Generating Contextually Appropriate Intonation*

Scott Prevost & Mark Steedman
Computer and Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia PA 19104-6389, USA
(Internet: prevost@linc.cis.upenn.edu steedman@cis.upenn.edu)

Abstract

One source of unnaturalness in the output of text-to-speech systems stems from the involvement of algorithmically generated default intonation contours, applied under minimal control from syntax and semantics. It is a tribute both to the resilience of human language understanding and to the ingenuity of the inventors of these algorithms that the results are as intelligible as they are. However, the result is very frequently unnatural, and may on occasion mislead the hearer. This paper extends earlier work on the relation between syntax and intonation in language understanding in Combinatory Categorial Grammar (CCG). A generator with a simple and domain-independent discourse model can be used to direct synthesis of intonation contours for responses to data-base queries, to convey distinctions of contrast and emphasis determined by the discourse model.

1 The Problem

Consider the exchange shown in example (1). Capitals indicate stress, and brackets informally indicate the intonational phrasing. The intonation contour is indicated underneath using Pierrehumbert's notation ([8], [1], see [13] for a brief summary). L+H* and H* are different high pitch accents, and LH% and LL% (and its relative L) are rising and low boundaries respectively. The other annotations indicate that the intonational tunes L+H* LH% and H* LL% convey two distinct kinds of discourse information. First, both pitch accents mark any word that they occur on (or rather, its interpretation) for "focus", which in the context of such simple queries as example (1) usually implies contrast of some kind. Second, the tunes as a whole mark the constituent that bears them (or rather, its interpretation) as having a particular function in the discourse. We have argued at length elsewhere that, at least in this same restricted class of dialogues, the function of the L+H* LH% tune is to mark the "theme" – that is, "what the participants have agreed to talk about". The H* LL% tune (and its relative the H* L tune) mark the "rheme" – that is, "what the speaker has to say" about the theme. This phenomenon is a strong one: the same intonation contour sounds quite anomalous in the context of a question that does not establish the correct open proposition as the theme, such as *Which device has the fast processor?*. One further point is worth noting: the unit that we are calling the theme is not in this example a traditional syntactic constituent. Many problems in the analysis and synthesis of spoken language result from the partial independence of syntactic and intonational phrase boundaries.

The architecture of our system (shown in Figure 1) is for the most part self-explanatory, but we note that we follow a long tradition in separating the process of generation itself into two phases. The "strategic" phase is one in which the content of the utterance is planned, including the division into theme and rheme, and the assignment of contrastive focus. The "tactical" phase is one in which content is mapped

*Keywords: Speech-synthesis; Generation. We thank Mark Beutnagel and AT&T Bell Laboratories for allowing us access to the TTS speech synthesiser. The research was supported in part by NSF grant nos IRI90-18513, IRI90-16592, and IRI91-17110, DARPA grant no. N00014-90-J-1863, and ARO grant no. DAAL03-89-C0031.

(1) Q: I know that the OLD widget had a SLOW processor.
But what processor does the NEW widget include?

A:	(The	NEW	widget includes)	(a	FAST	processor)
		L+H*	LH%		H*	LL%
	Ground	*Focus*	*Ground*	*Ground*	*Focus*	*Ground*
		Theme			*Rheme*	

