonym: "knight"/N|
"operation"|N|Number||$2||"operation"|
"oxygen"|N|plur||$2||"oxygen"|
"ownership"|N|Number,
have||$2||"ownership"||syn:"possession"/N||inact_verb:
"acquire"/V|
"pale"|ADJ|||#1|||homonym: "pail"/N|
"pail"|N|Number||$2||"pail"|homonym:
"pale"/ADJ|syn:""|per_or_animal: "cleaner"/N|
"parent"|N|Number,
have||$2||"parent"||syn:"father"/N||inact_verb: "listen to"/V|
"peck"|N|Number||$2||"peck"|
"peer"|N|Number||$2|||homonym: "pier"/N|
"pier"|N|Number, location||$4||"pier"|homonym:
"peer"/N|syn:"boardwalk"/N|per_or_animal: "sailor"/N||prep_assoc:
"on"/PREP|
"perform"|V|Number||$3||"perform"||syn:"do"/V|
"person"|N|Number||$2||"person"||syn:"individual"/N||inact_verb:
"love"/V|
"pheasant"|N|Number||$2||"pheasant"|
"plant"|V|Number||$3||"plant"||syn:"cultivate"/V|
"poet"|N|Number||$2||"poet"||syn:"writer"/N||inact_verb:
"study"/V|
"pony"|N|Number||$7||"pony"|
"possession"|N|Number||$2||"possession"|
"pressure"|N|Number||$2||"pressure"||syn:"stress"/N||inact_verb:
"bear"/V|
"prevent"|V|Number||$3||"prevent"||syn:"stop"/V|
"priest"|N|Number||$2||"priest"|
"region"|N|Number||$2||"region"|
"relation"|N|Number||$2||"relation"|
"requirement"|N|Number||$2||"requirement"|
"rumour"|N|Number||$2||"rumour"||syn:"gossip"/N||inact_verb:
"prevent"/V|
"sack"|N|Number, location||$2||"sack"|
"sail"|N|Number||$2|||homonym: "sale"/N|
"sailor"|N|Number||$2||"sailor"|
"sale"|N|Number||$2||"sale"|homonym:
"sail"/N|syn:"bazaar"/N|per_or_animal: "store-
keeper"/N|inact_verb:"have"/V|
"sandbank"|N|Number, location||$2||"sandbank"|
"seed"|N|Number||$2||"seed"|
```

```
"serial"|ADJ|||#1|||homonym: "cereal"/N|
"settlement"|N|Number, location||$2||"settlement"|
"shelter"|N|Number, have,
location||$2||"shelter"||syn:"home"/N||inact_verb:
"erect"/V|prep_assoc:"in"/PREP|
"ship"|N|Number, location,
have||$2||"ship"||syn:"boat"/N||inact_verb: "board"/V|prep_assoc:
"on"/PREP|
"shrub"|N|Number||$2||"shrub"|
"singing"|N|plur||$2||"singing"||syn:"wailing"/N||inact_verb:
"hear"/V|
"social worker"|N|Number||$10|||||"social"|"worker"|
"society"|N|Number,
have||$7||"society"||syn:"coalition"/N||inact_verb:"join"/V|
"speak"|V|Number||$3||"speak"||syn:"articulate"/V|
"station"|N|Number, location||$2||"station"|
"store-keeper"|N|Number||$10|||||"store"|"keeper"|
"stress"|N|plur||$5||"stress"|
"student"|N|Number||$2||"student"|
"study"|V|Number||$6||"study"||syn:"analyze"/V|
"surgeon"|N|Number||$2||"surgeon"|
"surgery"|N|Number,
have||$7||"surgery"||syn:"operation"/N||inact_verb: "perform"/V|
"tree"|N|Number, location,
have||$2||"tree"||syn:"shrub"/N||inact_verb: "plant"/V|
"truth"|N|Number, have||$2||"truth"||syn:"verity"/N||inact_verb:
"speak"/V|
"textile worker"|N|Number||$10|||||"textile"|"worker"|
"use"|V|Number||$3||"use"||syn:"practise"/V|
"verity"|N|Number||$7||"verity"|
"visit"|V|Number||$3||"visit"||syn:"meet with"/V|
"vocal chord"|N|Number||$10|||||"vocal"|"chord"|
"vocalist"|N|Number||$2||"vocalist"|
"voice"|N|Number, have||$2||"voice"||syn:"vocal
chord"/N||inact_verb: "use"/V|
"wailing"|N|plur||$2||"wailing"|
"whole"|ADJ|||#1|||homonym: "hole"/N|
"writer"|N|Number||$2||"writer"|
"yard"|N|Number, have,
location||$2||"yard"||syn:"garden"/N||inact_verb:
"dig"/V|prep_assoc:"in"/PREP|
```

Morphology Rules

