

Querying Databases of Annotated Speech

Steve Cassidy

Department of Linguistics
Macquarie University
Sydney, NSW 2109,
Australia

steve.cassidy@mq.edu.au

Steven Bird

Linguistic Data Consortium,
University of Pennsylvania,
3615 Market St, Suite 200,
Philadelphia, PA 19104-2608, USA

steven.bird@ldc.upenn.edu

Abstract

Annotated speech corpora are databases consisting of signal data along with time-aligned symbolic ‘transcriptions’. Such databases are typically multidimensional, heterogeneous and dynamic. These properties present a number of tough challenges for representation and query. The temporal nature of the data adds an additional layer of complexity. This paper presents and harmonises two independent efforts to model annotated speech databases, one at Macquarie University and one at the University of Pennsylvania. Various query languages are described, along with illustrative applications to a variety of analytical problems. The research reported here forms a part of several ongoing projects to develop platform-independent open-source tools for creating, browsing, searching, querying and transforming linguistic databases, and to disseminate large linguistic databases over the internet.

1. Databases of Annotated Speech Recordings

Annotated corpora have been an essential component of research and development in language-related technologies for some years. Text corpora have been used for developing information retrieval and summarisation software (e.g. MUC [1], TREC [14]), automatic taggers and parsers and machine translation systems [8]. In a similar way, annotated speech corpora have proliferated and have found uses across a rapidly expanding set of languages, disciplines and technologies [www.ldc.upenn.edu/annotation/].

Over the last 7 years, the Linguistic Data Consortium (LDC) has published over 150 text and speech databases [www.ldc.upenn.edu/Catalog/].

Typically, such databases are specified at the level of file formats. Linguistic content is annotated with a variety of tags, attributes and values, with a specified syntax and semantics. Tools are developed for each new format and linguistic domain on an ad hoc basis. These systems are akin to the databases of the 1960s. There is a physical representation along with a hand-crafted program offering a single view on the data. Recently, the authors have shown how the three-level architecture and the relational model can be applied to annotated speech databases [4, 5]. The goal of this paper is to illustrate our two approaches and to describe ongoing research on query algebras.

Before presenting the models we give an example of a collection of speech annotations. This illustrates the diversity of the physical formats and gives an idea of the challenge involved in providing a general-purpose logical characterisation of the data. The Boston University Radio Speech Corpus consists of 7 hours of radio news stories [www.ldc.upenn.edu/Catalog/LDC96S36.html]. The annotations include four types of information: orthographic transcripts, broad phonetic transcripts (including main word stress), and two kinds of prosodic annotation, all time-aligned to the digital audio files. The two kinds of prosodic annotation implement the system known as ToBI – Tones and Break Indices [www.ling.ohio-state.edu/phonetics/E_ToBI/]. We have added three further annotations: coreference annotation and named entity annotation in the style of MUC-7 [www.muc.saic.com/proceedings/

muc_7_toc.html], and syntactic structures in the style of the Penn TreeBank [11]. Fragments of the physical data are shown in Figure 1.

Coreference annotation (Figure 1, top left) associates a unique identifier to each noun phrase and a reference attribute which links each pronoun to its antecedent. The set of coreferring expressions is considered to be an equivalence class. Named-entity annotation (top centre) identifies and classifies numerical and name expressions. Penn Treebank annotation provides a syntactic parse of each sentence. The word-level annotation (bottom left) gives the end time of each word (a second offset into the associated signal data). The syllable annotation gives the Arpabet phonetic symbols (see [www.ldc.upenn.edu/doc/timit/phoncode.doc]). The tonal annotation provides time points and intonational units, and the part of speech annotation (bottom right) specifies the syntactic category of each word. This is but a small sample of the bazaar of data formats.

2. Data Models for Speech Databases

Two database models for multi-layered speech annotations have been developed by the authors. The Emu model (Macquarie) organises the data primarily in terms of its hierarchical structure, while the annotation graph model (Penn) foregrounds the temporal structure. In separate work we demonstrate the expressive equivalence of the two models [4, 7]. Here we give a brief overview of both models. In the remainder of this paper we will consider mainly the annotation graph data model, while the Emu system serves as an example of a working speech database system.

