Jens Michaelis, Uwe Mönnich

Towards a Logical Description of Trees in Annotation Graphs

The importance of annotation graphs increases rapidly due to new developments in multi-media applications, text technology, and semantic web technologies. The paper provides a logical specification of trees in annotation graphs commonly used to code documents that are inherently structured on various levels.

1 Introduction

It is a matter of fact that a long history in artificial intelligence and computational linguistics tries to develop tools to extract semantic knowledge from syntactic information. In particular, from a text technological point of view the general research perspective is to extract (semantic) information from annotated documents. Regarding this aim, some of the relevant annotation models used are:

- Multilayer annotations
- Hyperlinks
- Discourse structure
- (Classical) linguistic description levels

An important aspect concerning annotation models is the development of a suitable logic system in order to specify the syntactic and semantic structures of such models. Due to the fact that heterogeneous subsystems must be incorporated, an amalgamation process can be assumed as the underlying architecture. The challenges of such architectures are twofold: first, the dynamic interaction between syntactic and semantic information must be represented and second, the efficiency of algorithms must be guaranteed.

Fortunately, in the case of annotation graphs, the object of these remarks, techniques from parameterized complexity theory can be exploited. This theory provides powerful tools for a detailed investigation of algorithmic problems. As it turns out, the concept of treewidth, indicating the similarity of a graph or a relational structure with a tree, is a parameter which helps to show that many otherwise intractable problems become computable in linear time when restricted to tree-like inputs.

The key feature of annotation graphs is their abstraction from the diversity of concrete formats used for the transcription of text and speech. This feature makes them an ideal candidate for the comparison of different annotation systems, e.g., those currently developed by several linguistic collaborative research centres in Germany.

Translating these different representation schemes into the framework of annotation graphs is a necessary prerequisite for the transfer of the pleasant computational properties of annotation graphs to the original systems. This is particularly important in those circumstances where the natural data structure of the concrete markup system cannot be readily understood as describing trees, or at least, multi-rooted trees, taken into account the possibilty of multiple annotation layers. Our main result can be interpreted as specifying the conditions under which algorithmic methods that were designed for trees can be applied to the realm of annotation systems that are apparently based on underlying graph structures.

A classical domain for multilayered annotations is linguistics. Utterances of speakers can be considered from different perspectives: examples are syntactic, semantic, discourse and intonation aspects, just to mention some of them. Representing these types of data in one representation format yields overlapping hierarchies. Moreover, the resulting structures are naturally considered as graphs rather than trees.

These challenges can be met by representing annotation graphs in logical form. Benefits of such a representation are the low descriptive complexity, the representability as open diagrams, or the abstract format for the interaction of different linguistic levels. Furthermore logical representations are amenable to the arsenal of techniques from logical graph theory.

2 Annotation Graphs

We start this section by explicitly providing the corresponding formal definitions, first, and by looking, in particular, at one example in more detail, afterwards.

2.1 Definitions and Examples

The underlying definition of an annotation graph is specified as follows:

Definition 2.1 (Bird and Liberman 2001) An *annotation graph (AG)*, G, over a label set L and a family of timelines $\langle \langle T_i, \leq_i \rangle \rangle_{i \in I}$, I being some index set,[1] is a 3-tuple $\langle N, A, \tau \rangle$ consisting of a node set N, a collection of labeled arcs $A \subseteq N \times N \times L$, and a partial time function $\tau : N \rightharpoonup \bigcup_{i \in I} T_i$, which satisfies the following two conditions:

(i) $\langle N, A \rangle$ is a labeled acyclic digraph containing no nodes of degree zero, and

(ii) for any path from node n_1 to n_2 in A, if $\tau(n_1)$ and $\tau(n_2)$ are defined, then there is a timeline $\langle T_i, \leq_i \rangle$ such that $\tau(n_1), \tau(n_2) \in T_i$, and such that $\tau(n_1) \leq_i \tau(n_2)$.

Remarks. AGs may be disconnected or empty, and they must not have orphan nodes. It follows from the definition that every piece of *connected* annotation structure can refer to at least one timeline. In Figure 1, an AG from the linguistics domain is depicted.

[1] Thus, for each $i \in I$, timeline $\langle T_i, \leq_i \rangle$ consists of a nonempty set T_i and a total order $\leq_i$ on T_i.

