

# **Anaphoric reference and ellipsis resolution in a telephone-based spoken language system for accessing email**

Author: Sandra Williams, BT Labs.

## ***Abstract***

This paper describes the anaphoric reference and ellipsis resolution component of a Spoken Language System, currently being developed at BT Labs, which provides telephone-based access to email. The system, MailSec, is an amalgamation of three technologies: continuous speaker-independent speech recognition, Natural Language Processing (NLP) and text-to-speech.

The anaphoric reference and ellipsis resolution module is part of the NLP component. It is a separate module, controlled by a dialogue manager and resolution is performed by searching the data structures the dialogue manager builds.

References in the email conversations tend to be made to emails themselves, or to email headers, or folders, or to properties associated with emails (e.g. senders and recipients, dates and times).

The resolution of anaphoric references in this system is closely linked to the queries it makes in PROLOG to a database of user's email files. A set of emails can be recovered from a past query. In the case of: "Are any of **them** about elephants?" the "them" can be resolved by retrieving a past query, and the new query can be formed by adding the extra constraint that the emails found must be about elephants.

## ***1. Introduction***

Automatic access to information by telephone has been available since the mid 1980s. Earlier systems had touch-tone input only, then isolated word speech recognisers were developed that were accurate enough to allow callers to say numbers (digit by digit) and a few keywords ('yes', 'no', and so on). Current commercial systems can handle much larger vocabularies and speaker-independent speech recognisers can cope with a mixture of isolated words, connected digits, and connected alphabetical letters. An example of this is an automated directory enquiry system currently in use at BT Labs. [1] which holds a database of several thousand people's names and telephone numbers.

Dialogues in this type of information enquiry system are successful when very restricted amounts and types of information are to be supplied to the caller (e.g. a telephone number) and the information a caller has to input is similarly restricted (e.g. a first name and a surname).

For more complex dialogue systems where the information is more varied and various commands may also be available, and where the caller may want to 'browse', a more flexible dialogue approach is necessary. MailSec provides mixed-initiative dialogues where the caller can take the initiative and does not necessarily have to answer the system's questions or say what the system wants. The user can take control at any time by asking a question or giving a command [2].

In more natural dialogues, speaker's utterances normally contain discourse phenomena known as anaphoric references and ellipsis. Corpora of human-human conversations [3] or of dialogues between a human and a simulated (Wizard of Oz) machine [4] demonstrate the prevalence of these phenomena. Anaphoric references occur when a speaker refers back to something

mentioned earlier in the conversation, e.g. 'Read *that message*!', 'Are there any emails from *her*?'. Ellipsis occurs when a word or phrase is 'left out', but can be understood from what has gone before, e.g. 'Are there any ... from Peter?' where the noun 'emails' has been left out.

Other dialogue systems vary greatly in their handling of anaphoric references and ellipsis. Some, such as Sun Microsystem's Speech Acts [5] do not attempt reference resolution at all. Others, such as Lewin and Pulman's ellipsis resolver [6], provide a very comprehensive approach. Our system is similar to Lewin and Pulman's in that the resolution component is entirely separate from the dialogue manager, but our approach is, at present, more domain dependent and our linguistic processing is not so deep.

This paper concentrates on reference and ellipsis resolution in MailSec. For an overview of the MailSec system, see part 2. A description of the Natural Language Processing (NLP) part of MailSec is presented in Part 3. Part 4 describes types of reference resolution. Part 5 describes the process of reference and ellipsis resolution in MailSec. Our conclusions are presented in part 6.

## 2. Overview of the telephone email system: MailSec

MailSec is a Spoken Language System for accessing email over the telephone [7]. It is designed for use by people who are away from their offices and who do not have access to a computer and modem, people who nevertheless want to have access to their email. MailSec enables them to ring up and have their emails read out to them. In addition to this, emails can be forwarded, deleted, replied to, and filed.