2.1. The Emu model

The Emu speech database system [www.shlrc.mq.edu.au/emu] [6, 7] provides tools for creation, query and analysis of data from annotated speech databases. Emu is implemented as a core C++ library and a set of extensions to the Tcl scripting language which provide a set of basic operations on speech annotations. Emu provides a flexible annotation model into which a number of existing label file formats can be read.

The Emu annotation model is based on a set of *levels* which represent different types of linguistic data such as words, phonemes or pitch events.

Each level contains a set of *tokens* which have one or more *labels* and optionally a start and end time relative to an associated speech signal. Within a level, tokens are stored as a partial order representing their sequence in the annotation: each token may have zero or more previous and next tokens. The partial ordering must respect timing information if it is present in the tokens: that is, a token cannot follow a token with an later start time.

Within and between levels, tokens may be related by either *domination* or *association* relations. Domination relations relate a parent token to an ordered sequence of constituent child tokens and imply that the start and end times of the parent could be inferred from those of the children. Association relations have no in-built semantics and can be used for any application specific relation, such as that between a word and a tone target which denotes the point at which word stress is realised (Figure 2). Relations may be defined between any pair of levels which allows Emu to handle intersecting hierarchies such as that illustrated in Figure 2.

2.2. The annotation graph model

A second general purpose model supporting multiple independent hierarchical transcriptions of the same signal data is known as the *annotation graph* [3, 4]. This model forms the heart of a joint initiative between LDC, NIST [www.nist.gov] and MITRE [www.mitre.org] to develop an architecture and tools for linguistic analysis systems (ATLAS), and an NSF-sponsored project between LDC, the Penn database group, and the CMU Psychology and Informedia departments, to develop a multimodal database of communicative interaction called Talkbank [www.talkbank.org].

Annotation graphs are labelled DAGs with time references on some of the nodes. Bird and Liberman have demonstrated that annotation graphs are sufficiently expressive to encompass the full range of current speech annotation practice. A simple example of an annotation graph is shown in Figure 3, for a corpus known as TIMIT [10]. Annotation graphs (AGs) have the following structure. Let $L = \bigotimes L_i$ be the label data which occurs on the arcs of an AG. The nodes N of an AG reference signal data by virtue of a function mapping nodes to time offsets T . AGs are now defined as follows:

Coreference Annotation

```

<COREF ID="2" MIN="woman">
  This woman</COREF>
receives three hundred dollars
a month under
<COREF ID="5">
  General Relief</COREF>, plus
<COREF ID="16"
  MIN="four hundred dollars">
  four hundred dollars a month in
<COREF ID="17"
  MIN="benefits" REF="16">
  A.F.D.C. benefits</COREF>
</COREF> for
<COREF ID="9" MIN="son">
  son, who is a
<COREF ID="3" REF="2">
  her</COREF> son
</COREF>, who is
<COREF ID="10" MIN="citizen" REF="9">
  a U.S. citizen</COREF>.
<COREF ID="4" REF="2">
  She</COREF>'s among
<COREF ID="18" MIN="aliens">
  an estimated five hundred illegal
  aliens on
<COREF ID="6" REF="5">
  General Relief</COREF>
  out of
<COREF ID="11" MIN="population">
  the state</COREF>'s
  total illegal immigrant
  population of
<COREF ID="12" REF="11">
  one hundred thousand
</COREF>
</COREF>.
<COREF ID="7" REF="5">
  General Relief</COREF>
  is for needy families and
  unemployable adults who
  
```

Named Entity Annotation

```

This woman receives
<b_numex TYPE="MONEY">
  three hundred dollars
</b_numex>
a month under General
Relief, plus
<b_numex TYPE="MONEY">
  four hundred dollars
</b_numex>
a month in A.F.D.C.
benefits for her
son, who is a
<b_enamex TYPE="LOCATION">
  U.S.
</b_enamex>
citizen. brth She's among
an estimated five hundred
illegal aliens on General
Relief brth out of the
state's total illegal
immigrant population of
one hundred thousand. brth
General Relief is for
needy families and
unemployable adults brth
who don't qualify for other
public assistance. brth
Welfare Department
spokeswoman
Michael Reganburg
brth says the state will
save about
one million dollars
a year if illegal aliens
are denied General Relief.
  