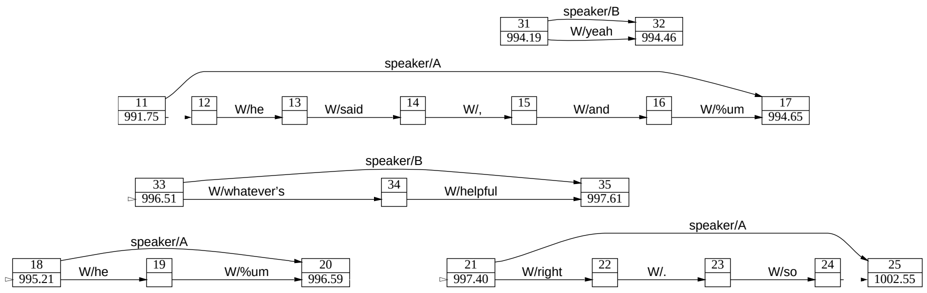


Figure 1: An example of an anchored annotation graph (cf. Bird and Liberman 2001).

Definition 2.2 (Bird and Liberman 2001) An *anchored AG* is an AG, G, defined as in Definition 2.1 and additionally satisfying the following condition:

(iii) if any node n does not have both incoming and outgoing arcs then $\tau : n \mapsto t$ for some time t.

Remarks. For anchored annotation graphs, it follows from the definition that every node has two bounding times, and timelines partition the node set. The AGs depicted in Figure 1 are anchored.

Definition 2.3 (Bird and Liberman 2001) A *totally anchored AG* is an anchored AG, G, defined as in Definition 2.2 such that the time function $\tau : N \rightharpoonup \bigcup_{i \in I} T_i$ is total.

Definition 2.4 A totally anchored AG, G, defined as in Definition 2.3 is *time-crossing arc free* if for all $\langle p, q, l \rangle, \langle r, s, m \rangle \in A$ such that $\tau(p), \tau(r) \in T_i$ for some i, and such that $\tau(p) \leq_i \tau(r)$ then either $\tau(q) \leq_i \tau(r)$ or $\tau(s) \leq_i \tau(q)$ holds.

In particular, the EXMARaLDA annotation tool, developed by the linguistic collaborative research centre in Hamburg, can be seen as strongly relying on the formal concept of AGs (cf. Schmidt 2005). In fact, not using the full range of possibilities provided by the general AG-definition given above, the core model underlying an EXMARaLDA basic transcription provides a so-called *single timeline, multiple tiers (STMT)* model, which in strict AG-terms can be understood as being a totally anchored AG consisting of exactly one timeline (cf. Figure 2). In line with the assumption of Bird and Liberman (2001, p. 12f) that reference to a single timeline implies that nodes with the same time reference should be considered to be identical, the AG depicted in Figure 2 can formally be specified as in the next example.

Example 2.5 For $T_{\mathrm{ex}} = \{0, 1, 2, 3, 4, 5\}$ and $\leq_{\mathrm{ex}} = \leq_{\mathbb{N}} \upharpoonright T_{\mathrm{ex}} \times T_{\mathrm{ex}}$ consider the timeline $\langle T_{\mathrm{ex}}, \leq_{\mathrm{ex}} \rangle$.[2] Then for $\langle T_{\mathrm{ex}}, \leq_{\mathrm{ex}} \rangle$ and the label set L_{ex} implicitly specified via the

[2] $\mathbb{N}$ denotes the set of all non-negative integers, including 0. $\leq_{\mathbb{N}}$ is the canonical order on $\mathbb{N}$, and for each set $M \subseteq \mathbb{N}$, $\leq_{\mathbb{N}} \upharpoonright M \times M$ is the restriction of $\leq_{\mathbb{N}}$ to $M \times M$.