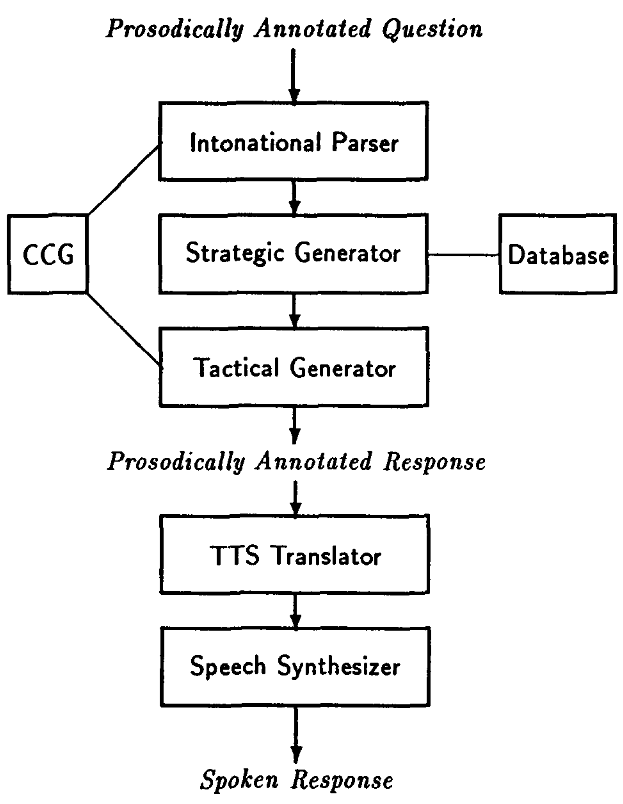

Figure 1: Architecture

onto strings of words.

2 CCG-Based Prosody

We will assume a standard CCG of the kind discussed in [11], [12], and [13]. For example, we shall write the category of a transitive verb like *prefers* either abbreviated, as in (2)a, or in full as in (2)b:

(2) a. $(S\backslash NP)/NP$
b. $(S : \mathit{include}'\ x\ y\backslash NP : y)/NP : x$

In b, syntactic types are paired with a semantic interpretation via the colon operator, and the category is that of a function from NPs (with interpretation x) to functions from NPs (with interpretation y) to Ss (with interpretation $\mathit{include}'\ x\ y$). Constants in interpretations bear primes, variables do not, and there is a convention of left associativity.

We also need the following two rules of functional application, where X and Y are variables over categories in either notation:

(3) FUNCTIONAL APPLICATION:
a. $X/Y \quad Y \quad \Rightarrow \quad X \quad (>)$
b. $Y \quad X\backslash Y \quad \Rightarrow \quad X \quad (<)$

CCG extends this strictly context-free categorial base in two respects. First, all arguments, such as NPs, bear only *type-raised* categories, such as $S/(S\backslash NP)$. Similarly, all functions into such categories, such as determiners, are functions into the raised categories, such as $(S/(S\backslash NP))/N$. For example, subject NPs bear the following category in the full notation:

(4) widgets $:= S : s/(S : s\backslash NP : \mathit{widgets}')$

The derivation of a simple transitive sentence appears as follows in the abbreviated notation:[1]

(5)
```
Widgets     include     sprockets
--------    ---------   ------------------
S/(S\NP)    (S\NP)/NP   (S\NP)\((S\NP)/NP)
            ------------------------------<
                       S\NP
-------------------------->
            S
```

Second, the combinatory rules are extended to include functional composition, as well as application. The following rule will be relevant below:

(6) FORWARD COMPOSITION (>B):
$X/Y \quad Y/Z \quad \Rightarrow_{\mathbf{B}} \quad X/Z$

This rule allows a *second* syntactic derivation for the above sentence, as follows:[2]

(7)
```
Widgets     include     sprockets
--------    ---------   ---------
S/(S\NP)    (S\NP)/NP   S\(S/NP)
-------------------->B
        S/NP
            ---------------------<
                     S
```

[1]The reader is encouraged to satisfy themselves using the full semantic notation that this derivation yields an S with the correct interpretation *include' sprockets' widgets'*. At first glance, it looks as though type-raising will expand the lexicon alarmingly. One way round this problem is discussed in [14].

[2]The reader is again strongly uged to satisfy themselves that the S yielded in the derivation bears the correct interpretation.

The reasons for making this move, which concern the grammar of coordinate constructions, the general class of rules from which the composition rule is drawn, and the problem of processing in the face of such associative rules, are discussed in the earlier papers, and need not concern us here. The point for present purposes is that the partition of the sentence into the object and a non-standard constituent $S : \mathit{include}'\ x'\ \mathit{widgets}'/NP : x$ makes this theory structurally and semantically perfectly suited to the demands of intonation, as exhibited in example (1).[3]

We can therefore directly incorporate intonational constituency in syntax, as follows (cf. [12], [13], and [15]). We assign to all constituents an autonomous prosodic category, expressing their potential for combination with other prosodic categories. Then we lock these two structural systems together via the following principle, which says that syntactic and prosodic constituency must be isomorphic:

(8) PROSODIC CONSTITUENT CONDITION: Combination of two syntactic categories via a syntactic combinatory rule is only allowed if their prosodic categories can also combine via a prosodic combinatory rule.