```

Penn Treebank Annotation

```

((S
  (NP-SBJ This woman)
  (VP receives
    (NP
      (NP
        (NP (QP three hundred) dollars)
        (NP-ADV a month)
        (PP under
          (NP General Relief))) , plus
        (NP
          (NP (QP four hundred) dollars)
          )
        (NP-ADV a month)
        (PP in
          (NP A.F.D.C. benefits))))
      )
    (PP for
      (NP
        (NP her son) ,
        (SBAR (WHNP-1 who)
          (S (NP-SBJ *T*-1)
            (VP is
              (NP-PRD a U.S. citizen))))))
      )
    )
  )
))
((S
  (NP-SBJ She)
  (VP 's
    (PP-PRD among
      (NP (NP an estimated
        (QP five hundred) illegal aliens)
        (PP on
          (NP General Relief)))
      )
    (PP out of
      (NP
        (NP
          (NP the state 's)
          total illegal immigrant population)
          (PP of
            (NP
              (QP one hundred thousand))))))
      )
    )
  )
))
  
```

Word-Level Annotation

```

0.320000 This
0.620000 woman
1.120000 receives
1.370000 three
1.670000 hundred
2.020000 dollars
2.060000 a
2.450000 month
2.740000 under
3.280000 General
3.800000 Relief
4.310000 plus
4.520000 four
4.800000 hundred
5.160000 dollars
5.190000 a
5.480000 month
5.610000 in
6.340000 A.F.D.C.
6.870000 benefits
7.060000 for
7.190000 her
7.620000 son
7.830000 who
7.970000 is
8.020000 a
  
```

Syllable Annotation

```

H# 0 2
H# 2 3
>endsil
DH 5 14 4.182398
IH 19 6 -0.184139
S 25 8 -0.387113
>This
W 33 6 -0.495798
UH+1 39 3 -0.792806
M 42 7 0.042605
>
EN 49 14 0.395379
>woman
R 63 3 -0.996359
IY 66 7 -0.658371
>
S 73 12 0.865892
IY+1 85 13 0.815127
V 98 9 0.815878
Z 107 6 -0.563102
>receives
TH 113 9 0.506469
R 122 5 -0.359288
IY+1 127 11 0.323961
>three
HH 138 3 -0.905714
  
```

Tonal Annotation

```

0.373684 HiF0
0.493698 H*
0.915000 !H*
1.100000 !H-
1.325000 L+H*
1.389472 HiF0
1.716865 L*
2.178711 !H*
2.434735 L-L%
2.969376 H*
3.552627 HiF0
3.630000 H* ; !HL%, maybe LL% ?
3.770074 H-L%
4.440000 H*
4.478946 HiF0
5.330000 L*
5.445000 L-H%
5.709989 H*
6.300000 H*
6.331575 HiF0
6.740000 L-H%
7.336837 HiF0
7.402120 H*
7.607943 L-L%
8.301393 H*
8.510248 HiF0
10.105260 HiF0
  
```

Part-of-speech Annotation

```

This DT
woman NN
receives VBZ
three CD
hundred CD
dollars NNS
a DT
month NN
under IN
General NP
Relief, NP
plus CC
four CD
hundred CD
dollars NNS
a DT
month NN
in IN
A.F.D.C. NP
benefits NNS
for IN
her PP$
son, NP
who WP
is VBZ
  