The application is similar in functionality to the email part of Sun Microsystems's Speech Acts [5], but our system allows callers to converse in a more natural manner. A fragment of typical conversation between a caller and MailSec might progress as follows:

User:	Do I have any messages from Anna?
MailSec:	You have one new message from Anna Cordon entitled 'MT Meeting'.
User:	Read it.

It can be seen from the above exchange that when speaking to MailSec, the caller can refer in a comfortable and natural way to an email mentioned earlier in the conversation, illustrated here by 'it' in 'Read it'.

Spoken Language Systems combine the three technologies of Speech Recognition, Natural Language Processing (NLP) and Text-To-Speech as shown in Figure 1 below.

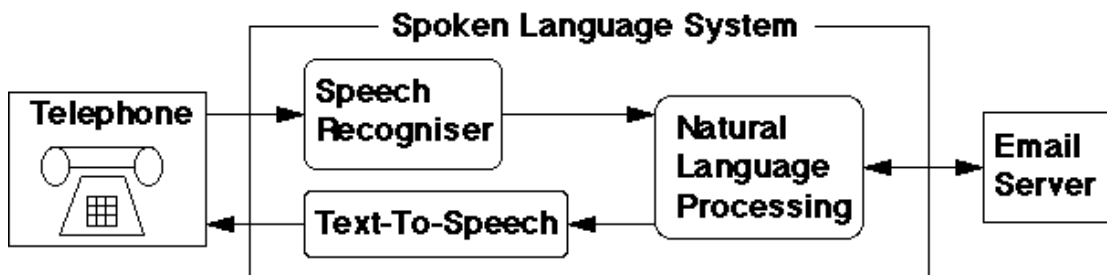


Figure 1. Schematic diagram showing MailSec's major technologies

Figure 1 is a schematic diagram and does not reflect the architecture of the system [7]. It is a simplification and many details such as the telephony interface software, interfaces to various

different email servers, and interfaces between the Spoken Language System components themselves have not been shown.

The speech recognition component of MailSec is the BT Labs continuous speech recogniser [8] which allows users to speak in a natural manner without having to leave pauses between words. It is a speaker-independent recogniser which means that it can recognise a wide variety of English speakers regardless of accent or voice pitch and modulation. This has obvious advantages over a speaker-dependent recogniser which would have to be trained on all callers' voices before it would be able to recognise them. The function of the speech recogniser is to recognise the incoming speech, and transcribe it from a speech waveform into text.

For outgoing speech, MailSec uses the BT Labs Text-To-Speech (TTS) system, Laureate [9]. Laureate's speech is derived from a real human voice and is one of the most natural-sounding TTS systems currently available. Laureate converts a string of outgoing text into a speech waveform to be played back the caller.

The NLP part of the system interprets the meaning of the incoming utterance from the caller which has been transcribed into text by the speech recogniser. It generates queries for the email server, and constructs the replies as text for Laureate to convert to a speech waveform. This paper is concerned with that part of the NLP component which resolves anaphoric references and ellipsis in the incoming utterances.

### 3. Overview of the NLP Component of MailSec

The anaphoric reference and ellipsis resolution module is part of the NLP component of MailSec. Other modules include: a dialogue manager, a parsing/semantics module, a database query module, a cooperative response module, and a text generation module. These modules, together with the data structures built by the dialogue manager are shown below in figure 2.

