TOWARDS A FORMAL FRAMEWORK FOR LINGUISTIC ANNOTATIONS

Steven Bird *Mark Liberman*

Linguistic Data Consortium, University of Pennsylvania,
3615 Market St, Philadelphia PA 19104-2608, USA
{sb,myl}@ldc.upenn.edu

Version presented at ICSLP, Sydney, December 1998

ABSTRACT

'Linguistic annotation' is a term covering any transcription, translation or annotation of textual data or recorded linguistic signals. While there are several ongoing efforts to provide formats and tools for such annotations and to publish annotated linguistic databases, the lack of widely accepted standards is becoming a critical problem. Proposed standards, to the extent they exist, have focussed on file formats. This paper focuses instead on the logical structure of linguistic annotations. We survey a wide variety of annotation formats and demonstrate a common conceptual core. This provides the foundation for an algebraic framework which encompasses the representation, archiving and query of linguistic annotations, while remaining consistent with many alternative file formats.

1. INTRODUCTION

'Linguistic annotation' is a cover term for any orthographic, phonetic or prosodic transcription; any speech, part-of-speech, disfluency or gestural annotation; and any free or word-level translation. Linguistic annotations may describe texts or recorded signals; our focus will be on the latter, broadly construed to include any kind of audio, video or physiological signal, or any combination thereof.

To date there have been several independent efforts to provide tools for annotating linguistic signals, to provide general formats for annotations, and to provide tools for searching databases of annotations. Additionally, hundreds of annotated linguistic databases have been published, where each database typically contains several different tiers of annotation. While the utility of such tools, formats and databases is unquestionable, the lack of standards is becoming a critical problem. The cause of the problem is understandable - databases are typically created with a particular need in mind, and with formats and tools tailored to that need. They are subsequently used for a wide variety of unforeseen purposes. Adapting existing software for creation, update, indexing, search and display of such databases typically requires extensive re-engineering. Attempts to standardise practice in this area have focussed on file formats (e.g. [11]). We contend that file formats, though important, are secondary.

In this presentation we report on a study of the logical structure of linguistic annotations. We demonstrate that, while the different annotation formats vary greatly in their form, their logical structure is remarkably consistent. In order to help us think about the form and meaning of annotations, we describe a simple mathematical framework endowed with a practically useful formal structure. This opens up an interesting range of new possibilities for creation, maintenance and search. We claim that essentially all existing annotations can be expressed in the framework.

2. DESIDERATA FOR A LINGUISTIC ANNOTATION FRAMEWORK

We will focus on three evaluation criteria for a linguistic annotation framework:

Generality
: The framework should be sufficiently expressive to encompass all commonly used kinds of linguistic annotation, including sensible variants and extensions. It should be capable of managing a variety of (partial) information about labels, temporal information and hierarchical structure.

Searchability
: There should be an efficient algebraic query formalism, whereby complex queries are composed out of well-defined combinations of simple queries, and where the result of querying a set of annotations is just another set of annotations. Annotations, however incomplete, should still be searchable. There should be an efficient indexing scheme providing near constant-time access into arbitrarily large annotation databases. The framework should also support the projection of natural sub-parts of annotations. For example, we may wish to project out just the prosodic content of annotations, or just the orthographic content, for display purposes.

Maintainability
: Annotation databases should be durable, remaining coherent and usable in the presence of corrections or the addition of new layers of annotation. Queries on prior versions should remain valid, and references into superseded annotations should persist whenever possible. Layers and versions of annotations should be modular so that revisions to one part do not entail global modification. For example, changing the spelling of a word should not entail changes to an annotation of phrase-level discourse function which covers the same text.

In addition to these desiderata, in the longer term we shall be concerned to provide realisations of annotations and queries in the finite-state realm, in the graphical domain, and as XML markup.

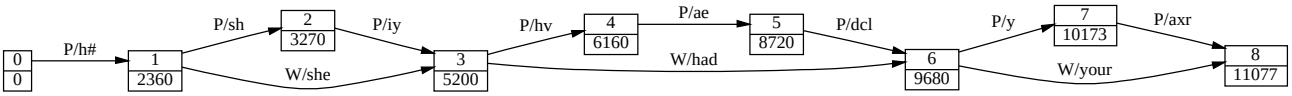

Figure 1: Graph Structure for TIMIT Example

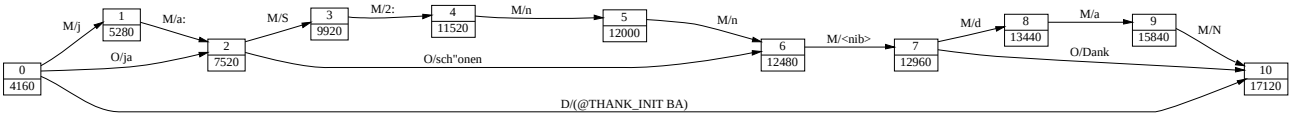

Figure 2: Graph Structure for Partitur Example

3. EXISTING ANNOTATION SYSTEMS

This section surveys a selection of examples drawn from existing annotations systems. We hope to demonstrate the existence of a common conceptual core of linguistic annotations.

3.1. TIMIT

The TIMIT corpus of read speech was designed to provide data for the acquisition of acoustic-phonetic knowledge and to support the development and evaluation of automatic speech recognition systems. Here, we just give one example taken from the TIMIT database [5]. The file `timit/train/dr1/fjsp0/sa1.wrd` contains:

```
2360 5200 she
5200 9680 had
9680 11077 your
11077 16626 dark
16626 22179 suit
22179 24400 in
24400 30161 greasy
30161 36150 wash
36720 41839 water
41839 44680 all
44680 49066 year
```

This file combines an ordinary string of orthographic words with information about the starting and ending time of each word (measured in audio samples at a sampling rate of 16 kHz). The path name `timit/train/dr1/fjsp0/sa1.wrd` tells us that this is training data, from 'dialect region 1', from female speaker 'jsp0', and that it contains words and audio sample numbers. The file `timit/train/dr1/fjsp0/sa1.phn` contains a corresponding broad phonetic transcription, which begins as follows:

```
0 2360 h#
2360 3720 sh
3720 5200 iy
5200 6160 hv
6160 8720 ae
8720 9680 dcl
9680 10173 y
10173 11077 axr
11077 12019 dcl
12019 12257 d
```

We can interpret each line `<time1> <time2> <label>` as an edge in a directed acyclic graph, where the two times are attributes of nodes and the label is a property of an edge which connects those nodes. The resulting 'annotation graph' for the above fragment is shown in Figure 1.