One way to do this is to make the boundaries arguments and the pitch accents functions over them. The boundaries are as follows:[4]

$$\text{(9)}\quad \begin{array}{lll} \text{L} & := & b : l \\ \text{LL\%} & := & b : ll \\ \text{LH\%} & := & b : lh \end{array}$$

As in CCG, categories consist of a structural type, here b for boundary, and an interpretation, associated via a colon. The pitch accents have the following functional types:[5]

$$\text{(10)}\quad \begin{array}{lll} \text{L+H*} & := & p : \mathit{theme}/b : lh \\ \text{H*} & := & p : \mathit{rheme}/b : l,\ P : \mathit{rheme}/b : ll \end{array}$$

We further assume, following Bird [2], that the presence of a pitch accent causes some element(s) in the translation of the category to be marked as focussed, a matter which we will for simplicity assume occurs at the level of the lexicon. For example, when *includes* bears a pitch accent, its category will be as follows:

$$\text{(11)}\quad (S : (*\mathit{include}')\ x\ y \backslash NP : y)/NP : x$$

The categories that result from the combination of a pitch accent and a boundary may or may not constitute entire prosodic phrases, since there may be a prenuclear null tone. There may also be a null tone separating the pitch accent(s) from the boundary. (Both possibilities are illustrated in (1)). We therefore assign the following category to the null tone, which can thereby apply to the right to any non-functional category of the form $X : Y$, and compose to their right with any function into such a category, including another null tone, to yield the same category:

$$\text{(12)}\quad \emptyset \ := \ X : Y/X : Y$$

It is this omnivorous category that allows intonational tunes to be spread over arbitrarily large constituents, since it allows the pitch accent's desire for a boundary to propagate via composition into the null tone category (see the earlier papers).

In order to allow the derivation to proceed above the level of complete prosodic phrases identifying themes and rhemes, we need two unary category-*changing* rules to mark the interpretation σ of the corresponding grammatical category with that discourse function and change the phonological category, thus:[6]

$$\text{(13)}\quad \begin{array}{c} \Sigma \\ p : X \end{array} \Rightarrow \begin{array}{c} \Sigma \\ p/p \end{array}$$

$$\text{(14)}\quad \begin{array}{c} \Sigma \\ P : X \end{array} \Rightarrow \begin{array}{c} \Sigma \\ p \end{array}$$

These rules change the prosodic category either to p, or to an endocentric function over p. (These types capture the fact that the LL% boundary can only occur at the end of a sentence, thereby correcting an overgeneration in the version of this theory in Steedman [13], noted by Bird [2]). The fact that p is an atom rather than a term of the form $X : Y$ is important, since it means that it can combine only with another p. This is vital to the preservation of the intonation structure.[7]

The application of the above two rules to a complete intonational phrase should be thought of as precipitating a side-effect whereby a copy of the category Σ is associated with the clause as its theme or rheme. (We gloss over details of how themes and rhemes are associated with a particular clause, as well as a number of further complications arising in sentences with more than one rheme).

In [13] and [15], a related set of rules of which the present ones form a subset are shown to be well-behaved with a wide range of examples. Example (15) gives the derivation for an example related to (7) (since the raised object category is not crucial, it has been replaced by NP to ease comprehension):[8] Note that it is the identification of the theme and

[3] A similar argument in a related categorial framework is made by Moortgat [6].

[4] These categories slightly depart from Pierrehumbert.

[5] Here we are ignoring the possibility of multiple pitch accents in the same prosodic phrase, but cf. [13].

[6] These rules represent both a departure from the earlier papers and a slight simplification of what is actually needed to allow prosodic phrases to combine correctly.

[7] The category has the same effect of preventing further composition into the null tone achieved in the earlier papers by a restriction on forward prosodic composition.

[8] Note the focus-marking effect of the pitch accents.