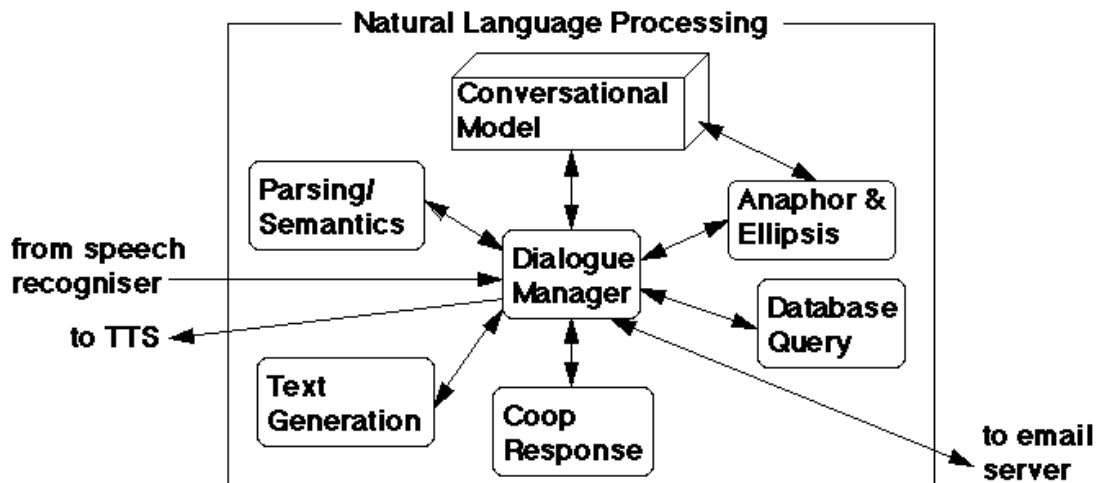


Figure 2. Schematic diagram of MailSec's NLP component

Figure 2 is a schematic diagram and leaves out many details of the individual modules and those data sources used by modules other than the dialogue manager and the anaphoric reference and ellipsis resolution module.

A typical processing sequence is controlled by the dialogue manager and proceeds in a clockwise direction in figure 2 from the speech recogniser input, through the parsing/semantics module, then to anaphor/ellipsis resolution, then to database query, then to cooperative response analysis, then to text generation and finally out to text-to-speech. The conversational model is modified

and consulted at various stages throughout the processing. Note however that the NLP modules may be called by the dialogue manager in a flexible way. An example of this is when the dialogue manager uses the cooperative response module to modify the query, and then database query is attempted again.

Brief descriptions of what each module does and how the modules interact are given below.

### 3.1 The Dialogue Manager

The dialogue manager maintains the conversation between MailSec and the caller. It coordinates the operation of all other system components and builds a dynamic model of the conversation as it progresses.

### 3.2 The Parsing and Semantics Module

On receiving input from the speech recogniser, the dialogue manager sends it to the parsing and semantics module which interprets the meaning of the caller's utterance. The meaning is represented by what we have called an Extended Logical Form (ELF). The ELF is constructed using the standard PROLOG notation where variables are capitalised, and atoms are lower case. An example of an ELF for 'List any emails from Anna' is:

ELF:	list(A)
Expect List:	[plural(A)]
Spec List:	[message(A),from(A,B),name(B,anna),gender(B,feminine)].

The ELF is a non-recursive form with three parts. The first shows that there exists something, A, that is to be listed. The second is a list of facts indicating what the caller is expecting, the Expect List. In this example, the caller is expecting A to be plural ('emails' in the input string). The Expect List is used by the text generator (together with the database query result and the Spec List) to build an appropriate response. The third part is a specification of what the caller requires ('emails from Anna'). The Expect List and Spec List contain a number of conjoined constraints. In this case, there exists something, A, where A is a message, and A is from B where B is named 'anna' and feminine. The Spec List is used by the dialogue manager as the base for building the database query.

### 3.3 Conversational Modelling

Individual utterances cannot be totally meaningful and coherent unless they are interpreted in their correct context within the conversation. If a caller says 'Anna', or 'Yes', or 'Read the next', the ELFs alone do not give enough information for MailSec to understand and know what to do next. Therefore on receiving the ELF for the input utterance, the dialogue manager interprets its meaning in the context of the conversation and builds it in to the conversational model. The model ensures that answers are matched up with corresponding questions, and so on. The Conversational Model is a representation of the dialogue meaning above the level of individual utterances.

Our model of the conversation is built dynamically as the conversation progresses and it is based on the theory of Games Structure in Conversation devised by Kowtko et al [10]. Part of the model is the History List which contains all previous utterances, and the other part is a stack of conversational games, see [2].