```

Figure 1. Multiple Annotations of the Boston University Radio Speech Corpus

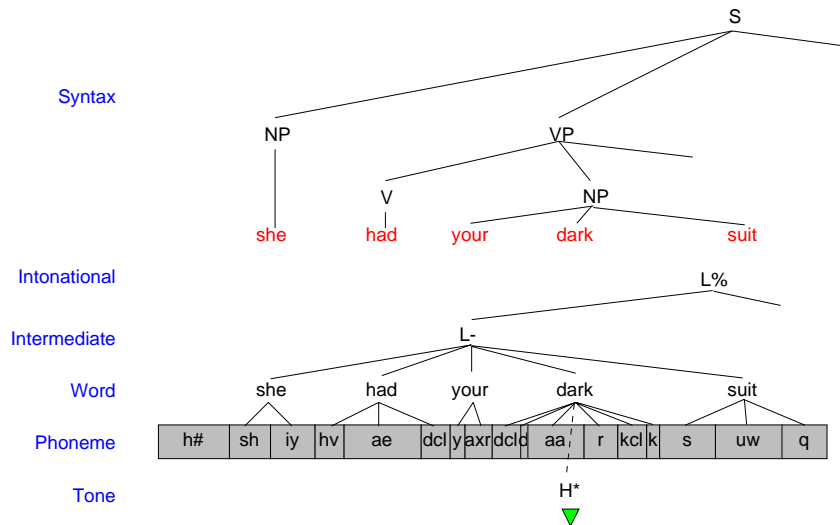


Figure 2. An example utterance from the TIMIT database which has been augmented with both a syntactic annotation and a ToBI style intonational annotation. The names of the levels are shown on the left, the Word level has been duplicated to show the links to both the syntactic and intonational hierarchies. The single Tone event H* is associated with the word ‘dark’. Time information at the phoneme level is used to derive times for all higher levels.

Definition 1 An *annotation graph* G over a label set L and a timeline T is a 3-tuple $\langle N, A, \tau \rangle$ consisting of a node set N , a collection of arcs A labelled with elements of L , and a time function τ , which satisfies the following conditions:

1. $\langle N, A \rangle$ is an acyclic digraph labeled with elements of L , and containing no nodes of degree zero;
2. $\tau : N \rightarrow T$, such that, for any path from node n_1 to n_2 in A , if $\tau(n_1)$ and $\tau(n_2)$ are defined, then $\tau(n_1) \leq \tau(n_2)$;

Note that AGs may be disconnected or empty, and that they must not have orphan nodes. The AG corresponding to the Emu annotation structure in Figure 2, for the first five words of a TIMIT annotation, is given in Figure 3. The arc types are interpreted as follows: S – syntax; W – word; P – phoneme; T – tone; Int – intermediate phrase; Int1 – intonational phrase.

3. Annotations as Relational Tables

Annotation data expressed in either the Emu or annotation graph data models can be trivially recast as a set of relational tables [5]. For the

purposes of this paper it is instructive to consider the relational form of annotation data in order to explore the requirements for a query language for these databases.

An annotation graph can be represented as a pair of tables, for the arc relation and time relations. The arc relation is a six-tuple containing an arc id, a source node id, a target node id, and three labels taken from the sets L_1, L_2, L_3 respectively. The choice of three label positions is somewhat arbitrary, but it seems to be both necessary and sufficient for the various annotation structures considered here.

We let L_1 be the set of types of transcript information (e.g. ‘word’, ‘syllable’, ‘phoneme’), and let L_2 be the substantive transcript element (e.g. particular words, phonetic symbols, and so on). We let L_3 be the names of equivalence classes, used here to model so-called ‘phonological association’. (This kind of association is discussed in depth in [2].) Let T be the set of non-negative integers, the sample numbers. Figure 4 gives an example for the TIMIT data of Figures 2, 3.

We form the transitive closure of the (unlabeled) graph relation to define a structural (graph-wise) precedence relation using a datalog program:

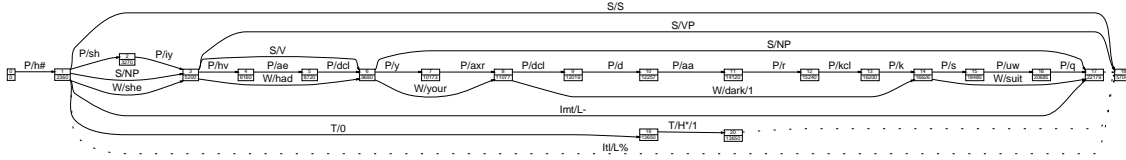


Figure 3. TIMIT Graph Structure

Arc	id	X	Y	L ₁	L ₂	L ₃	Arc	id	X	Y	L ₁	L ₂	L ₃	Time	N	T
	1	0	1	P	h#			18	1	3	W	she			0	0
	2	1	2	P	sh			19	3	6	W	had			1	2360
	3	2	3	P	iy			20	6	8	W	your			2	3270
	4	3	4	P	hv			21	8	14	W	dark	1		3	5200
	5	4	5	P	ae			22	14	17	W	suit			4	6160
	6	5	6	P	dcl			23	1	18	S	S			5	8720
	7	6	7	P	y			24	3	18	S	VP			6	9680
	8	7	8	P	axr			25	1	3	S	NP			7	10173
	9	8	9	P	dcl			26	3	6	S	V			8	11077
	10	9	10	P	d			27	6	17	S	NP			9	12019
	11	10	11	P	aa			28	1	17	Imt	L-			10	12257
	12	11	12	P	r			29	1	18	Id	L%			11	14120
	13	12	13	P	kcl			30	1	19	T	0			12	15240
	14	13	14	P	k			31	19	20	T	H*	1		13	16200
	15	14	15	P	s										14	16626
	16	15	16	P	uw										15	18480
	17	16	17	P	q										16	20685
															17	22179
															18	57040
															19	13650
															20	13650