(15)
```
    Widgets                   include              sprockets
(    L+H*                        LH%           )  ( H* LL%  )
---------------------- -------------------------- -------------
S:s/(S:s\NP:*widget') (S:include'x y\NP:y)/NP:x NP:*sprockets'
    p:theme/b:lh                b:lh                P:rheme
------------------------------------------------>B
           S:include' x *widget'/NP:x
                   p:theme
           ===========================           ==============
           S:include' x *widget'/NP:x            NP:*sprockets'
                      p/p                             p
           -------------------------------------------------->
                  S: include' *sprockets' *widget'
                               p
```
Theme: $S : include\ z\ {*widget}/NP : z$
Rheme: $NP : {*sprockets}$

rheme at the stage *before* the final reduction that determines the information structure for the response, for it is at this point that discourse elements like the open proposition are explicit, and can be used in semantically-driven synthesis of intonation contour directly from the constituents.

Of course, such gushingly unambiguous intonation contours are comparitively rare in normal dialogues. Even in the context given in (7), a more usual response to the question would put low pitch – that is, the null tone in Pierrehumbert's terms – on everything except the focus of the rheme, *sprockets*, as in the following:

(16) Widgets include SPROCKETS

Such an utterance is of course ambiguous as to whether the theme is *widgets* or *what widgets include*. The earlier papers show that such "unmarked" themes, which include no pitch accent because they are entirely background, can be captured by a "Null Theme Promotion Rule", as follows:[9]

(17) $\begin{array}{ccc} \Sigma & & \Sigma \\ \mathtt{X:Y/X:Y} & \Rightarrow & \mathtt{p:theme} \end{array}$

3 Parsing

Having established the relationship between prosody, information structure and CCG syntax, we can now address the computational problem of automatically directing the synthesis of intonation contours for responses to database queries. Our computational model (shown in Figure 1) starts with a prosodically annotated *wh*-question given as a string of words with associated Pierrehumbert-style pitch accent and boundary markings. We employ a simple bottom-up shift-reduce parser of the kind presented in [14], making direct use of the CCG-Prosody theory described above, to identify the semantics of the question. The inclusion of prosodic categories in the grammar allows the parser to identify the information structure (theme and rheme) within the question as well. The focus and background information within the theme and rheme (if any) is further marked by the focus predicate * in the semantic representation. For example, given the question (18) below, the parser produces the semantic and information structure representations shown in (19).[10]

(18) I know that widgets contain cogs,
but what parts do WODGETS include?
L+H* LH% H* LL%

(19)

prop:	$s : \lambda x[part(x) \& include({*wodgets}, x)]$
theme:	$s : \lambda x[part(x) \& include({*wodgets}, x)]/$ $(s : incl({*wodgets}, x)/np : x)$
rheme:	$s : include({*wodgets}, x)/np : x$

The nondeterminism inherent in unmarked themes is handled by default: the present implementation of Null Theme Promotion delivers the ***longest*** unmarked theme that the syntax permits.[11]

4 Strategic Generation

The strategic phase of generating a response is somewhat simplified in the current implementation, and we have cut a number of corners. In particular, we currently assume that the question is the sole determinant of the information structure in the answer. This is undoubtedly an oversimplification. The complete specification of the semantic and information structures provided by the parser is used by the generator to determine the intelligible and prosodically natural response. For

[9]See the next section concerning the nondeterminism inherent in this rule.

[10]The alert reader will note that the notation for constants, variables, and functional application is slightly changed in these sections, to correspond to the Prolog implementation.

[11]This is a simplification, but a harmless one for the simplified query domains that we are dealing with here.

a *wh*-question, the semantic representation corresponds to a lambda expression in one or more variables ranging over individuals in the database, and has the structure of a Prolog query which we can evaluate to determine the possible instantiations of the open proposition. The instantiated proposition determines the semantic proposition to be conveyed in the response. For the example above, this is $part(sprockets)\&include(*wodgets, *sprockets)$ – "Wodgets include sprockets".

Note that the derived semantics includes the necessary occurrences of the focus predicate *, determined as follows. All terms that are focused in the question semantics are focused in the response semantics. Intuitively, the instantiated variable in the response semantics must also be focused since it represents the information which is *new* in the response. For more complex rhemes such as quantified NPs with modifiers, we focus those elements of the semantic representation that are new in the current context. (That is, ones which did not figure in the interpretation of the original query). Thus, given a question such as (1), we choose to focus the modifier "fast" rather than the noun "processor" in the rheme. Similary, in the exchange below, we focus "processor" instead of "fast" because of its newness in the context.