### 3.4 Anaphoric Reference and Ellipsis Resolution Module

This is the module that resolves ellipsis and anaphoric references.

Not all incoming utterances undergo reference resolution. For instance if MailSec has just asked a yes-no question (e.g. `Do you want to delete the message from Alison Simons entitled "Meeting"') and the caller just answers 'yes' or 'no', then this module is bypassed.

The ellipsis and reference resolution module is described in detail in part 5.

### 3.5 Database Query Module

The database query module is the part of the system that searches the email database for emails that match the query supplied by the dialogue manager. The form of the queries is based on the Spec List of the ELF described above.

When the database query finds a set of emails corresponding to the list of constraints in the Spec List. It returns a list of the email IDs and the Spec List instantiated with the information it finds. For instance, for the utterance `List messages from Glen', the Spec List is:

```
[message(A), from(A,B), gender(B,male), named(B,glen)]
```

Suppose the database query finds two messages from Glen Long, it would return a list of the message IDs (say [1003, 1006]) and would instantiate the Spec List with Glen's full name:

```
[message(A), from(A, [glen, long]), gender([glen, long], male), named([glen, long], glen)]
```

Everywhere a B occurs in the Spec List, it is replaced by the full name: [glen, long].

Suppose the database query finds two messages from Glen Long and one message from Glen Adams, then it would return two list of message ID s (say [1003, 1006] and [1025]) and two instantiated Spec Lists:

```
[message(A), from(A, [glen, long]), gender([glen, long], male), named([glen, long], glen)]  
  
[message(A), from(A, [glen, adams]), gender([glen, adams], male), named([glen, adams], glen)]
```

The first is instantiated with the full name [glen, long] and the second with the full name [glen, adams].

### 3.6 Cooperative Response Module

If the database query does not find anything, the cooperative response module is used to modify the query in order that a cooperative response might be given to the caller, see [2].

### 3.7 Text Generation Module

The text generation module constructs the text for the system's response to the caller, see [2].

## 4. Types of reference

Anaphoric references occur in conversations when a speaker makes mention of something which can only be uniquely identified by what has been said previously.

Halliday and Hasan [11] identify three major groups of references: personals, demonstratives and

comparatives. These groups are summarised in Table 1 below:

Personal Reference	I, me, mine, my, you, yours, your, we, us, ours, our, he him, his, she, he, hers, they, them, theirs, their, it, its, one, one's
Demonstrative Reference	this, these, here, now, that, those, there, then, the
Comparative Reference	same, identical, equal, similar, additional, other, different, else, better, more, identically, similarly, likewise, so, such, differently, otherwise, so, less, equally

Table 1: Types of reference as defined by Halliday and Hasan [11]

Personal references indicated in Table 1 include pronouns and possessive pronouns. We also include reflexive pronouns (myself, yourself, himself, itself, etc.) in this group. Demonstrative references are indicated by determiners and adverbs. Comparative references are indicated by certain adjectives and certain adverbs as indicated.

Examples from the email domain from a corpus we collected include: "Read **her** message.", "Are there any **other messages**?", "What does **he** say?", "Not **that folder!**", "Read **the one** David sent **then**?", "Read **me the message** from Anna." and "Have **you** got any new messages for **me**". Our system does not cover all of these at the present time, but it is still under development and we certainly hope to cover all references (and ellipsis) present in our corpus in the future. The next section, part 5, describes our present coverage.

### 5. Resolution of Ellipsis and Anaphoric References

The resolution of anaphoric references in this system is closely linked to the queries it makes in PROLOG to the database of user's email files. A set of emails can be recovered from a past query, by reprocessing that query. A query is based on the Spec List part of the logical representation of the meaning of the input utterance, the ELF.