Observe that edge labels have the form `<type>/<content>` where the `<type>` here tells us what kind of label it is. We have used `P` for the (phonetic transcription) contents of the .phn file, and `W` for the (orthographic word) contents of the .wrd file. The top number for each node is an arbitrary node identifier, while the bottom number is the time reference. We distinguish node identifiers from time references since nodes may lack time references, as we shall see later.

3.2. Partitur

The Partitur format of the Bavarian Archive for Speech Signals [12] is similar to the TIMIT format, extended and reconceptualised to encompass a wide range of types of annotation. Partitur permits time-aligned, multi-tier description of speech signals, along with links between units on different tiers which are independent of the temporal structure.

For ease of presentation, the example Partitur file will be broken into a number of chunks, and certain details (such as the header) will be ignored. The fragment under discussion is from one of the Verbmobil corpora at BAS [www.phonetik.uni-muenchen.de/Bas]. The KAN tier provides the canonical transcription, and introduces a numerical identifier for each word to serve as an anchor for all other material.

```
KAN: 0 j'a:
KAN: 1 S'2:n@n
KAN: 2 d'aNk
KAN: 3 das+
KAN: 4 vE:r@+
KAN: 5 z'e:6
KAN: 6 n'Et
```

Tiers for orthographic and transliteration information then reference these anchors:

```
ORT: 0 ja
ORT: 1 sch"onen
ORT: 2 Dank
ORT: 3 das
ORT: 4 w"are
```

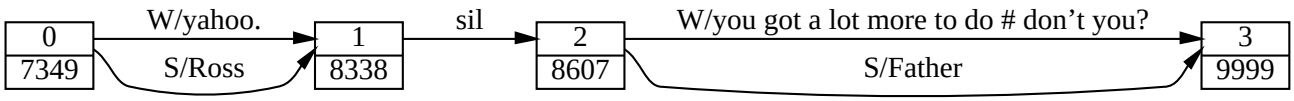

Figure 3: Graph Structure for CHILDES Example (Version 1)

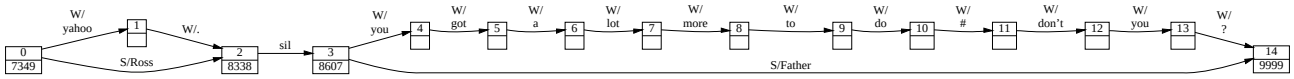

Figure 4: Graph Structure for CHILDES Example (Version 2)

```
ORT: 5 sehr
ORT: 6 nett

TRL: 0 <A>
TRL: 0 ja ,
TRL: 1 sch"onen
TRL: 1 <:<#Klopfen>
TRL: 2 Dank:> ,
TRL: 3 das
TRL: 4 w"ar'
TRL: 5 sehr
TRL: 6 nett .
```

Higher level structure representing, say, dialogue acts, can refer to contiguous sequences of anchors thus:

```
DAS: 0,1,2 @(THANK_INIT BA)
DAS: 3,4,5,6 @(FEEDBACK_ACKNOWLEDGEMENT BA)
```

Speech data can be referenced using annotation lines containing offset and duration information. As before, links to the KAN anchors are also specified (as the second-last field).

```
MAU: 4160 1119 0 j
MAU: 5280 2239 0 a:
MAU: 7520 2399 1 S
MAU: 9920 1599 1 2:
MAU: 11520 479 1 n
MAU: 12000 479 1 n
MAU: 12480 479 -1 <nib>
MAU: 12960 479 2 d
MAU: 13440 2399 2 a
MAU: 15840 1279 2 N
MAU: 17120 639 3 d
MAU: 17760 1119 3 a
MAU: 18880 1279 3 s
MAU: 20160 959 4 v
MAU: 21120 639 4 E:
MAU: 21760 1119 4 6
MAU: 22880 1119 5 z
MAU: 24000 799 5 e:
MAU: 24800 1119 5 6
MAU: 25920 1279 6 n
MAU: 27200 1919 6 E
MAU: 29120 2879 6 t
MAU: 32000 2559 -1 <p:>
```

The content of the first few words of the ORT (orthography), DAS (dialog act) and MAU (phonetic segment) tiers can apparently be expressed as in Figure 2. Note that we abbreviate the types, using O/ for ORT, D/ for DAS, and M/ for MAU.

3.3. CHILDES

The CHILDES database includes a vast amount of transcript data collected from children and adults who are learning languages [8]. All of the data are transcribed in the 'CHAT' format; a typical instance is provided by this opening fragment of a CHAT transcription:

```
@Begin
@Filename:      boys73.cha
@Participants:  ROS Ross Child, MAR Mark Child,
                FAT Brian Father, MOT Mary Mother
@Date:  4-APR-1984
@Age of ROS:    6;3.11
@Sex of ROS:    Male
@Birth of ROS:  25-DEC-1977
@Age of MAR:    4;4.15
@Birth of MAR:  19-NOV-1979
@Sex of MAR:    male
@Situation:     Room cleaning
*ROS:   yahoo.
%snd:   "boys73a.aiff" 7349 8338
*FAT:   you got a lot more to do # don't you?
%snd:   "boys73a.aiff" 8607 9999
*MAR:   yeah.
%snd:   "boys73a.aiff" 10482 10839
*MAR:   because I'm not ready to go to
        <the bathroom> [>] +/.
%snd:   "boys73a.aiff" 11621 13784
```

The `%snd` lines, by the conventions of this notation, provide times for the previous transcription lines, in milliseconds relative to the beginning of the referenced file. The first two lines of this transcript might then be represented graphically as in Figure 3. Observe that the silent period between nodes 1 and 2 is represented using a special label `sil`, rather than having no label at all; the reason for this choice will become clear later. Note also that the `%snd` annotations in the original chat file included a file name as well as a time. We could easily include a reference to such information in the node attribute.

The representation in Figure 3 is inadequate, for it treats entire phrases as atomic arc labels, causing problems for word-level indexing of annotations. We favour the representation in Figure 4, where labels have the same ontological status independently of

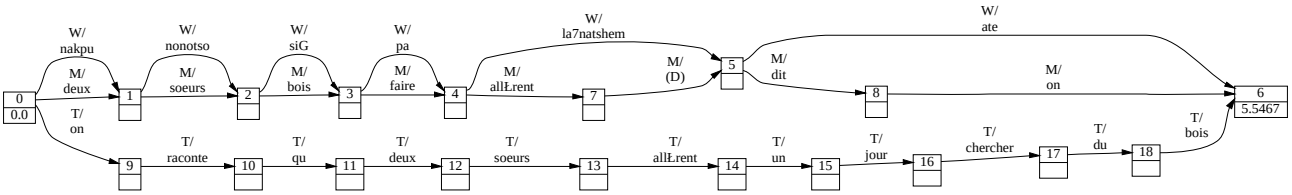

Figure 5: Graph Structure for Projet Archivage Example

the presence vs. absence of time references. Observe that most of the nodes in Figure 4 *could* have been given time references in the 'chat' format but were not. However, some of the tokens of the transcript (namely the punctuation marks) are not conceptually references to discrete stretches of time in the same way that orthographic words are. This illustrates the point that it is not always meaningful to assign time references to the nodes of an annotation. We will see a more pervasive example of this in the next section.