(20) Q: What fast component does the widget include?
A: The widget includes a fast PROCESSOR.

To determine an appropriate intonation contour for the utterance, we must further determine the appropriate information structure. Fortunately, for the simple question-answering task, the information structure of the response can be assumed to be completely determined by the original query. The theme of a question corresponds to "what the question is about" – in this case, "parts". The rheme of a question corresponds to "what the speaker wants to know about the theme" – here, "What wodgets include". It follows that we expect the rheme of the question to determine the theme of the response. For example (18), the theme of the response should be $S : include(*wodgets, x)/NP : x$, as in (21) below. Note that we simplify the strategic generation problem by including the syntactic category in our represention of the theme (as determined by the syntactic category of the rheme of the original question).[12] Given the syntactic and semantic representation of the theme of the response, the CCG combination rules can easily be invoked to determine the rheme of the response. The rheme is simply the complement of the theme with respect to the overall semantics of the response, as in (21) below, obtained by instantiating the result and one input of the appropriate combinatory rule (cf. [7]):[13]

(21) prop: $s : include(*wodgets, *sprockets)$
theme: $s : include(*wodgets, x)/np : x$
rheme: $np : *sprockets$

5 Tactical Generation and CCG

Just as the shift-reduce parser sketched above can readily be made to construct the interpretations and information structures shown in the examples, specifically marking themes, rhemes and their foci, so it is relatively easy to do the reverse—to generate prosidically annotated strings from a focus-marked semantic representation of themes and rhemes.

For simplicity, we start by describing the syntactic and semantic aspects of the generator, ignoring prosody for the moment. In constructing a tactical generation schema, several design options are available, including bottom-up, top-down and semantic head-driven models ([3], [10]). We adopt a hybrid approach, employing a basic top-down strategy that takes advantage of the CCG notion of "functional head" to avoid fruitless search. While this technique exhibits some inefficiencies characteristic of a depth-first search, it has several significant advantages. First, it does not rely on a specific semantic representation, and requires only that the semantics be compositional and representable in Prolog. Thus the generating procedure is independent of the particular grammar. This modular character of the system has been very useful in developing the competence grammar proposed in the preceding section, and offers a basis for proving the completeness of the implementation with respect to the competence theory.

The tactical generation program is written in Prolog, and works as follows. Starting with a syntactic constituent (initially *s*) and a semantic formula, we utilize the CCG reduction rules to determine possible subconstituents that can combine to yield the original constituent, invoking the generator recursively to generate the proposed subconstituents. The base case of the recursion occurs when a category we wish to generate unifies with a category in the lexicon. For example, suppose we wish to generate an utterance corresponding to the category *s:walks'(mary')*. Since the given category does not unify with any category in the lexicon, the program proposes possible subconstituents by checking the CCG combination rules in some pre-determined order. By the backward function application rule, we might hypothesize that the categories x and *s:walks'(mary')*$\backslash x$ are the subconstituents of *s:walks'(mary')*, where x is

[12]Here we are cutting another corner: the theme, and hence the rheme, are fully specified syntactically, as well as semantically, as a result of the analysis of the question: in a more general system, we would presumably need to specify syntactic type from scratch, starting from pure semantics.

[13]Again the example is simplified by the use of a nonraised category for the object.

```
(22) gen(s:def(x, ((engine(x)&new(x))&shiny(x))&
                 def(y, ((gear(y)&rotating(y))&largest(y))&contains(x,y)))).
     RESULT: the shiny new engine contains the largest rotating gear.
```

```
(23) gen(s:exists(z, (engineer(z)&brilliant(z))&exists(x,(design(x)&revolutionary(x))&
                                                 def(y, (engine(y)&new(y))&gave(z,y,x))))).
     RESULT: a brilliant engineer gave the new engine a revolutionary design.
```

```
(24) gen(s:def(x,(widget(x)&*new(x))&probably(contains(x,y)))/np:y @ p:theme).
     RESULT: the new@lhstar widget probably contains@lhb.
```

```
(25) gen(np:(x^s)^def(x,(processor(x)&*fastest(x))&s) @ ph:rheme).
     RESULT: the fastest@hstar processor@llb.
```

```
(26) gen(s:def(x,(widget(x)&new(x))& *probably(contains(x,y)))/np:y @ p:theme).
     RESULT: the new widget probably@lhstar contains@lhb.
```

```
(27) gen(s:def(x,(*widget(x)&new(x))&contains(x,y))/np:y @ p:theme).
     RESULT: the new widget@lhstar contains@lhb.
```