The inputs to the reference resolution component are ELF's. Ellipsis and references are not indicated explicitly in the ELF, they are implicit, i.e. information is missing from the Spec List or certain variables are uninstantiated in the Spec List. It is easier to see what we mean with the two following examples.

First we will show an example of ellipsis. Table 2 below shows ELF's for two sentences, The first has no ellipsis, the second demonstrates ellipsis with a missing noun `messages':

	No Ellipsis: Do I have a message from Peter?		Ellipsis: Do I have any from Peter?
ELF:	ynq	ELF:	ynq(A)
Expect List:	[singular(A)]	Expect List:	[]
Spec List:	[message(A), from(A,B), gender(B,masculine), named(B,peter)]	Spec List:	[from(A,B), gender(B,masculine), named(B,peter)]

Table 2: ELF representation of ellipsis

In the second ELF, the predicate, message(A), is missing from the Spec List. The ELF reads: there is something, A, which is from B, who is masculine and named Peter.

It is the task of the reference resolution module to fill in missing information, message(A), in the ELF.

Spec Lists for all utterances which the dialogue manager sends to the anaphoric reference and ellipsis resolution module are searched for an entity which is known to exist in the domain. For the email domain, these entities include: `message`, `email`, `header`, `folder`, etc. If an entity is missing from the Spec List, the History List is searched to find the most recent Spec List containing an entity, and this entity is inserted into the current Spec List. In effect, if the last thing the caller spoke about was `messages` (e.g. `List my messages`) and the next utterance contains an ellipse (e.g. `Do I have any from Anna?`) then message(A) will be found in the most recent Spec List on the History List and this will be added to the current Spec List.

Note that some inputs from the caller are not sent for resolution, these include noun phrase answers to questions MailSec has asked the user. For example, if MailSec asked `What is his name?`, the dialogue manager would be expecting the caller to say a name such as `Peter`. If the name is input as expected, then the anaphoric reference and ellipsis resolution module will not be called.

Our second example contains an anaphoric reference. Table 3 below shows ELFs for two sentences: the first with no anaphoric references and the second with an anaphoric reference:

	No Anaphoric reference: Read Peter's email.		Anaphoric reference: Read his email
ELF:	read(A)	ELF:	read(A)
Expect List:	[singular(A)]	Expect List:	[singular(A)]
Spec List:	[message(A), from(A,B), gender(B,male), named(B,peter)]	Spec List:	[message(A), from(A,B), gender(B,male), named(B,C)]

Table 3: ELF representation of an anaphoric reference

The second ELF differs from the first by having a variable, C, instead of the name, Peter. It is the task of the reference resolution module to instantiate this variable in the ELF. The History List is searched for the most recent Spec List containing a male name and this name is instantiated with the variable, C, in the current Spec List.

All references are resolved by finding the most recent matching item. For example the most recent masculine name for `him`, and the most recent single email for `it`. Obviously this may not always be the correct referent. However, MailSec always states explicitly which referent it is using, so the caller can be in no doubt when it has got the referent wrong. The caller can correct MailSec by stating more fully what she/he means. The following is an example of this:

Caller:	List emails from Anna.
MailSec:	You have one message from Anna Cordon entitled `Aardvarks` and one from Anna Bloggs entitled `Summariser`.

Caller:	Delete it.
MailSec:	Delete the message from Anna Bloggs entitled `Summariser`?
Caller:	No. Delete the one from Anna Cordon.

Careful thought needs to be given to the kinds of entities a caller will want to refer to in a conversation in a given domain before the anaphoric reference and ellipsis resolution module can be designed. At present our module is partially domain independent in that the entities for the email domain are supplied as data. However, some references remain closely bound to specific constraints in the Spec List of the ELF at present. It is our aim to work towards a totally domain independent module for future systems.

In the sub-sections that follow (5.1 - 5.8) we explain how the various kinds of reference and ellipsis are resolved in MailSec.

### 5.1 Simple singular pronominals

The singular pronominals we call `simple' in our system are `it', and `that'. In the exchange below, the caller's second utterance contains an anaphoric reference: "it".