Figure 4. The Arc and Time Relations

```

s_prec(X,X) :-
s_prec(X,Y) :- arc(_,X,Y,_,_,_)
s_prec(X,Y) :- s_prec(X,Z),
                arc(_,Z,Y,_,_,_)

```

Now we further define a temporal precedence relation, where leq is the \leq relation (minimally defined on the times used by the graph):

```

t_prec0(X,Y) :- time(X,T1),
                 time(Y,T2),
                 leq(X,Y)

t_prec(X,Y) :- t_prec0(X,Y)
t_prec(X,Y) :- t_prec(X,Z),
                 t_prec0(Z,Y)

```

4. Exploring Annotated Linguistic Databases

4.1. General architecture

In our experience with the analysis of linguistic databases, we have found a recurrent pattern of use having three components which we will call query, report generation, and analysis.

The query system proper can be viewed as a function from annotation graphs to sets of sub-graphs, i.e. those meeting some (perhaps complex) condition. The report generation phase is able to access these query results, but also the signals underlying the annotations. For example, the report generation phase can calculate such things as ‘mean F_2 in signal S during time interval (t_1, t_2) .’ Each hit constitutes an ‘observation’ in the statistician’s sense, and we extract a vector of specified values for each observation, to be passed along to the analysis system. The analysis phase is then some general-purpose data crunching system such as Splus or Matlab.

This architecture saves us from having to incorporate all possible calculations over annotated signals into the query language. The report generation phase can perform such calculations, as well as compute properties of the annotation data itself. This seems to simplify the query system a good deal; now things like ‘count the number of syllables to the end of the current phrase’ (which we do need to be able to do) are tasks for the report generator, not the query system proper.

In general, the result of a query is a set of sub-graphs, each of which forms one matching instance. If we use the relational model proposed above, these would be returned as a result table having the same structure as the arc relation of Figure 4, but containing just the tuples which took part in each matching instance. We are then faced with the problem of how to differentiate the matching instances, for example, if we wished to collect together the word labels for the query ‘find all words dominated by noun phrases’ we need some way of treating each sub-graph separately. Hence, we would prefer the result to be a set of tables rather than a single table containing all matching tuples.

In a sense, then, the only role of the query is to define an iterator for the report generator over a set of sub-graphs of the overall annotation graph.

The Emu query language

The Emu query language uses simple conditions on token labels which match only tokens at a specified level, for example: `Phonetic=A|I|O|U|E|V`. These conditions can be combined by sequence, domination or association operators to constrain the relational structure of the tokens of interest. Examples of each are:

Find a sequence of `vowel` followed by `stop` at the phoneme level:

```
[Phoneme=vowel -> Phoneme=stop]
```

Find Words not labelled `x` dominating `vowel` phonemes:

```
[Word!=x ^ Phoneme=vowel]
```

Find words associated with `H*` tones:

```
[Word!=x => Tone=H*]
```

The `Word!=x` query is intended to match any word in lieu of a query language construct which allows matching any label string.

Each query matches either a token or, in the case of the sequence query, a sequence of tokens. The result of a domination or association query is the result of the left hand side of the bracketed term; this can be changed by marking the right hand side term with a hash (`#`). Compound queries can be arbitrarily nested to specify complex constraints on tokens. As an example the following query finds sequences of stop and vowels dominated by strong syllables where the vowel is associated with an `H*` tone target, the result is a list of the vowel labels with associated start and end times.

```
[Syllable=S ^
```

```
[Phoneme=stop ->
  [Phoneme=vowel => Tone=H*]]]
```

The result of an Emu query is a table with one entry per matching token:

```
database:timit
query:Phoneme!=x
type:segment
#
h#      0      147.5   fjsp0:sa1
sh      147.5   232.5   fjsp0:sa1
iy      232.5   325     fjsp0:sa1
hv      325     385     fjsp0:sa1
...
```

This table is used to extract any of the associated time-series data associated with the database, an operation usually carried out from an analysis environment such as `Splus` or `XlispStat`. Emu provides libraries of analysis functions for these environments which facilitate, for example, mapping signal processing operations over each token in a query result or overlaying plots of the time series data for each token.