some variable. If we recursively call the generator on *s:walks'(mary')*$\backslash x$, we find that it unifies with the category *s:walks'(y)*$\backslash$*np:y* in the lexicon, corresponding to the lexical item *walks*. This unification forces the complementary category x to unify with *np:mary'*, which yields the lexical item *mary* when the generator is recursively invoked. Concatenating the results of generating the proposed subconstituents therefore gives the string *"Mary walks."*

The top-down nature of the generation scheme has a number of important consequences. First, the order in which we generate the postulated subconstituents determines whether the generation succeeds. Had we chosen to generate x before *s:walks'(mary')*$\backslash x$, we would have entered a potentially infinite recursion, since x unifies with every category in the lexicon. For this reason, our generator always chooses to recursively generate the subconstituent that acts as the functional head before the subconstituent that acts as the argument under the CCG combinatory rules. By strictly observing this principle, we ensure that as much semantic information as possible is deployed, thereby constraining the search space by prohibiting spurious unifications with incorrect items in the lexicon. For this reason, we refer to our generation scheme as a "functional head"-driven, top-down approach.

One disadvantage of the top-down generation technique is its susceptibility to the non-termination problem. If a given path through the search space does not lead to unification with an item in the lexicon, some condition which aborts the path in question at some search depth must be imposed. Note that whenever the CCG function application rules are used to propose possible subconstituents to be recursively generated, the subconstituent acting as the functional head has one more curried argument than its parent. Since we know that in English there is a limit to the number of arguments that a functional category can take, we can abort fruitless search paths by imposing a limit on the number of curried arguments that a CCG category can possess. The current implementation allows categories with up to three arguments, the minimum needed for constructions involving di-transitive verbs. Note that this strategy does not prohibit the generation of categories whose arguments themselves are complex categories. Thus, we allow categories such as $((s\backslash np)/np)\backslash(((s\backslash np)/np)/np)$ for raised indirect objects, but not categories such as $(((s\backslash np)/np)/np)/np$.

When the CCG composition rule is used to propose possible subconstituents, the subconstituents do *not* have more curried arguments than their parent. Consequently, imposing a bound of the type described above will not necessarily avoid endless recursion in all cases. Suppose, for example that we wish to generate a category of the form s/x, where s is a fully instantiated expression and x is a variable. If the function application rules fail to produce subconstituents that generate the category, we rely on the CCG composition rule to propose the possible subconstituents s/y and y/x. Since s/x and s/y are identical categories to within renaming of variables, the recursion will continue indefinitely. We rectify this situation by invoking the composition rule only if the original category has an instantiation for both its argument and result. Such a solution imposes limitations on the types of derivations allowed by the system, but retains the simplicity and transparency of the algorithm. Merely imposing a limit on the depth of the recursion provides a more general solution. Examples of the types of sentences that can be generated appear in (22) and (23).

This procedure can immediately be applied to the prosodically augmented grammar. To do so, we merely enforce the Prosodic Constituent Condition at each step in the generation. That is, whenever a pair of subconstituents are considered (by reversing the CCG combination rules), a pair of prosodic subconstituents are also considered and recursively generated using the prosodic combinatory rules. Examples (24) and (25) illustrate the generation of intonation for the theme and rheme of the utterance "The NEW widget probably contains the FASTEST processor".[14] Examples (26) and (27) manifest the intonational results of moving the thematic focus among the various propositions in the semantic representation of the theme "The new widget probably contains ...".

6 Synthesis

We showed in the previous section how constituents of the type shown in (21) can generate intonationally annotated strings. The resulting string for the current example is "wodgets@lhstar include@lhb sprockets@[hstar, llb]." The final aspect of generation involves translating such a string into a form usable by a suitable speech synthesiser. Currently, we use the Bell Laboratories TTS system ([5]) as a post-processor to synthesise the speech wave. Example (28) shows the translated output for the same example, as it is sent to this synthesiser.

```
(28) \!> \!*L+H*1 wodgets \!fL1 include
     \!pL1 \!bH1 \!*H*2 sprockets
     \!pL1 \!bL1 . \( *[20] \)
```

We stress that we use TTS as an unmodified output device, without any fine tuning other than in the lexicon. While TTS is particularly easy to use with Pierrehumbert's notation, we are confident that our system can easily be adapted to other synthesisers.