3.4. Projet Archivage

For another example of the same type, consider the following fragment of a Hayu story from Projet Archivage [ref], indented to show the structure more clearly:

```
<?XML version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ARCHIVE SYSTEM "Archive.dtd">

<ARCHIVE>
<HEADER>
  <TITLE>Deux soeurs</TITLE>
  <SOUNDFILE href="hayu.wav"/>
</HEADER>
<TEXT lang="hayu">
  <S id="s1">
    <TRANSCR>
      <W>nakpu</W>
      <W>nonotso</W>
      <W>si&#x014b;</W>
      <W>pa</W>
      <W>la&#x0294;natshem</W>
      <W>are.</W>
    </TRANSCR>
    <AUDIO type="wav" start="0.0000" end="5.5467"/>
    <TRADUC>On raconte que deux soeurs
            all&egrave;rent un jour
            chercher du bois.</TRADUC>
    <MOTAMOT>
      <W>deux</W>
      <W>soeurs</W>
      <W>bois</W>
      <W>faire</W>
      <W>all&egrave;rent(D)</W>
      <W>dit.on.</W>
    </MOTAMOT>
  </S>
</TEXT>
</ARCHIVE>
```

A possible graphical representation of the annotation of the sentence, expressed as a labelled DAG of the type under discussion, is shown in Figure 5. Here we have three types of edge labels: `W` for the words of the Hayu story; `M` for a word-by-word interlinear translation into French; and `T` for a phrasal translation into French.

(We have taken a small liberty with the word-by-word annotation in the Lacito original, which is arranged so that the `<W>` (for 'word') tokens in the Hayu are in one-to-one correspondence with the `<W>` tokens in the French `<MOTAMOT>` interlinear version. In such cases, it's normal for individual morphemes in the source language to correspond to several morphemes in the target language. This happens twice in the sentence in question, and we have split the interlinear translations to reflect the natural tokenisation of the target language.)

In this example, the time references (which are in seconds) are again given only at the beginning and end of the phrase, as required by the Archivage format. Nevertheless, the individual Hayu words have temporal extent and one might want to indicate that in the annotation. However, observe that there is no meaningful way of assigning time references to word boundaries in the phrasal translation. So whether the time references happen to be unknown, as in the upper half of Figure 5, or are intrinsically unknowable, as in the lower half of Figure 5, we can treat the `W`, `M` and `T` word-level annotations on a par.

3.5. LDC Broadcast News Transcripts

The LDC broadcast news corpora contain some 200 hours of speech data with SGML annotation. Here is the beginning of a radio program transcription, from the Hub-4 corpus [www.ldc.upenn.edu/Catalog/LDC98T28.html].

```
<Background Type=Music Time=0.000 Level=High>
<Background Type=Music Time=4.233 Level=Low>
<Section S_time=4.233 E_time=59.989 Type=Filler>

<Segment S_time=4.233 E_time=13.981 Speaker="Tad_Bile"
Fidelity=Low Mode=Spontaneous>
it will certainly make some of these districts more
competitive than they have been
<Sync Time=8.015>
so there will be some districts which are republican
<Sync Time=11.040>
but all of a sudden they may be up for grabs
</Segment>

<Segment S_time=13.981 E_time=40.840 Speaker="Noah_Adams"
Fidelity=High Mode=Planned>
politicians get the maps out again
<Sync Time=15.882>
for friday june fourteenth this is n. p. r.'s all
things considered
<Sync Time=18.960>
<Background Type=Music Time=23.613 Level=Low>
<Sync Time=23.613>
in north carolina and other states officials are
```

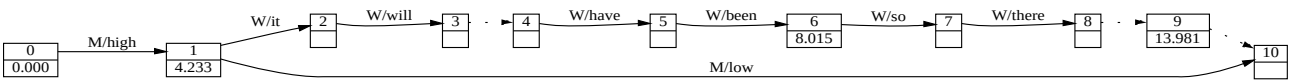

Figure 6: Graph Structure for LDC Broadcast Transcript Example

```
trying to figure out the effects of the supreme court
ruling against minority voting districts {breath}
<Sync Time=29.454>
a business week magazine report of a federal criminal
investigation {breath}
<Sync Time=33.067>
into the cause and the aftermath of the ValuJet crash
in florida {breath}
<Sync Time=36.825>
efforts in education reform {breath} and the question
will the public pay
</Segment>
```

Transcriptions are divided into `Section`s, where each consists of a number of `Segment`s. At various times during a `Segment` a `Sync Time` element is inserted to align a word boundary with an offset into a speech file. Elements specifying changes in background noise and signal quality function independently of the hierarchy. For example, a period of background music might bridge two segments, beginning in one segment and ending in the next. Figure 6 represents the structure of this annotation.

3.6. Emu

The Emu Speech Database System [4] permits hierarchical annotations arrayed over any number of levels, where each level is a linear ordering.

This defines the levels of the hierarchy and the immediate dominance relations. The hierarchy is a linear ordering.

```
level Utterance
level Intonational      Utterance
level Intermediate      Intonational
level Word              Intermediate
level Syllable          Word
level Phoneme           Syllable
level Phonetic          Phoneme    many-to-many
```

The final line licenses a many-to-many relationship between phonetic segments and phonemes, rather than the usual many-to-one relationship. According to the user's manual, this is only advisable at the bottom of the hierarchy, otherwise temporal ambiguities may arise.

At any given level of the hierarchy, the elements may have more than one attribute. For example, in the following declarations we see that elements at the `Word` level may be decorated with `Accent` and `Text` information.

```
label Word              Accent
label Word              Text
label Syllable          Pitch_Accent
```

The next line sets up a dependency between the `Phonetic` level of the hierarchy and an xwaves label file.

```
labfile Phonetic :format ESPS :type SEGMENT
  :mark END :extension lab :time-factor 1000
```

The `type` declaration distinguishes 'segments' with duration from 'events' which are instantaneous. Here, the time associated with a segment will mark its endpoint, as indicated by the `mark END` declaration. The timing information from the label file is adopted into the hierarchy (scaled from μs to ms), and can propagate upwards. In this way, the end of a phonetic segment may also become the end of a syllable, for example.