Caller:	List messages from Anna.
MailSec:	There is only one, you have one message from Anna Cordon.
Caller:	Read it.

The ELF for the caller's first utterance, `List messages from Anna', is:

*ELF:*  $list(A)$ ,

*Expect List:*  $[plural(A)]$

*Spec List:*  $[message(A),from(A,B),named(B,anna),gender(B,feminine)]$ .

The Spec List is used as the basis for the database query where it is instantiated with the full name [anna,cordon] as described in 3.5:

$[message(A),from(A,[anna,cordon]),gender([anna,cordon],feminine),named([anna,ordon],anna)]$

This, and the result of the database query, a single email, are placed on the History List.

The ELF for the caller's second utterance is:

*ELF:*  $read(A)$ ,

*Expect List:*  $[salient(A,it),singular(A)]$

*Spec List:*  $[ ]$ .

The reference resolution module finds the referent by searching back through the History List for preceding interactions to find one where the response contained just a single message.

We assume that the last single message mentioned will be the referent of "it", although this

might not always be the case as discussed above.

The module now replaces the empty Spec List with the Spec List from the History List to produce the Resolved Extended Logical Form (RELF):

*RELF: read(A),*

*Expect List: [salient(A,it),singular(A)]*

*Spec List: [message(A), from(A,[anna,cordon]), gender([anna,cordon], feminine),  
named([anna,cordon],anna)].*

This will be used as the basis for the new database query, which will locate the correct message.

## 5.2 Simple Plural pronominals

The plural pronominals we call 'simple' in our system are 'them', and 'those'. In the exchange below, the caller's second utterance contains an anaphoric reference: 'them'.

Caller: List messages from Peter.  
MailSec: You have 2 messages from Peter Wyard.  
Caller: Read them.

The ELF for the caller's first utterance, 'List messages from Peter', is:

*ELF: list(A),*

*Expect List: [plural(A)]*

*Spec List: [message(A),from(A,B),named(B,peter),gender(B,masculine)].*

The Spec List is instantiated by the database query stage as described in 3.5:

*[message(A),from(A,[peter,wyard]),gender([peter,wyard],masculine),named([pete,wyard],peter)  
]*

This, together with the emails the database query found, are placed on the History List:

The ELF for the caller's second utterance is:

*ELF: read(A),*

*Expect List: [salient(A,them),plural(A)]*

*Spec List: [].*

The reference resolution module finds the referent by searching back through the History List for preceding interactions to find one where the response contained more than one message.

We assume that the last set of messages, greater than one, mentioned will be the referent of "them", although this might not always be the case as discussed above.

The module now replaces the empty Spec List with the Spec List from the History List to produce the RELF:

*RELF: read(A),*

*Expect List: [salient(A,them),singular(A)]*

*Spec List: [message(A),from(A,[ peter,wyard]), gender([peter,wyard], masculine), named([peter,wyard],peter)].*

### 5.3 Definite ordinals

Suppose the following exchange takes place:

Caller: List Alison's emails.  
MailSec: You have 10 messages from Alison Simons.  
Caller: Read the first message.

MailSec needs to understand 'the first message'. Anaphors are represented by variables in the ELF, e.g. the ELF for 'Read the first message' is:

*ELF: read(A),*

*Expect List: [salient(A,the),singular(A)]*

*Spec List: [message(A),ord(A,1)]*

where the variable A is something to be read, the caller is expecting A to exist (salient(A,the)), and is expecting A to be singular. ord(A,1) specifies the first message in a set of messages.

The reference resolution module finds the referent by searching back through the logical forms for preceding utterances in the History List to find something which matches the missing information. In this case, MailSec would be looking for a Spec List associated with a database query result which is a set of messages and it finds the set associated with the query for 'List Alison's emails'.