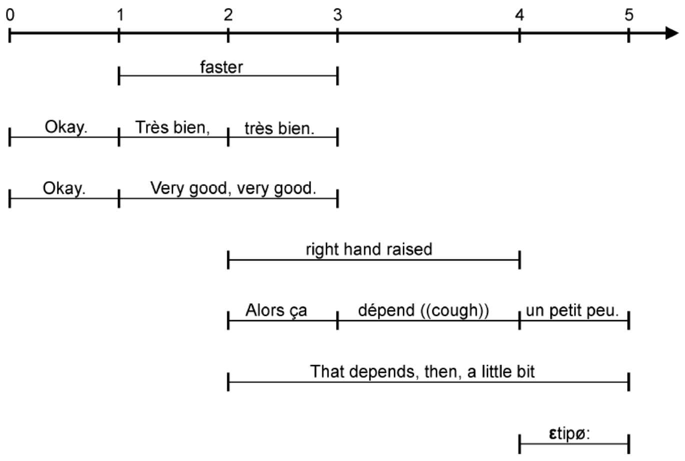


Figure 2: Example of an STMT model (cf. Schmidt 2005).

definition of the arc collection A_{ex}, let G_{ex} be the totally anchored AG $\langle N_{\text{ex}}, A_{\text{ex}}, \tau_{\text{ex}} \rangle$ over $\langle T_{\text{ex}}, \leq_{\text{ex}} \rangle$ (as the single timeline) and L_{ex}, where N_{ex} is the set $\{0, 1, 2, 3, 4, 5\}$, τ_{ex} is the identity function on $\{0, 1, 2, 3, 4, 5\}$ and A_{ex} consists of the following directed labeled edges:[3]

$$
\begin{aligned}
a_1^{(1)} &= \langle 1\,, 3\,, \text{faster} \rangle\,, \\
a_1^{(2)} &= \langle 0\,, 1\,, \text{Okay.} \rangle\,, \\
a_2^{(2)} &= \langle 1\,, 2\,, \text{Très bien,} \rangle\,, \\
a_3^{(2)} &= \langle 2\,, 3\,, \text{Très bien.} \rangle\,, \\
a_1^{(3)} &= \langle 0\,, 1\,, \text{Okay.} \rangle\,, \\
a_2^{(3)} &= \langle 1\,, 3\,, \text{Very good, very good.} \rangle\,, \\
a_1^{(4)} &= \langle 2\,, 4\,, \text{right hand hand raised} \rangle\,, \\
a_1^{(5)} &= \langle 2\,, 3\,, \text{Alors ça} \rangle\,, \\
a_2^{(5)} &= \langle 3\,, 4\,, \text{dépend ((cough))} \rangle\,, \\
a_3^{(5)} &= \langle 4\,, 5\,, \text{un petit peu.} \rangle\,, \\
a_1^{(6)} &= \langle 2\,, 5\,, \text{That depends, then, a little bit} \rangle\,, \\
a_1^{(7)} &= \langle 4\,, 5\,, \varepsilon\text{tipø:} \rangle\,.
\end{aligned}
$$

[3]If the collection of arcs is, in fact, "simply" a set then $a_1^{(2)}$ and $a_1^{(3)}$ are identical. Since the intention here is actually that they are not identical, we assume them to be distinguishable either by some (at least technically) different labeling or by treating the collection of arcs as a multiset.

2.2 Logical Properties

In order to specify some logical properties, we need some additional concepts. First, we define a tree-decomposition of a graph.

Definition 2.6 (Robertson and Seymour 1986) A *tree-decomposition* of a graph $G = \langle V, E \rangle$ is a tree $T = \langle N, F \rangle$ together with a collection of subsets $\{T_u \mid u \in N\}$ of V such that $\bigcup_{u \in N} T_u = V$ and the following properties hold:[4]

1. For every edge $e = \langle a, b \rangle$ of G there is a $u \in N$ such that $\{a, b\} \subseteq T_u$

2. For all $u, v, w \in N$, if w lies on a path from u to v in T then $T_u \cap T_v \subseteq T_w$

The *width* of a tree-decomposition is equal to $\max_{u \in T}\{|T_u| - 1\}$

Note that a consequence of condition 2 from the above definition could be formulated as 2', namely, for all $u \in V$, $\langle \{v \in N \mid u \in T_v\}, F \cap (\{v \in N \mid u \in T_v\} \times \{v \in N \mid u \in T_v\}) \rangle$ is a tree. The important concept of *treewidth*, defined next, is an indicator for the tree-likeness of a given graph G.

Definition 2.7 (Robertson and Seymour 1986) The *treewidth* of a graph G is the minimum value of k such that G has a tree-decomposition of width k.