The sequence of labels from the xwaves label file is reproduced in the Emu annotation, while the timing information remains in the xwaves label file. Therefore the latter file is an essential part of the annotation structure, and that is why the Emu annotation must reference it explicitly. The labels are assigned unique numerical identifiers, as shown below for the sentence 'the price range is smaller than any of us expected'. For compactness, multiple lines have been collapsed to a single line.

```
Phonetic Phonetic
0 D     9 @     11 p    16 H    17 Or
19 r    20 ai   22 s    24 Or   30 r
31 ei   33 n    35 Z    37 I    44 zs
50 Om   52 m    53 o:   55 l    58 @
60 D    65 @    67 n    69 EC   76 E
77 n    80 i:   82 @    88 v    90 @
95 s    97 I    102 k   104 H   105 s
109 p   111 H   112 E   114 k   116 H
117 t   120 H   121 @   123 d   125 H
```

The labels on the more abstract, phonemic level are assigned a different set of identifiers.

```
Phoneme Phoneme
1 D     10 @    12 p    18 r    21 ai
23 s    25 r    32 ei   34 n    36 Z
38 I    45 z    46 s    51 m    54 o:
56 l    59 @    61 D    66 @    68 n
70 E    78 n    81 i:   83 @    89 v
91 @    96 s    98 I    103 k   106 s
110 p   113 E   115 k   118 t   122 @
124 d
```

Here is the remainder of the hierarchy.

```
Utterance Utterance
8

Intonational Intonational
7 L%

Intermediate Intermediate
5 L-      42 L-      74 L-

Word Word Accent Text
2 F W the             13 C S price
```

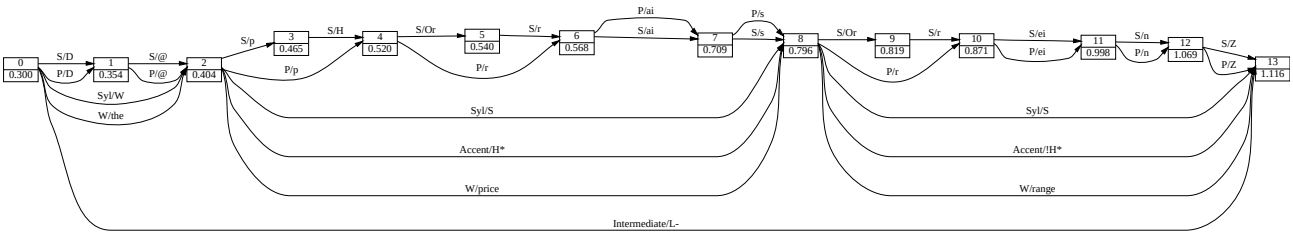

Figure 7: Graph Structure for Emu Annotation Example

```
26 C S range          39 F W is
47 C S smaller        62 F W than
71 F S any            84 F W of
92 F W us             99 C S expected

Syllable Syllable Pitch_Accent
4 W        15 S H*   28 S !H*   41 W
49 S H*    57 W      64 W       73 S
79 W H*    86 W      94 W       101 W
108 S H*   119 W
```

A separate section of an Emu annotation file lists each identifier, followed by all those identifiers which it dominates. For example, the line `4 0 1 9 10` states that the first `W` syllable (id=4) directly or indirectly dominates phonetic segments `D` (id=0) and `@` (id=9) and phonemes `D` (id=1) and `@` (id=10). The first intermediate phrase label `L-` (id=5) dominates this material and much other material besides:

```
5 0 1 2 4 9 10 11 12 13 15 16 17 18 19 20
  21 22 23 24 25 26 28 30 31 32 33 34 35 36
```

This exhaustive approach greatly facilitates the display of parts of the annotation hierarchy. If the syllable level is switched off, it is a trivial matter to draw lines directly from words to phonemes.

The first three words of this annotation can be represented as shown in Figure 7.

There are several other speech annotation formalisms we could have described, including the speech concordance facility of LDC Online [15], the NIST Universal Transcription Format [11], the Festival system [13], Altosaar's system [1], and the Delta language [7]. We intend to discuss these and others in an extended version of this paper.

4. ARCHITECTURAL CONSIDERATIONS

Considering our original desiderata and the systems surveyed above, we now describe some architectural issues which we believe should be addressed by any general purpose annotation model.

4.1. Representation of Partial Information

In the discussion of CHILDES and Projet Archivage above, there were cases where our graph representation had nodes which bore no time reference. Perhaps times were not measured, as in typical annotations of extended recordings where time references might only be given at major phrase boundaries (c.f. CHILDES). Or perhaps time measurements were not applicable in principle, as for phrasal translations (c.f. Projet Archivage). Various other possibilities suggest themselves. We might create a segment-level annotation automatically from a word-level annotation by looking up each word in a pronouncing dictionary and adding an arc for each segment, prior to hand-checking the segment annotations and adding time references to the newly created nodes. The annotation should remain well-formed (and therefore usable) at each step in this enrichment process.

Just as the temporal information may be partial, so might the label information. For example, we might label syllables with a `syl` type but with no actual name. A search for ternary feet could refer to these syllables without requiring an explicit name. Or a program might automatically detect and add segment boundaries without identifying the segments themselves. If syllable structure was present we could search for tri-segment syllables, again without needing to know the identity of the segments. This use of unlabelled arcs precludes their use to denote silence, hence the use of the explicit `sil` label in the CHILDES annotation structure above.

Beyond these two kinds of partiality there is an even more obvious kind of partiality we should recognise. An annotated corpus might be annotated in a fragmentary manner. Perhaps only 1% of the contents have any bearing on a particular research question. It should be possible to have a well-formed annotation structure with arbitrary amounts of annotation detail at certain interesting loci, and limited or no detail elsewhere. Naturally, one could always extract a sub-corpus and annotate that material completely, thereby removing the need for partiality, but this may have undesirable consequences for managing a corpus:

1. special intervention is required each time one wants to expand the sub-corpus as the research progresses;
2. it is difficult to make annotations of a sub-corpus available to someone working on a related research question with an overlapping sub-corpus, and updates cannot be propagated easily;
3. provenance issues arise, e.g. it may be difficult to identify the origin of any given fragment, in case access to broader context is necessary to retrieve the value of some other independent variable one might need to know;
4. it is difficult to combine the various contributions into the

larger task of annotating a standard corpus for use in perpetuity.

These problems with annotating derived corpora do not mean that all annotations of a corpus should be combined. On the contrary, even with one physical copy of a corpus, we could have several independent partial annotations, possibly owned by different people and possibly stored remotely from each other. Nor does this mean to say that the creation of sub-corpora is never warranted. The point is simply this: the annotation formalism should not force users to create a derived corpus just so that a partial annotation is well-formed.

4.2. Encoding Hierarchical Information

It is possible to identify at least three approaches to the encoding of hierarchical information.