```
{Some morphology rules for nouns and verbs}

rule 2
{Nouns like "book" whose plural forms require an "s"}
sing : #7;
plur : #7 + "s";
%

rule 3
{Verbs like "run", 3rd person present}
sing : #7 + "s";
plur : #7;
%

rule 4
{Verbs like "trespass" become "trespasses"}
sing : #7 + "es";
plur : #7;
%

rule 5
{Nouns like "bus" become "buses"}
sing : #7;
plur : #7 + "es";
%

rule 6
{Verbs like "study" become "studies"}
sing : #7-+"ies";
plur : #7;
%

rule 7
{Nouns like "surgery" become "surgeries"}
```

```
sing : #7;
plur : #7-+"ies";
%


rule 8
{For two word verb phrase like "listen to" become "listens to"
when 3rd person sing}
sing : #10 + "s" + #11;
plur : #10 + #11;
%


rule 9
{For verbs like "have" become "has" for 3rd person sing}
sing : #7--+"s";
plur : #7;
%

rule 10
{For compound nouns like "charley horse" become "charley horses"}
sing : #1;
plur : #10 + #11 + "s";
%

rule 13
{this rule decides if the indefinite article "a" becomes "an"}
1="a*"|1="e*"|1="i*"|1="o*"|1="u*" : #7 + "n";
* : #7;
%
```

Attributes

```
Number (sing, plur)
Semantics {location, have}
%
```

Terminals

```
CP
N
V
ADJ
ART
PUNCT
PROP
PRON
PREP
```

Appendix B - The implemented schemata

{This is algorithm 6b which deals with adj-noun or noun-noun cps
-the adjective or the first noun is the homonym
}

BASE = CP %

ROOT = MAKEPIVOT: BASE %

MAKEPIVOT = TRANSFORMATION

CP:

ART/"The"	{The}
1/@11:CPpt4/@8:homonym/@10:per_or_animal[sing]	{diver}
1/@13:CPpt6/@11:inact_verb[sing]	{joins}
ART/"a"	{a}
1/@13:CPpt6/@9:syn	{coalition}
PUNCT/"^."	{.}
ART/"A"	{A}
1/@11:CPpt4/@8:homonym	{coral}
1/@13:CPpt6	{society}
PUNCT/"^."	{.}

%

{This is algorithm 7 which deals with adj-noun or noun-noun cps
-the adjective or the first noun has a noun homonym which is a
location.}

BASE = CP %

ROOT = MAKEPIVOT: BASE %

MAKEPIVOT = TRANSFORMATION

CP:

PROP/"John"	{John}
1/@13:CPpt6/@11:inact_verb[sing]	{plants}
ART/"a"	{a}
1/@13:CPpt6/@9:syn	{shrub}
1/@12:CPpt5/@8:homonym/@12:prep_assoc	{on}
ART/"the"	{the}
1/@12:CPpt5/@8:homonym/@9:syn[location]	{shore}
PUNCT/"^."	{.}
ART/"A"	{A}
1/@12:CPpt5/@8:homonym	{beach}
1/@13:CPpt6	{tree}