We assume that the last set of messages mentioned will be the referent, although this might not always be the case as discussed above. The ELF output by the parsing and semantics module is thus converted into an RELF where the missing information: from(A,[alison,simons]), name([alison,simons],alison), gender([alison,simons],feminine), is merged with the existing information in the Spec List:

*RELF: read(A),*

*Expect List: [salient(A,the),singular(A)],*

*Spec List: [message(A),from(A, [alison,simons]), ord(A,1), name([alison,simons],alison), gender([alison,simons],feminine)]*

Note that the Spec List from the History List has been instantiated with the full name [alison,simons] as discussed in 3.5.

This merging process allows the original query to be specialised. The new query is the same as the query on the History List that retrieved all Alison's emails, but it now has the additional specifier ord(A,1), which will retrieve only the first of the original set.

### 5.4 Complex singular pronominals

An example of what we call 'complex singular pronominals' are noun phrases 'the one', and 'that one'. These are resolved by searching back through the History List for a Spec List which retrieved a single message from the email database. Again the History Spec List is merged with the current Spec List.

Note that sometimes a query can retrieve multiple sets of messages:

Caller: Are there any messages from Franklin?  
MailSec: Yes, you have 2 messages from Tony Franklin, and one message from David Franklin.  
Caller: Read the one from David.

When the first utterance, 'Are there any messages from Franklin', is processed. The database query finds two sets of emails: one set from Tony Franklin and one set from David Franklin. As described in 3.5, the result is two instantiated Spec Lists, one associated with a set of two messages (from Tony Franklin), and one associated with a single message from David Franklin:

*[message(A),from(A,[tony,franklin]),gender([tony,franklin],masculine),named([ony,franklin],franklin)]*

*[message(A),from(A,[david,franklin]),gender([david,franklin],masculine),named[david,franklin],franklin)]*

Both results will be recorded on the History List. The reference resolution process searches through all past Spec Lists and database query results on the History List. It is able to select the correct one in this case.

## 5.5 Complex plural pronominals

An example of what we call 'complex plural pronominals' are noun phrases 'the ones', and 'those ones'. These are resolved in a similar manner to complex singular pronominals above except that the search is for a set of emails rather than a single one.

## 5.6 Complex plural comparatives

The system does not resolve many comparatives in its current stage of development. We are experimenting with references like 'the other messages', as in:

MailSec: You have 3 messages from Anna Cordon.  
Caller: Read the first message from Anna.  
MailSec: Message from Anna Cordon, message reads ....  
Caller: Read the other messages from Anna.

This can be resolved by searching back for two sets of messages, one a subset of the other. The first set (the first message from Anna) is a subset of the second set (all messages from Anna). The set we need is the result of subtracting the smaller set from the larger.

## 5.7 Ellipsis resolution

An example of ellipsis resolution occurs in the following exchange:

Caller: Do I have any messages from Peter?  
MailSec: Yes, you have 2 messages from Peter Wyard.  
Caller: Do I have any from David?  
MailSec: No, you have no messages from David Franklin.

Here in the second utterance from the caller, 'messages' has been left out and the caller has simply said 'Do I have any from David?'. For this utterance the parsing/semantics module will produce the ELF:

*ELF: ynq(A),*

*Expect List: [],*

*Spec List: [from(A,B),gender(B,masculine),named(B,david)]*

with message(A) missing.

The Anaphoric reference and ellipsis module searches the Spec List to find out whether it contains any of the constraints it knows are entities in the domain (e.g. message(A), folder(A), etc.) as described above. If an entity is missing, it looks in the History List, finds the last-mentioned entity from the Spec List for the last utterance, and inserts it into the current Spec List:

*[message(A),from(A,B),gender(B,masculine),named(B,david)]*

Note that the variables are consistent so that the A in message(A) is the same as the A in from(A,B).

## 5.8 Senders and Recipients of emails

The caller can refer to a sender by 'his', 'him' or 'her', for example 'Read **his** message.', 'Read the message from **her**'. The anaphoric reference and ellipsis module checks for from(A,B) and named(B,C) in the Spec List, where C is a variable. It then searches back through previous logical forms to find one with named(B,C) where C is not a variable and the gender is the same as the current utterance.