Although this query system has proved useful and useable in the environment of acoustic phonetics research, it is now evident that there are a number of shortcomings which prevent it’s wider use. The query syntax is unable to express some queries, such as those involving disjunction or optional elements, and the query result is only really useful for data extraction. It is for these reasons that we are now looking more formally at the requirements for a query language for annotation data.

4.2. A query language on annotation graphs

A high-level query language for annotation graphs, founded on an interval-based tense logic, is currently being developed and will be reported in a later version of this paper.

Here we describe a variety of useful queries on annotation graphs and formulate them as datalog programs. As we shall see, it turns out that datalog is insufficiently expressive for the range of queries we have in mind. Finding a more expressive yet tractable query language is the focus of ongoing research.

A number of simple operations, extending our two relations `arc/6` and `time/2`, will be necessary for succinct queries. The first and most obvious is for hierarchy. Observe in Figure 3 that there is a notion of structural inclusion defined by the arcs. We formulate this as follows:

```
s_incl(I,J) :-
    arc(I,W,Z,_,_,_),
    arc(J,X,Y,_,_,_),
    s_prec(W,X), s_prec(Y,Z)
```

Now, since `s_prec` is reflexive, so is `s_incl`. Observe that nodes 3 and 6 in Figure 3 are connected by both an `S/V` arc and a `W/had` arc. The syntactic verb arc `S/V` should dominate the word arc `W/had`, but not vice versa. Therefore we need to have a hierarchy defined over the types. We achieve this with a (domain-specific) ordering on the type names:

```
type_hierarchy(word,syl)
type_hierarchy(syl,seg)
```

Now dominance is expressed by the predicate:

```
dom(I,J) :-
    arc(I,_,_,L1,_,_),
    arc(J,_,_,L2,_,_),
    type_hierarchy(L1,L2),
    s_incl(I,J)
```

In some cases it is necessary to have an intransitive dominance relation that is sensitive to phrase structure rules. For simplicity of presentation, we assume binary branching structures. The first of the rules below states that a sentence arc `s` will immediately and exhaustively dominate an `np` arc followed by a `vp` arc.

```
ps_rule(s,np,vp)
ps_rule(np,det,n)
ps_rule(vp,v,np)
```

Now we define immediate dominance over the syntax arcs `syn` as follows:

```
i_dom(I,J) :-
    arc(I,X,Z,syn,P,_),
    ps_rule(P,C1,C2),
    arc(J,X,Y,syn,C1,_),
    arc(_,Y,Z,syn,C2,_)
```

```
i_dom(I,J) :-
    arc(I,X,Z,syn,P,_),
    ps_rule(P,C1,C2),
    arc(_,X,Y,syn,C1,_),
    arc(J,Y,Z,syn,C2,_)
```

Another widely used relation between arcs is association. In the instance of the AG model in Figure 4, association amounts to sharing the value of `L3`, as we saw in the tuples for `dark` and `H*` in Figure 4. The `assoc` predicate simply does a join on the third label field:

```
assoc(I,J) :-
    arc(I,_,_,_,_,A),
    arc(J,_,_,_,_,A)
```

Finally, it is convenient to have a kleene star relation. Unfortunately in `datalog` we are unable to collect up the arbitrary length sequence it matches. Here we have it returning the two nodes which bound the sequence, which is often enough to uniquely identify the sequence in practice.

```
node(N) :- arc(_,N,_,_,_,_)
node(N) :- arc(_,_,N,_,_,_)
```

```
kleene1(X,X,_):- node(X)
kleene1(X,Y,L):-
    arc(_,X,Z,L,_,_),
    kleene1(Z,Y,L)
```

```
kleene2(X,X,_):- node(X)
kleene2(X,Y,L):-
    arc(_,X,Z,_,L,_),
    kleene2(Z,Y,L)
```

```
kleene3(X,X,_):- node(X)
kleene3(X,Y,L):-
    arc(_,X,Z,_,_,L),
    kleene3(Z,Y,L)
```