```
PUNCT/"^."                                    {.}
%
```

{This is algorithm 8 which deals with adj-noun or noun-noun cps
-the adjective or the first noun has a noun homonym. The last
word in the phrase is a location. }

```
BASE = CP  %
```

```
ROOT =  MAKEPIVOT: BASE %
```

```
MAKEPIVOT = TRANSFORMATION
```

```
CP:
PROP/"Joan"                                   {John}
1/@12:CPpt5/@8:homonym/@11:inact_verb[sing]   {has}
ART/"a"                                       {a}
1/@12:CPpt5/@8:homonym/@9:syn                 {bazaar}
1/@13:CPpt6/@12:prep_assoc                    {on}
ART/"the"                                     {the}
1/@13:CPpt6/@9:syn[location]                  {ship}
PUNCT/"^."                                    {.}
ART/"A"                                       {A}
1/@12:CPpt5/@8:homonym                        {sale}
1/@13:CPpt6                                   {boat}
PUNCT/"^."                                    {.}
%
```

Appendix C - The generated jokes

6b jokes:

The lord cultivates a region. A baron land.
The grizzly speaks a verity. A bear truth.
The housewife captures a murderer. A cereal killer.
The diver joins a coalition. A coral society.
The textile worker fulfils a requirement. A dyer need.
The knight acquires a possession. A duel ownership.
The pheasant breathes oxygen. Fowl air.
The butcher commits a carelessness. A gross negligence.
The chimney-sweep digs a burrow. A grate hole.
The pony uses a vocal chord. A horse voice.
The miner plants a seed. A hole grain.
The social worker hates a foe. A hostel enemy.
The priest prevents a gossip. An idol rumour.
The banker listens to a father. A loan parent.
The lion harnesses energy. Mane force.
The general performs an operation. A major surgery.
The juvenile studies a writer. A minor poet.
The exotic dancer boards a boat. A navel ship.
The cleaner loves an individual. A pail person.
The baseball player looks at a musical sign. A base clef.
The lifeguard plants a shrub. A beach tree.
The sailor earns a diploma. A berth certificate.
The student hears wailing. Carrel singing.
The sailor visits a relation. A ferry godmother.
The sailor bears stress. Pier pressure.
The first born carries a sack. A heir bag.
The relation climbs a mound. An aunt hill.
The vocalist arrives at a settlement. A bass camp.
The grave digger enters a basement. A bier cellar.
The lover erects a home. A buss shelter.
The surgeon digs a garden. A doc yard.
The hero goes to a disco. A knight club.
The man carries a sack. A male bag.
The store keeper boards a ship. A sale boat.

Algorithm #7
Joan hates a foe in the hotel. A hostel enemy.
Joan looks at a musical sign on the station. A base clef.
Joan plants a shrub on the sandbank. A beach tree.
Joan earns a diploma in the bunk. A berth certificate.
Joan hears wailing in the booth. Carrel singing.
Joan visits a relation on the boat. A ferry godmother.
Joan bears stress on the boardwalk. Pier pressure.

Algorithm #8

Joan marries a first born in the sack. A heir bag.
Joan visits a relation on the mound. An aunt hill.
Joan hears a vocalist at the settlement. A bass camp.
Joan visits a grave in the basement. A bier cellar.
Joan gives a peck in the home. A buss shelter.
Joan visits a surgeon in the garden. A doc yard.
Joan kisses a hero at the disco. A knight club.