Token-based hierarchy Here, hierarchical relations among annotations are specifically indicated in the annotation data with respect to tokens: 'this particular segment is a daughter of this particular syllable.' Formalisms that adopt this approach include Partitur, Emu and Festival.

Type-based hierarchy Here, hierarchical information is given only with respect to types – whether once and for all in the database, or ad hoc by a user, or both. This allows (for instance) the subordination of syllables to words to be indicated, but only as a general fact about all syllables and words, not as a specific fact about particular syllables and words.

Time-based hierarchy Here, annotations are akin to parse charts [6, 179ff], where each named period can be conceived of as a labelled arc connecting a start time to an end time. No hierarchical information is encoded at all, other than graphwise or timewise inclusion. Thus in a monosyllabic word, an onset consonant might be just as much the first sub-element of the word as it is of the syllable, and the fact that the syllable is subordinate to the word is not encoded. Examples of this approach are the LDC Broadcast News Transcripts, CHILDES and Delta [7].

The first approach is obviously convenient for programming, quite apart from any considerations of efficiency. For example, the text-to-speech systems developed by the second author at Bell Labs from 1977 onwards employed explicitly linked hierarchies, along with explicit links of other kinds, e.g. forward and backward pointers at every level of hierarchy, pointers from phrases to their initial and final elements at each level, and so on.

The addition of such redundant links could take place in the annotations themselves, when they are created, or in a subsequent compilation step, or as part of index creation. It could also take place when an annotation is loaded into a data-structure. Or it could be done at query time. Whether the addition of this predictable structure occurs at creation time, compile time, load time or query time is not an intrinsic part of the definition of general-purpose annotation structures. Accordingly, we distinguish the annotation formalism itself from enriched data structures having all manner of additional redundant links.

If these extra links are redundant, then so is the hierarchical structure itself, in many cases. Hierarchical structure may be read off the type structure and the temporal structure, so long as the following conditions hold:

1. labels exist in a type hierarchy;
2. the periods which are annotated are convex, having no holes;
3. the periods have specified endpoints so that a temporal inclusion relation exists; and

If a label p temporally includes a label q and the type of p immediately dominates the type of q in the type hierarchy, then we can infer that p immediately dominates q. This amounts to the second approach listed above – type-based hierarchy – the one we advocate here.

Three further arguments are sometimes raised in the areas of creation, query and display. We discuss them here and contend that they are tangential.

First, one can imagine cases where it is easier for annotators to specify non-terminals by using dominance relations rather than chart-like extents. Syntactic treebanking is a case in point [9]. However, this particular view of an annotation does not have to be ascribed first-class status in the annotation formalism itself. Moreover, what is convenient for creation might turn out to be inconvenient for query in some cases.

Second, in a database of annotated linguistic signals one could allow users to express queries making explicit reference to hierarchical structure. Perhaps the annotations need to be enriched in advance to permit efficient execution of such queries. Yet it may turn out to be more efficient to compile the query into a predicate over the original (impoverished) annotation structures. It is premature and probably impossible to commit to either tactic independently of a specific application. A similar point can be made about building indexes for annotated corpora; hierarchical relationships that are implicit in the annotations could be made explicit by the indexing scheme (see §6).

Third, we may want to restrict the display of annotations to certain levels of hierarchy. Some levels may be irrelevant for certain uses of a corpus, cluttering the display, or they may be at too fine a granularity, preventing the simultaneous viewing of items which are adjacent at a higher level. Some levels might participate in more than one hierarchy, such as syllables which simultaneously exist in an overarching metrical structure, while containing internal onset-rhyme structure. Accordingly, we might specify a view on annotations in terms of the kind of structure (e.g. metrical tree) rather than the particular levels the structure comprises. While the type structure underlying this layering of annotations is an important part of the formalism, the mechanism by which one turns on/off the display of (sets of) levels is a separate issue.

4.3. Instants

Even though a speech event might have duration, such as the attainment of a pitch target, the most perspicuous annotation may be tied to an instant rather than an interval. Some annotation formalisms (e.g. Emu, Festival, Partitur) provide a way to label instants. The alignment of these instants with respect to other

instants or intervals can then be investigated or exploited.

There are at least five conceivable approaches to labelled instants:

1. nodes could be optionally labelled; or
2. an instant can be modelled as a self-loop on a node, and again labelled just like any other arc; or
3. instants can be treated as arcs between two nodes with the same time reference; or
4. instants can be treated as short periods, where these are labelled arcs just like any other; or
5. certain types of labels on periods would be interpreted as referring to the commencement or the culmination of that period.

With little evidence on which to base a decision between these options we opt for the most conservative, which is that embodied in the last two options (which are not mutually exclusive). Thus with no extension to the ontology we already have two ways to model instants.

4.4. Overlap and gaps

As we have seen, annotations are often stratified, where each layer describes a different property of a signal. What are the possible temporal relationships between the pieces of a given layer? Some possibilities are diagrammed in Figure 8, where a point is represented as a vertical bar, and an interval is represented as a horizontal line between two points.

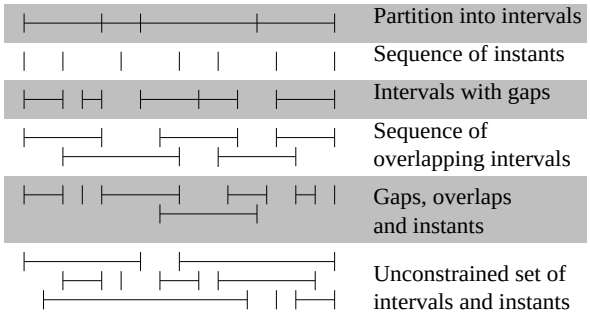

Figure 8: Possible Structures for a Single Layer

In the first row of Figure 8, we see a layer which exhaustively partitions the timeflow into a sequence of non-overlapping intervals (or perhaps intervals which overlap just at their endpoints). The second row we see a layer of discrete instants. The next two layers illustrate the notions of gaps and overlaps. Gaps might correspond to periods of silence, or periods which have yet to be annotated. Overlaps will be motivated below in the context of hierarchies built over multiple streams. The fifth row combines the possibilities of rows 2-4, while preserving the notion of sequence. The final row abandons the sequencing requirement, permitting arbitrary sets of intervals and instants. One motivation for abandoning the sequencing requirement is in order to represent a syntactic tree as a single layer of annotation. In fact, this is a reasonable approach for any hierarchical structure which either cannot be stratified in principle, or whose stratification requires an unbounded number of layers.

For the sake of simplicity and generality, we adopt this last option, dropping its use of instants (see §4.3). Thus, a layer of annotation is just an arbitrary collection of intervals. Additionally, some intervals may lack time references for one or both of their endpoints, while remaining (partially) ordered with respect to other endpoints.