With this simple machinery we can start defining some annotation queries. Find a sequence of vowel followed by stop at the phoneme level (assumes suitably defined `vowel` and `stop` unary relations):

```
vowel_stop(I,J) :-
    arc(I,_,Y,phoneme,V,_),
    arc(J,Y,_,phoneme,S,_),
    vowel(V), stop(S)
```

If we do not want both the vowel and the stop, but just the vowel, we could write:

```
vowel_stop(I) :-
    arc(I,_,Y,phoneme,V,_),
    arc(_,Y,_,phoneme,S,_),
    vowel(V), stop(S)
```

Find words dominating vowel phonemes:

```
strongWrdDomVowels(I) :-
    arc(I,_,_,word,s,_),
    arc(J,_,_,phoneme,V,_),
    vowel(V),
    dom(I,J)
```

Find words associated with `H*` tones:

```
sylHtone(I) :-
    arc(I,_,_,word,_,A),
    arc(_,_,_,tone,h*,A)
```

Find stop-vowel sequences dominated by words in noun phrases where the word is associated with an H* tone target.

```
stop_vowel_seq(I,J) :-
    arc(I,_,Y,phoneme,S,_), stop(S),
    arc(J,Y,_,phoneme,V,_), vowel(V),
    arc(W,_,_,word,_,_),
    arc(N,_,_,syn,np,_),
    dom(N,W), dom(W,I), dom(W,J),
    arc(T,_,tone,h*,_), assoc(W,T)
```

Find the intermediate phrase containing the main verb of a sentence:

```
imt_phrase(P) :-
    arc(K,_,_,syn,s,_),
    arc(J,_,_,syn,vp,_),
    arc(I,_,_,syn,v,_),
    i_dom(K,J), i_dom(J,I),
    dom(P,I),
    arc(P,_,_,imt,_,_)
```

Return the set of syllables between an H* and an L% tone (inclusive).

```
syls(K) :-
    arc(_,_,N,tone,h*,A1),
    arc(_,N,_,tone,l%,A2),
    arc(I,_,N1,syl,_,A1),
    arc(J,N2,_,syl,_,A2),
    kleene1(N1,N2,syl),
    arc(K,N2,N3,syl,_,_),
    kleene1(N3,N4,syl)
```

The above query shows how the datalog model breaks down. We would like it to return sets of sets of syllable arcs. Instead it returns a flat set structure. In many cases we will know that some arc participating in the query expression can be used to recover the nested structure. For example, if the head of the above clause was changed from `syls(K)` to `syls(I,K)`, then I will aggregate K in just the right way.

5. Applying XML Query Languages to Annotations

It is worth briefly considering the suitability of existing XML query languages such as XML-QL [9] and XQL [12] for the domain of annotated speech. At first glance the problems we face querying annotated speech data are similar to those present with XML queries in that both present a hierarchical data model. A number of formulations of annotation data as XML are possible, indeed some projects make use of XML/SGML based formats entirely (e.g.

MATE [mate.nis.sdu.dk/tpq], LACITO [lacito.vjf.cnrs.fr/ARCHIVAG/ENGLISH.htm]). XML can represent trees using properly nested tags, in the obvious way. In order to represent multiple independent hierarchies built on top of the same material one must construct trees using IDREF pointers. This idea was proposed by the Text Encoding Initiative [13] and recently adopted by the MATE project. We believe this approach is vastly more expressive than necessary for representing speech annotations, and we prefer a more constrained approach having desirable computational properties with respect to creation, validation and query.

The XQL proposal [12] describes a query language which is intended to select elements from within XML documents according to various criteria; for example, the query `text/author` returns all author elements that are children of text elements. The XQL data model ignores the order of elements within a parent element and has no obvious way to query for sequences of tokens.