7 Results

The system just described produces sharp and natural-sounding distinctions of intonation contour in minimal pairs of queries like the following:

(29) Q: I know that widgets contain cogs, but
what gadgets include SPROCKETS?
```
L+H*    LH%           H*       LL%
```

prop: $s : \lambda x[gadget(x) \& incl(x, *sprockets)]$
theme: $s : \lambda x[gadget(x) \& incl(x, *sprockets)] / (s : incl(x, *sprockets) \backslash np : x)$
rheme: $s : incl(x, *sprockets) \backslash np : x$

A: prop: $s : incl(*wodgets, *sprockets)$
theme: $s : incl(x, *sprockets) \backslash np : x$
rheme: $np : *wodgets$

```
WODGETS include SPROCKETS.
  H*       L          L+H*    LH%
```

(30) Q: I know that widgets contain cogs, but
what parts do WODGETS include?
```
L+H* LH%      H*            LL%
```

prop: $s : \lambda x[part(x) \& incl(*wodgets, x)]$
theme: $s : \lambda x[part(x) \& incl(*wodgets, x)] / (s : incl(*wodgets, x) / np : x)$
rheme: $s : incl(*wodgets, x) / np : x$

A: prop: $s : incl(*wodgets, *sprockets)$
theme: $s : incl(*wodgets, x) / np : x$
rheme: $np : *sprockets$

```
WODGETS include SPROCKETS.
  L+H*          LH% H*       LL%
```

(31) Q: I know that programmers use widgets,
but which people DESIGN widgets?
```
        L+H* LH%    H*          LL%
```

prop: $s : \lambda x[people(x) \& *design(x, widgets)]$
theme: $s : \lambda x[people(x) \& *design(x, widgets)] / (s : *design(x, widgets) \backslash np : x)$
rheme: $s : *design(x, widgets) \backslash np : x$

A: prop: $s : *design(*engineers, widgets)$
theme: $s : *design(x, widgets) \backslash np : x$
rheme: $np : *engineers$

```
ENGINEERS DESIGN widgets.
        H* L    L+H*       LH%
```

(32) Q: If engineers design widgets,
which people design WODGETS?
```
 L+H* LH%              H*   LL%
```

prop: $s : \lambda x[people(x) \& design(x, *wodgets)]$
theme: $s : \lambda x[people(x) \& design(x, *wodgets)] / (s : design(x, *wodgets) \backslash np : x)$
rheme: $s : design(x, *wodgets) \backslash np : x$

A: prop: $s : design(*programmers, *wodgets)$
theme: $s : design(x, *wodgets) \backslash np : x$
rheme: $np : *programmers$

```
PROGRAMMERS design WODGETS.
     H*           L           L+H*  LH%
```

Examples (29) and (30) illustrate the ability of our system to produce appropriately different intonation contours for identical strings of words depending on the context, which determines the information structure of the response. If the responses in these examples are interchanged, the result sounds distinctly

[14]The @ symbol separates syntactic categories from their corresponding prosodic categories and lexical items from their pitch/boundary markings.

unnatural in the given contexts. From examples (31) and (32), it will be apparent that our system has the ability to make distinctions in focus placement within themes and rhemes based on context. The issue of focus placement can be crucial in more complex themes and rhemes, as shown below:

```
(33) Q: I know the old widget has the slowest processor,
        but which widget has the FASTEST processor?
             L+H* LH%          H*            LL%
     A: The NEW widget has the FASTEST processor.
            H*      L        L+H*           LH%

(34) Q: The old widget has the slowest processor,
        but which processor does the NEW widget have?
             L+H*       LH%          H*          LL%
     A: The NEW widget has the FASTEST processor.
            L+H*       LH%      H*               LL%

(35) Q: The new WODGET has the slowest processor,
        but which processor does the new WIDGET have?
             L+H*       LH%              H*     LL%
     A: The new WIDGET has the FASTEST processor.
                L+H* LH%      H*                LL%
```