As promised, the remainder of this section motivates the overlap requirement. Browman and Goldstein [3] describe a gestural score formalism with separate levels for each of the independent articulators (lips, tongue-tip, tongue-body, velum, glottis). A word typically contributes gestures on more than one level, where the levels do not stand in any kind of hierarchical relationship with respect to each other.

In this context, Browman and Goldstein discuss the example /ten pin/ $\rightarrow$ [tempin]. Here, the labial gesture of the p overlaps the coronal gesture of the n. Since these gestures come from different words, the words themselves must be overlapping rather than strictly ordered. An annotation structure showing the relationship between the `L` (lip) gestures, the `T` (tongue-tip) gestures and the words is shown in Figure 9. Observe the overlap between the two `W` (word) annotations. Here is a situation where the word level is no longer a strict linear ordering. Similar constructions can be devised at other levels of hierarchy, such as the syllable and the segment. (See [2] for a logical formalism for exploring interactions between hierarchical, sequential and overlapping structure in speech.)

4.5. Multiple nodes at a time-point

Two labelled periods of an annotation might begin (or end) at the same time. The alignment of two such boundaries might be necessary, or pure coincidence.

As an example of necessary alignment, consider the case of phrase-initial words. Here, the left boundary of a phrase lines up with the left boundary of its initial word. Changing the time of the phrase boundary should change the time of the word boundary, and vice versa. In the general case, an update of this sort must propagate both upwards and downwards in the hierarchy.

We would say that these two pieces of annotation actually share the same boundary; their arcs emanate from a single node. Change to the time reference of that node does not need to propagate anywhere, since the information is already shared by the relevant arcs.

As an example of coincidental alignment, consider the case of gestural scores. Two component gestures of a segment (or syllable, word, ...) might happen to start (or end) at the same time. However, changing the start time of one gesture usually carries no implication for the start time of the other gesture. Therefore, we would say that these two gestures are represented by arcs emanating from distinct nodes, where these nodes happen to have the same time reference. A change to the time on one node does not propagate to the other node.

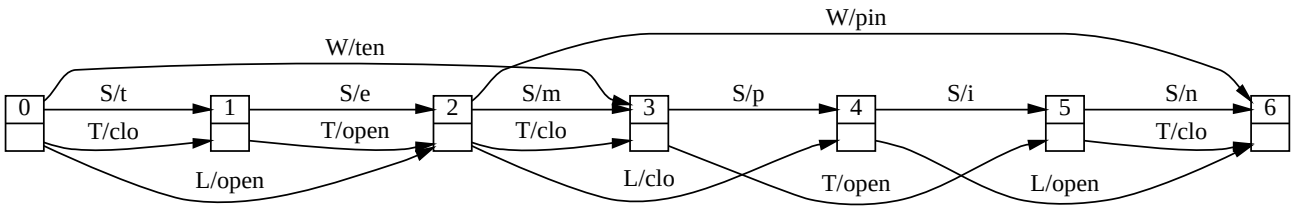

Figure 9: Example Hierarchy with Multiple Streams

4.6. Multiple arcs and labels

It is often the case that a given fragment of speech has multiple possible labels. For example, the stretch of speech corresponding to a monosyllabic word is both a syllable and a word, and in some cases it may also be a complete utterance. The combination of two independent annotations into a single annotation (through union) may also result in two labels for the same fragment.

In the general case, a label could be a (typed) attribute-value matrix, possibly incorporating nested structure, list- and set-valued attributes, and even disjunction.

Our working hypothesis is that using typed labels, with atomic types and labels, is sufficient. Multiple labels spanning the same material can reside on their own arcs. This way, their endpoints can be varied independently if necessary (see §4.5), and the combining and projection of annotations does not require collapsing and splitting of arcs.

(It will often be the case that multiple arcs emanating from a node will have distinct types. This explains why we have opted for node identifiers but not arc identifiers. We can usually reference an arc uniquely using a node identifier and a label type.)

4.7. Durable citations

Supposing that a query returned a reference into an annotated corpus and we wished to include that reference in a document. We would like this citation to be as durable as possible, surviving updates to the annotation and its indexes, whether the citation refers to a particular (superseded) version of the corpus, or whether it is subsequently construed with respect to a more recent version.

The most immutable aspect of a corpus is its signal data. Modulo a single base value, an offset into signal data is the most durable kind of reference possible. The unique node identifiers of an annotation provide another kind of anchor for citations. For durability of citation, node identifiers should not be changed unnecessarily, i.e. an edit operation on a fragment of an annotation should preserve node identifiers wherever possible. Another kind of reference might be to structural position within a hierarchical annotation. For example, we could refer to the 3rd syllable of the 2nd phrase of the 14th utterance of the 5th speaker turn of a corpus. In our conception, this information does not reside in the annotation, but it is compiled into the hierarchy-local index of the annotation. This kind of citation is the least durable in terms of the speech data it references, but the most durable in terms of the abstract structure. Other kinds of reference are almost never durable, such as byte offsets into raw annotation data, and so these should be avoided.

4.8. Associations between annotations and files

An ‘annotated corpus’ is a set of annotation graphs and an associated set of files. Multiple annotations may reference a single file, or a single annotation may reference multiple files. Equally, any given layer of an annotation may reference no file at all, or the same file as other layers, or its own unique file, or even a collection of files which may be (partially) shared with other layers. Apart from files, an annotation may reference other annotations. The relationship between a set of annotation graphs and a set of files might be particularised at query time.

How should this connection be formalised? While the annotations include time references, time-function files may have arbitrary offsets of their own, along with a sampling rate (or some other mechanism) that provides the time line. The file format may not support direct indexing at the granularity used in the annotation (e.g. if block compression has been used) in which case the resolution of time references must be mediated by some other program.

The situation is more complex in the spatial domain. Although it is not our primary focus, we would like the annotation formalism to be extensible to spatially-specific annotations of video signals or of any other time-function data having similar properties, perhaps by enriching the temporal anchors with spatial information. Anthropologists, conversation analysts and sign-language researchers are already producing annotations that are anchored not only to time spans but also to a particular spatial trajectory through the corresponding series of video frames.

In the more restricted domain, time anchors are absolute references into data, modulo a single time offset number. However, in this more general domain, the spatial anchoring used for annotating video sequences is necessarily relative to a specific video; spatial annotations will be mostly not about absolute space but rather about a particular camera's 'angle' on space. Such an annotation cannot be transferred from one video sequence to another unless there is some way to transform any camera's view of a scene (location/angle/zoom) into quasi-absolute 3-D coordinates.

In recognition of these issues, all we can do is to leave implicit the connection between sets of annotations and the collections of files they describe. There seems to be little point in formally encoding the relationship when its potential structure is so diverse, or even ill-defined. Where this connection can be made its character will be obvious and the details can be included in the documentation of the corpus.