Joan charms a man in the sack. A male bag.
Joan has a bazaar on the ship. A sale boat.

Appendix D - A sample questionnaire

Please rate the following using this scale:

1. Not a joke. Does not make sense.
2. Recognizably a joke but a pathetic one.
3. OK. A child might like it.
4. Quite good.
5. Really good.

1. The lord cultivates a region. A baron land.
2. Joan hates a foe in the hotel. A hostel enemy.
3. The grizzly speaks a verity. A bear truth.
4. Joan looks at a musical sign on the station. A base clef.
5. The housewife captures a murderer. A cereal killer.
6. The diver joins a coalition. A coral society.
7. Joan plants a shrub on the sandbank. A beach tree.
8. The textile worker fulfils a requirement. A dyer need.
9. Joan earns a diploma in the bunk. A berth certificate.
10. The knight acquires a possession. A duel ownership.
11. The pheasant breathes oxygen. Fowl air.
12. Joan hears wailing in a booth. Carrel singing.
13. The butcher commits a carelessness. A gross negligence.
14. The chimney-sweep digs a burrow. A grate hole.
15. Joan visits a relation on the boat. A ferry godmother.
16. The pony uses a vocal chord. A horse voice.
17. The miner plants a seed. A hole grain.
18. Joan bears stress on the boardwalk. Pier pressure.

19. The social worker hates a foe. A hostel enemy.

20. The priest prevents gossip. An idol rumor.

21. Joan marries a first born in the sack. An heir bag.

22. The banker listens to a father. A loan parent.

23. The lion harnesses energy. Mane force.

24. The general performs an operation. A major surgery.

25. The juvenile studies a writer. A minor poet.

Please rate the following using this scale:

1. Not a joke. Does not make sense.
2. A joke but a pathetic one.
3. OK. A child might like it.
4. Quite good.
5. Really good.

1. The exotic dancer boards a boat. A navel ship.
2. Joan visits a relation on the mound. An aunt hill.
3. The cleaner loves an individual. A pail person.
4. Joan hears a vocalist at the settlement. A bass camp.
5. The baseball player looks at a musical sign. A base clef.
6. Joan visits a grave in the basement. A bier cellar.
7. The lifeguard plants a shrub. A beach tree.
8. The sailor earns a diploma. A berth certificate.
9. Joan gives a peck in the home. A buss shelter.
10. The student hears wailing. Carrel singing.
11. Joan kisses a hero at the disco. A knight club.
12. The sailor visits a relation. A ferry godmother.
13. Joan visits a surgeon in the garden. A doc yard.
14. The sailor bears a stress. Pier pressure.
15. Joan charms a man in the sack. A male bag.
16. The first-born carries a sack. An heir bag.
17. The relation climbs a mound. An aunt hill.
18. The vocalist arrives at a settlement. A bass camp.
19. The grave digger enters a basement. A bier cellar.

20. The lover erects a home. A buss shelter.

21. Joan has a bazaar on the ship. A sale boat.

22. The surgeon digs a garden. A doc yard.

23. A hero goes to a disco. A knight club.

24. A man carries a sack. A male bag.

25. A store keeper boards a ship. A sale boat.

Appendix E - The jokes' scores.

Joke #	The joke	Scores	Average
1	*The lord cultivates a region. A baron land.*	4 4 4 4 2 4 2	3.4
2	*Joan hates a foe in the hotel. A hostel enemy.*	5 2 4 2 2 4 4	3.3
3	*The grizzly speaks a verity. A bear truth.*	2 2 2 3 4 5 3	3