The system always chooses the last named person of the same gender. We find that in the majority of cases this works.

Recipients are resolved in a similar way to senders. The system can resolve examples such as 'List my messages to **him**', and 'Forward the message from Peter to **her**'.

## 6. Conclusions

We have built a system which successfully resolves many kinds of anaphoric references. Our system covers examples of all three types of reference proposed by Halliday and Hasan [11]: personal references such as "Read **her** message.", "Are there any emails from **him**?"; demonstrative references such as "Read **that email** from David"; and comparatives such as: "Are there any **other messages** from John?". The system also successfully resolves noun ellipsis such as "How many from David do I have?"

All references in MailSec are resolved by finding the most recent matching item. Obviously the referent found may not always be the correct one. However, MailSec always states explicitly which referent it is using, so the caller can be in no doubt when it has got the referent wrong. The flexibility of the dialogue management in MailSec is such that the caller can always make a correction by stating more fully what she/he means.

Apart from increasing the coverage, we think the system could be made more domain independent. At present our module is partially domain independent in that the entities for the email domain are supplied as data. However, some references remain closely bound to specific constraints in the logical form. Future porting to other domains will give us better insight into how to achieve this aim.

## **Acknowledgements**

We are grateful to the members of the BT Labs NLP Group, and members of the BT Labs Spoken Language System development team for their help, encouragement and advice during this work. In particular we would like to thank David Franklin, Louise Helliker, Christopher Holt and Alison Simons. Special thanks to Peter Wyard, Edward Kaneen and Stephen Appleby for their work on the parsing and semantics module and to Keith Preston for his work on the database query module.

## **References**

1. Attwater, D.J. and Whittaker, S.J. 'Issues in large vocabulary interactive speech recognition', BT Technology Journal, Vol. 14, No. 1, January 1996.
2. Williams, S.H. 'Dialogue management in a mixed-initiative cooperative spoken language system', TWLT II, Twente, 1996.
3. Anderson, A.H. Bader, M. Bard, E.B. Boyle, E. Docherty, G. Garrod, S. Isard, S. Kowtko, J. McAllister, J. Miller, J. Sotillo, C. Thomson, H. and Weinert, R. 'The HCRC Map Task Corpus', HCRC/RP-29, Edinburgh University, 1992.
4. Moore, R. and Browning, S. 'Results of an exercise to collect 'genuine' spoken enquiries using woz techniques', in Proceedings of the Institute of Acoustics, Volume 14, No. 6, 1992.
5. Yankelovich, N. and Baatz, E. 'SpeechActs: a framework for building speech applications', AVIOS '94 Conference Proceedings, San Jose, CA, September 20-23, 1994. SMLI 94-0243.
6. Lewin, I. and Pulman, S.G. 'Inference in the resolution of ellipsis', ESCA Workshop on Spoken Dialogue Systems, Denmark, 1995.
7. Wyard, P.J. Simons, A.D. Appleby, S. Kaneen, E. Williams, S.H. and Preston, K.R. 'Spoken Language Systems - beyond prompt and response', BT Technology Journal, Vol. 14, No. 1, January 1996.
8. Scahill, F. Talintyre, J.E. Johnson, S.H. Bass, A.E. Lear, J.A. Franklin, D.J. and Lee, P.R. 'Speech Recognition - making it work', BT Technology Journal, Vol. 14, No. 1, January 1996..
9. Page, J.H. and Breen, A.P. 'The Laureate text-to-speech system - architecture and applications', BT Technology Journal, Vol. 14, No. 1, January 1996.
10. Kowtko, J.C. Isard, S.D. and Doherty, G.M. 'Conversational games within dialogue', Human Communication Research Centre, Edinburgh, 1993.
11. Halliday, M.A.K. and Hasan, R. 'Cohesion in English', Longman, Harlow, 1976.