5. TOWARDS AN ALGEBRAIC FRAMEWORK

We maintain that most, if not all, existing annotation formats can naturally be treated, without loss of generality, as directed acyclic graphs having typed labels on (some of) the edges and time-marks on (some of) the vertices. We call these 'annotation graphs'. For the sake of explicitness, these structures are defined formally in this section, although our primary commitment is to the details of the preceding discussion rather than any particular aspect of this formalisation.

5.1. Labels and types

A **label set L** is a collection of ordered pairs $\langle t, n \rangle$ where $t \in T$ is a type and $n \in N_t$ is a name drawn from a set of names specific to type t. The set of types T is a poset, representing a partial hierarchical structure on the types.

In the limiting cases, the poset T will either be a chain – a total ordering – forcing the annotation structure to be strictly hierarchical (c.f. the 'strict layer hypothesis' of prosodic phonology), or an anti-chain – an unordered set – forcing a flat structure of independent parallel streams (c.f. the 'gestural score' notation).

5.2. Annotation graphs

Formally, an **annotation graph A** over a label set L is a 4-tuple $\langle V, E, \lambda, \tau \rangle$ which satisfies the following conditions:

1. E is an irreflexive, asymmetric, transitive relation on V;
2. $\lambda : E \rightharpoonup L \cup T$ is a partial function assigning a label or just a type to an arc.
3. $\tau : V \rightharpoonup \Re$ is an order-preserving map assigning times to (some of) the vertices.

There is no requirement that annotation graphs be connected or rooted, or that they cover the whole of the speech file they describe. However, we believe that most annotations will satisfy the following property, and it may be a useful constraint to build in:

An annotation graph is **semi-anchored** if any vertices lacking an incoming arc and any vertices lacking an outgoing arc are assigned a time reference.

This means that, from any unanchored vertex there is a chain to some anchored vertex, and a chain from some anchored vertex, constraining the temporal locus of the unanchored vertex. Cursory examination of the example annotation graphs given in §3 will show that they are all semi-anchored.

5.3. Query

On our algebraic approach, queries are nothing other than expressions in a calculus defined over annotation graphs. This calculus is built up recursively from elementary graphs by combining them in various ways, including conjunction, disjunction, concatenation and Kleene closure, in the analogous fashion to the way regular expressions are built up in an RE calculus. Coreference of arbitrary edges between conjuncts is accomplished using operations analogous to the reference operators available in extended regular expression formalisms (such as that of Perl).

A variety of approaches is possible for the syntax. We sketch one possibility below, although we are not committed to the details. The syntax will be exemplified using the information in the Emu example (Figure 7).

```
TYPE   ::=  <type> | <type-var> | '.';
NAME   ::=  <name> | <name-var> | '.';
LABEL  ::=  TYPE '/' NAME | '.';
```

A 'Label' is used to describe an arc of an annotation graph. Some example labels follow: `S/r`, `P/r`, `./r`, `W/price`, `Syl/.`, `Accent/H*`.

We adopt the abbreviatory convention that the wildcard dot for types and names is not written, and the slash is only used when a name appears. Variables will be distinguished using the '$' prefix.

Next we define query syntax for the nodes of our annotation graphs.

```
ID    ::=  <id> | <id-var> | '.';
TIME  ::=  <time> | <time-var> | '.';
NODE  ::=  '<' ID '/' TIME '>';
```

Nodes contain an identifier and a time, or a variable or wildcard ranging over either of these. Some example nodes follow: `<13/.>`, `<$i/.>`, `<./0.520>`, `<./$t>`, `<4/$t>`, `<$i/1.069>`, `<$i/$t>`.

We adopt the abbreviatory convention that the wildcard dot is not written, and the slash is only used when a time is specified.

Next we define a term, the simplest expression of the query language, as any sequence of nodes and labels.

```
TERM  ::=  NODE
       |   LABEL
       |   TERM+
       |   '(' TERM ')';
```

The parenthesis notation allows the identification of arbitrary

subexpressions; to which references will be returned by the query, as discussed below.

Here are some terms specifying an individual arc and the nodes on either side: `<2/0.404> S/p <3/0.465>`, `<2> S/p <3>`, `<$i> S/p`, `S/p <3>`. `S/p`. Here are some terms describing sequences of arcs (see Figure 7):

```
<2> P/p <4> P/r <6> P/ai <7> P/s <8>
<2> P/p (P/r) (P/ai) P/s <8>
P/p (P/r <6>) P/ai P/s
P/p P/r P/ai P/s
<2> P P P P <8>
Syl/S P/r
</.404> (.)
```

Finally, complex expressions are built up out of simple expressions using the following syntax.

```
UNARY-OP   ::=   '*' | '+' | '?';
BINARY-OP  ::=   '&' | '|' | '+';
EXPR       ::=   EXPR UNARY-OP
               |  EXPR BINARY-OP EXPR;
```

One could conceivably add a restricted kind of negation, covering the situation where a positive and a negative expression are conjoined at the same position (c.f. Perl's negative look-ahead assertions).

The following examples illustrate the use of the expression syntax, with particular emphasis on the role of variables of different kinds.

```
<$i> Syl/S <$j> & <$i> P/p P/r P/ai P/s <$j>
<$i> Syl/S <$j> & <$i> P P (P) P <$j>
<$i> Syl/S <$j> & <$i> .* P/r .* <$j>
<$i> Syl/S & <$i> P (P)
<$i> Syl/S <$j> Syl/S <$k> &
   <$i> .* P/$p .* <$j> .* P/$p .* <$k>
<$i> S/$p & <$i> P/$p
```

The query language is not necessarily what users would employ. For example, there could be a macro `incl(Syl, P/r)` which expands to the third query above.

In considering how queries should operate and what kinds of results they should return, we are influenced by the Unix 'egrep' program, and by the parenthesis notation found in the extended regular expressions used by the Perl language [14]. In one case, we iterate over the data in certain units (which may be paragraphs, lines, words or whatever) and return the whole unit if there is a match. In the other case, certain subexpressions of the query are specially marked (using parentheses) and references to the material matching these subexpressions is returned.

Note that the query syntax is not complete. For example, additional syntax is needed in order to refer to the type hierarchy, and in order for a query to particularise the association between the annotations and the files (cf §4.8).

6. INDEXING

Corpora of annotated texts and recorded signals may range in size from a few thousand words up into the billions. The data may be in the form of a monolithic file, or it may be cut up into word-size pieces, or anything in between. The annotation might be dense as in phonetic markup or sparse as in discourse markup, and the information may be uniformly or sporadically distributed through the data.