4	*Joan looks at a musical sign on the station. A base clef.*	1 2 1 1 3 1 1	1.4
5	*The housewife captures a murderer. A cereal killer.*	1 1 2 2 1 2 1	1.4
6	*The diver joins a coalition. A coral society.*	5 2 2 4 3 3 1	2.9
7	*Joan plants a shrub on the sandbank. A beach tree.*	5 3 3 2 3 3 2	3
8	*The textile worker fulfils a requirement. A dyer need.*	5 4 4 3 5 5 2	4
9	*Joan earns a diploma in the bunk. A berth certificate.*	5 3 3 1 2 3 1	2.6
10	*The knight acquires a possession. A duel ownership.*	5 1 4 5 2 1 1	2.7
11	*The pheasant breathes oxygen. Fowl air.*	4 3 5 5 3 5 3	4
12	*Joan hears wailing in a booth. Carrel singing.*	4 2 1 2 2 4 4	2.7
13	*The butcher commits a carelessness. A gross negligence.*	1 1 2 3 5 1 1	2
14	*The chimney-sweep digs a burrow. A grate hole.*	2 1 4 2 2 1 1	1.9
15	*Joan visits a relation on the boat. A ferry godmother.*	4 4 3 2 4 5 3	3.6
16	*The pony uses a vocal chord. A horse voice.*	2 4 2 3 3 4 3	3
17	*The miner plants a seed. A hole grain.*	1 2 3 3 3 3 2	2.4
18	*Joan bears stress on the boardwalk. Pier pressure.*	5 5 3 4 2 5 4	4
19	*The social worker hates a foe. A hostel enemy.*	3 2 4 2 1 1 1	2
20	*The priest prevents gossip. An idol rumor.*	4 3 5 2 4 1 2	3
21	*Joan marries a first born in the sack. An heir bag.*	5 2 4 5 4 1 1	3.1
22	*The banker listens to a father. A loan parent.*	4 2 1 4 4 3 1	2.7
23	*The lion harnesses energy. Mane force.*	5 4 4 3 4 1 1	3.1
24	*The general performs an operation. A major surgery.*	5 4 4 5 4 5 1	4
25	*The juvenile studies a writer. A minor poet.*	4 4 4 4 4 4 2	3.7
26	*The exotic dancer boards a boat. A navel ship.*	2 2 4 4 4 2 3 2	2.9
27	*Joan visits a relation on the mound. An aunt hill.*	2 2 2 3 3 3 4 3	2.8
28	*The cleaner loves an individual. A pail person.*	1 1 4 4 1 2 2 1	2
29	*Joan hears a vocalist at the settlement. A bass camp.*	1 2 2 3 2 1 3 3	2.1
30	*The baseball player looks at a musical sign. A base clef.*	2 1 2 2 2 3 4 2	2.3
31	*Joan visits a grave in the basement. A bier cellar.*	1 1 3 2 3 2 1 2	1.9
32	*The lifeguard plants a shrub. A beach tree.*	3 4 5 3 3 4 2 3	3.4
33	*The sailor earns a diploma. A berth certificate.*	2 2 4 4 4 4 4 5	3.6
34	*Joan gives a peck in the home. A buss shelter.*	1 1 5 1 2 1 1 2	1.8
35	*The student hears wailing. Carrel singing.*	1 1 1 3 2 3 5 4	2.5

36	*Joan kisses a hero at the disco. A knight club.*	4 4 5 5 3 3 5 5	4.3
37	*The sailor visits a relation. A ferry godmother.*	3 3 5 4 4 4 4 4	3.9
38	*Joan visits a surgeon in the garden. A doc yard.*	2 3 4 3 3 4 3 4	3.3
39	*The sailor bears a stress. Pier pressure.*	4 4 5 4 4 5 4 5	4.4
40	*Joan charms a man in the sack. A male bag.*	2 1 1 5 3 2 3 2	2.4
41	*The first-born carries a sack. An heir bag.*	2 1 5 1 2 2 4 4	2.5
42	*The relation climbs a mound. An aunt hill.*	1 1 2 2 2 1 3 3	1.9
43	*The vocalist arrives at a settlement. A bass camp.*	2 2 2 2 2 1 3 2	2
44	*The grave digger enters a basement. A bier cellar.*	1 1 3 2 3 1 1 2	1.8
45	*The lover erects a home. A buss shelter.*	1 1 5 1 4 1 1 2	2
46	*Joan has a bazaar on the ship. A sale boat.*	4 4 5 2 4 4 4 3	3.8
47	*The surgeon digs a garden. A doc yard.*	1 2 1 3 4 1 3 3	2.3
48	*The hero goes to a disco. A knight club.*	4 2 5 4 4 1 2 4	3.3
49	*The man carries a sack. A male bag.*	1 1 5 3 2 4 4 1	2.6
50	*The store keeper boards a ship. A sale boat.*	2 2 3 2 2 4 4 3	2.8