As noted earlier, such precisely specified themes are uncommon in normal dialogue. Consequently, the Null Tone Promotion rule is employed for unmarked themes, allowing the types of responses in (36) and (37) below. The theme is taken to be the longest possible prosodically unmarked constituent allowed by the syntax.

```
(36) Q: I know that programmers use widgets,
        but which people DESIGN widgets?
                         H*      LL%
     A: ENGINEERS design widgets.
           H* L

(37) Q: If engineers design widgets,
        which people design WODGETS?
                              H* LL%
     A: PROGRAMMERS design wodgets.
           H*        L
```

Although we have only briefly discussed the possibility of multiple pitch accents within a theme or rheme, we have included such a capability in our implementation. The system's ability to handle multiple pitch accents is illustrated by the following example.

```
(38) Q: I know that students USE WODGETS,
        but which people DESIGN WIDGETS?
                          H*     H* LL%
     A: ENGINEERS design widgets.
           H* L
```

While many important problems remain, examples like these show that it is possible to produce synthesized speech with contextually appropriate intonational contours using a combinatory theory of prosody and information structure that is completely transparent to syntax and semantics. The model of utterance generation for Combinatory Categorial Grammars presented here implements the prosodic theory in a similarly transparent and straightforward manner.

8 References

[1] Beckman, Mary and Janet Pierrehumbert: 1986, 'Intonational Structure in Japanese and English', *Phonology Yearbook*, 3, 255-310.

[2] Bird, Steven: 1991, 'Focus and phrasing in Unification Categorial Grammar', in Steven Bird (ed.), *Declarative Perspectives on Phonology*, Working Papers in Cognitive Science 7, University of Edinburgh. 139-166.

[3] Gerdeman, Dale and Erhard Hinrichs: 1990. Functor-driven Natural Language Generation with Categorial Unification Grammars. *Proceedings of COLING 90, Helsinki*, 145-150.

[4] Jackendoff, Ray: 1972, *Semantic Interpretation in Generative Grammar*, MIT Press, Cambridge MA.

[5] Liberman, Mark and A.L. Buchsbaum: 1985, 'Structure and Usage of Current Bell Labs Text to Speech Programs', Technical Memorandum, TM 11225-850731-11, AT&T Bell Laboratories.

[6] Moortgat, Michael: 1989, *Categorial Investigations*, Foris, Dordrecht.

[7] Pareschi, Remo and Mark Steedman: 1987, 'A Lazy Way to Chart-parse with Categorial Grammars', *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford CA, July 1987, 81-88.

[8] Pierrehumbert, Janet: 1980, *The Phonology and Phonetics of English Intonation*, Ph.D dissertation, MIT. (Dist. by Indiana University Linguistics Club, Bloomington, IN.)

[9] Pierrehumbert, Janet, and Julia Hirschberg, 1990, 'The Meaning of Intonational Contours in the Interpretation of Discourse', in Philip Cohen, Jerry Morgan, and Martha Pollack (eds.), *Intentions in Communication*, MIT Press Cambridge MA, 271-312.

[10] Shieber, Stuart and Yves Schabes: 1991, 'Generation and Synchronous Tree-Adjoining Grammars', *Computational Intelligence*, 4, 220-228.

[11] Steedman, Mark: 1990. 'Gapping as Constituent Coordination', *Linguistics & Philosophy*, 13, 207-263.

[12] Steedman, Mark: 1990, 'Structure and Intonation in Spoken Language Understanding', *Proceedings of the 25th Annual Conference of the Association for Computational Linguistics*, Pittsburgh, PA, June 1990, 9-17.

[13] Steedman, Mark: 1991, Structure and Intonation, *Language*, 68, 260-296.

[14] Steedman, Mark: 1991, 'Type-raising and Directionality in Categorial Grammar', *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley CA, June 1991, 71-78.

[15] Steedman, Mark: 1991, 'Surface Structure, Intonation, and "Focus"', in Ewan Klein and F. Veltman (eds.), *Natural Language and Speech*, Proceedings of the ESPRIT Symposium, Brussels, Nov. 1991. 21-38, 260-296.