At present, the annotational components of most speech databases are still relatively small objects. Only the largest annotations would cover a whole hour of speech (or 12,000 words at 200 words per minute), and even then, a dense annotation of this much material would only occupy a few hundred kilobytes. In most cases, serial search of such annotations will suffice. Ultimately, however, it will be necessary to devise indexing schemes; these will necessarily be application-specific, depending on the nature of the corpus and of the queries to be expressed. The indexing method is not a property of the query language but a way to make certain kinds of query run efficiently. For large corpora, certain kinds of query might be essentially useless without such indexing.

At the level of individual arc labels, we envision three simple indexes, corresponding to the three obvious dimensions of an annotation graph:

A time-local index
: For each time locus where we have annotation data (at some suitable level of granularity) we maintain pointers to edges in the annotation file (perhaps the line number). We can then quickly find any edges in a temporal region, even if their endpoints fall far outside the region. We can also find the set of nodes having the same time-reference (for queries using nodes with time variables).

A type-local index
: For each type we maintain a table of pointers to edges of that type.

A hierarchy-local index
: This index stores the hierarchical relationship between arcs (cf. §4.2), allowing us to find the daughters of any label.

These indices would be application specific. Under one approach, they would provide three categories of iterators. It would be the task of any implementation to make sure that the basic encoding is consistent with itself, and that the conglomerate structure (basic encoding plus indexes) is consistent.

More broadly, the design of an application-specific indexing scheme will have to consider what kinds of sequences or connections among tokens are indexed. In general, the indexing method should be based on the same elementary structures from which queries are constructed. Indices will specify where particular elementary annotation graphs are to be found, and so a complex search expression can be limited to those regions for which these graphs are necessary parts.

In future work we intend to apply methods from textual indexing and from graph indexing [10], building on the existing LDC-Online indexing scheme [15].

7. FILE ENCODINGS

As stated at the outset, we believe that the standardisation of file formats is a secondary issue. The identification of a common conceptual framework underlying all work in this area is an earlier milestone along any path to standardisation of formats and tools. That said, we believe that file formats should be transparent encodings of the annotation structure.

The flattest data structure we can imagine for an annotation graph is an unordered list of 5-tuples, one per arc, consisting of an optional label plus an identifier and optional time reference for both nodes spanned by the arc. This is the 'basic encoding' of the annotation, to be accessible in perpetuity. This basic encoding will often be accompanied with various application-specific indexes, as discussed above, facilitating queries and updates. However, all queries and updates should operate on the basic encoding, using the indexes as a kind of look-through cache, exploited where available and built on the fly when necessary.

Let us consider the implications of various kinds of annotation updates for the file encoding. The addition of new nodes and arcs simply involves concatenation to the basic encoding (recall that the basic encoding is an unordered list of arcs). The same goes for the addition of new arcs between existing nodes. For the user adding new annotation data to an existing read-only corpus – a widespread mode of operation – the new data can reside in one or more separate files, to be concatenated at load time. The insertion and modification of labels for existing arcs involves changing one line of the basic encoding.

Adding, changing or deleting a time reference involves nonlocal change to the basic encoding of an annotation. This can be done in either of two ways: a linear scan through the basic encoding, searching for all instances of the node identifier; or indexing into the basic encoding using the time-local index to find the relevant lines of the basic encoding. Of course, the time reference could be localised in the basic encoding by having a separate node set. This would permit the time reference of a node to be stored just once. However, we prefer to keep the basic encodings as simple as possible.

Maintaining consistency of the temporal and hierarchical structure of an annotation under updates requires further consideration. In the worst case, an entire annotation structure would have to be validated after each update. To the extent that information can be localised, it is to be expected that incremental validation will be possible. This might apply after each and every update, or after a collection of updates in case there is a sequence of elementary updates which unavoidably takes us to an invalid structure along the way to a final, valid structure.

8. CONCLUSION

We have presented a general purpose annotation formalism for linguistic signals, satisfying the three desiderata of generality, searchability and maintainability. Our commitment is not to the details of the particular formalism sketched here, but to the adoption of a simple and expressive conceptual model. The model can be encoded in a variety of file formats, and it can be enriched in appropriate ways by specific applications, to support particular user interface needs and to optimise certain kinds of search.

9. ACKNOWLEDGEMENTS

We are grateful to the following people for discussions which have helped to clarify our ideas about annotations: Peter Buneman, Steve Cassidy, Paul Taylor, and to participants of the COCOSDA workshop at ICSLP-98.

10. REFERENCES

1. Toomas Altosaar. Presentation at COCOSDA-98, 1998. www.acoustics.hut.fi.
2. Steven Bird. *Computational Phonology: A Constraint-Based Approach.* Studies in Natural Language Processing. Cambridge University Press, 1995.
3. Catherine Browman and Louis Goldstein. Articulatory gestures as phonological units. *Phonology*, 6(2):201–51, 1989.
4. Steve Cassidy and Jonathan 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/].
5. John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathon G. Fiscus, David S. Pallett, and Nancy L. Dahlgren. *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM* NIST, 1986. [www.ldc.upenn.edu/lol/docs/TIMIT.html].
6. Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog: An Introduction to Computational Linguistic*s Addison-Wesley, 1989.
7. Susan R. Hertz. The delta programming language: an integrated approach to nonlinear phonology, phonetics, and speech synthesis. In John Kingston and Mary E. Beckman, editors,*Papers in Laboratory Phonology I: Between the Grammar and Physics of Speech*, chapter 13, pages 215–57. Cambridge University Press, 1990.
8. Brian MacWhinney. *The CHILDES Project: Tools for Analyzing Talk.* Mahwah, NJ: Lawrence Erlbaum., second edition, 1995. poppy.psy.cmu.edu/childes/.
9. Mitchell P. Marcus, Beatrice Santorini, and Mary Ann 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.
10. Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.
11. NIST. A universal transcription format (UTF) annotation specification for evaluation of spoken language technology corpora. [www.nist.gov/speech/hub4_98/utf-1.0-v2.ps], 1998.
12. Florian Schiel, Susanne Burger, Anja Geumann, and Karl Weilhammer. The Partitur format at BAS. In*Proceedings of the First International Conference on Language Resources and Evaluation*, 1998. [www.phonetik.uni-muenchen.de/Bas/BasFormatseng.html].
13. Paul A. Taylor, Alan W. Black, and Richard J. Caley. The architecture of the the Festival speech synthesis system. In*Third International Workshop on Speech Synthesis*, Sydney, Australia, November 1998.
14. Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl.* O'Reilly and Associates, 2nd edition, 1996.
15. Zhibiao Wu and Mark Liberman. LDC Online: A digital library for linguistic research and development. In*Proceedings of the Second ACM Conference on Digital Libraries*, New York: ACM, 1997. [www.ldc.upenn.edu/lol].