The XML-QL proposal [9] provides for a data model where the order of elements is respected. A query for a word-internal vowel-stop sequence could be expressed as follows (assuming suitably tagged annotation data for TIMIT):

```
<word>
  <phoneme label=&vowel;/>
  <phoneme label=&stop;/>
</word>
```

The result of this query would have the following form:

```
<word label=had>
  <phoneme label=ae/>
  <phoneme label=dcl/>
</word>
<word label=dark>
  <phoneme label=ar/>
  <phoneme label=k/>
</word>
```

Queries which refer to two independent hierarchies, such as syntactic and intonational phrase structure, need to use joins. For example, to find words that are simultaneously at the end of both clauses and intermediate phrases, we could have the following query:

```
<intermediate>
  <word id=$i></>[end()]
</intermediate>

<clause>
```

```
<word id=$i></>[end()]\n</clause>
```

We assume the existence of some mechanism to pick out the last child element. The ID attribute ensures that the words are the same in each part of the join.

Perhaps either of these approaches could be made to work for a useful range of query needs. However they do not appear to be sufficiently general. For example, it is often useful to have query expressions involving Kleene star: 'select all pairs of consonants, ignoring any intervening vowels' (CV*C). Such queries may ignore hierarchical structure, finding sequences across (say) word boundaries. Using regular expressions over paths, XML-QL could provide access to strings of terminal symbols ignoring intervening levels of hierarchy. Yet it does not provide regular-expression matching over those sequences. Alternatively, sequences at each level of a hierarchy could be chained together using IDREF pointers, but it is unclear how we would manage closures over such pointer structures.

6. Conclusions

Annotated speech corpora are an essential component of speech research, and the variety of formats in which they are distributed has become a barrier to their wider adoption. To address this issue, we have developed two data models for speech annotations which seem to be sufficiently expressive to encompass the full range of practice in this area. We have shown how the models can be stored in a simple relational format, and how many useful queries in this domain are first-order. However, existing query languages lack sufficient expressive power for the full range of queries we would like to be able to express, and we hope stimulate new research into the design of general purpose query languages for databases of annotated speech recordings.

Acknowledgements

We are grateful to Peter Buneman, Mark Liberman and Gary Simons for helpful discussions concerning the research reported here.

References

- [1] *Message Understanding Conference Proceedings (MUC-7)*. Science Applications International

- Corporation, 1998. [www.muc.saic.com/proceedings/muc_7_toc.html].
- [2] S. Bird. *Computational Phonology: A Constraint-Based Approach*. Studies in Natural Language Processing. Cambridge University Press, 1995.
- [3] S. Bird and M. Liberman. Annotation graphs as a framework for multidimensional linguistic data analysis. In *Towards Standards and Tools for Discourse Tagging – Proceedings of the Workshop*, pages 1–10. Somerset, NJ: Association for Computational Linguistics, 1999. [xxx.lanl.gov/abs/cs.CL/9907003].
- [4] S. Bird and M. Liberman. A formal framework for linguistic annotation. Technical Report MS-CIS-99-01, Department of Computer and Information Science, University of Pennsylvania, 1999. [xxx.lanl.gov/abs/cs.CL/9903003].
- [5] S. Cassidy. Compiling multi-tiered speech databases into the relational model: experiments with the Emu system. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, 1999. [www.shlrc.mq.edu.au/emu/eurospeech99.shtml].
- [6] S. Cassidy and J. Harrington. Emu: An enhanced hierarchical speech data management system. In *Proceedings of the Sixth Australian International Conference on Speech Science and Technology*, 1996. [www.shlrc.mq.edu.au/emu/].
- [7] S. Cassidy and J. Harrington. Multi-level annotation of speech: an overview of the emu speech database management system. manuscript, 1999.
- [8] K. W. Church and R. L. Mercer, editors. *Special Issue on Computational Linguistics Using Large Corpora*, volume 19(1,2). MIT Press, 1993.
- [9] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for XML, 1998. [www.w3.org/TR/NOTE-xml-ql/].
- [10] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM*. NIST, 1986. [www ldc.upenn.edu/lol/docs/TIMIT.html].
- [11] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–30, 1993. www.cis.upenn.edu/~treebank/home.html.
- [12] J. Robie, J. Lapp, and D. Schach. XML query language (XQL), 1998. [www.w3.org/TandS/QL/QL98/pp/xql.html].
- [13] Text Encoding Initiative. *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*. Oxford University Computing Services, 1994. [www.uic.edu/orgs/tei/].
- [14] E. M. Voorhees and D. K. Harman, editors. *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*. NIST, Government Printing Office, 1998. [trec.nist.gov/pubs/trec7/t7_proceedings.html].