The treewidth of a class of graphs C is naturally defined as the smallest number k such that for all graphs G in C, their treewidth is smaller or equal to k. Annotation graphs have unbounded treewidth since arbitrary large grids can be considered as annotation graphs. In practical applications, though, one is mainly concerned with finite families of documents represented as annotation graphs. Trivially, these families are of bounded treewidth. In the following we therefore restrict our attention to such finite families of annotation graphs.

Fact 2.8 *Finite families of anchored annotation graphs are of bounded treewidth.*

Monadic Second-Order Logic (MSO) is a subset of second-order logic. MSO extends first-order logic by allowing quantification over subsets of the universe of discourse. In other words quantification over second-order variables with at most one argument position is allowed.

Annotation graphs can be conceived as finite relational structures with the set of nodes understood as universe of discourse, the family of arcs as binary relations and the times as monadic predicates. A compact representation can be given in terms of an open diagram (cf. Prolog facts).

Theorem 2.9 (Courcelle) *Every property expressible in MSO is verifiable in linear time on graphs of bounded treewidth.*

[4]Here, a *tree* is taken to be a connected acyclic graph. Later we will restrict our attention to the concept of a finite ordered tree as it underlies our Definition 3.2 of a *finite labeled tree.*

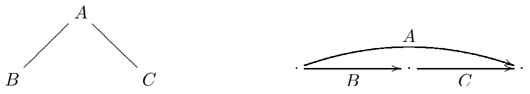
Courcelle's result is contained in Courcelle 1990. The principal tool in the proof of linear time decidability is the annotation of tree-like graph structures with logical types of bounded quantifier rank. These types assume the role of states in a bottom-up tree automaton.

Let MSO_2 designate the extended monadic second-order logic with quantification over sets of nodes and sets of edges.

Theorem 2.10 (Seese 1991) *If a set of graphs G has a decidable MSO_2 theory then it is the subset of the (homomorphic images of) a recognizable set of trees, i.e. it is* tree-definable *in the sense of Courcelle (2006).*

Remark. Trees are constructed from a finite set $\mathcal{F}$ of graph operations. Typical examples are disjoint union, relabeling of edges, addition of edges. Seese's theorem is an outgrowth of a combination of techniques from MSO-definable graph transductions and from the fundamental work of Robertson and Seymour on graph minors.

The representation of trees within the AG-framework is possible in terms of so-called *chart constructions*, where each tree node is mapped to an AG-arc, as outlined in the following easy example (cf. Cotton and Bird 2000):



The representation of trees in MSO_2-terms, relying on an order $\leq$, can be given via the notion of a *matching relation*. A set of edges M is called a *matching relation* if the following conditions are satisfied:

- $e \in M \wedge inc(e, u, v) \rightarrow u \leq v$ (M is compatible with $\leq$)
- $e \in M \wedge e' \in M \wedge inc(e, u, v) \wedge inc(e', w, z) \wedge u \leq w \leq v \rightarrow u \leq z \leq v$ (M is non-crossing)

where *inc* denotes the usual incidence relation.

Recall from the previous section the notion of a *totally anchored AG* that is *time-crossing arc free*. In case such an AG is anchored w.r.t. a single timeline, that notion is a particular instance of a matching relation.

The present section has served to emphasize the advantages that come with a logical description of graphs of bounded treewidth. The fact that a set of finite graphs is of bounded treewidth does not lead automatically to a representation of this set as a family of trees beyond the obvious tree decomposition as defined in Definition 2.6. Restricting the attention to the family of annotation graphs that satisfy the combined conditions characteristic of the single timeline, multiple tiers model, it can indeed be shown that these graphs allow a presentation in the form of multi-rooted trees. The next section is devoted to an elaboration of this claim.

3 Multilayer Annotation and STMT models

3.1 Multi-rooted Trees

Since our final aim will be to formally reconstruct an STMT model as a so-called *multi-rooted tree*, we here give some further explicit definitions setting the stage.

Definition 3.1 A *tree domain* is a nonempty set $D_\tau \subseteq \mathbb{N}^*$ such that for all $\chi \in \mathbb{N}^*$ and $i \in \mathbb{N}$ it holds that $\chi \in D_\tau$ if $\chi\chi' \in D_\tau$ for some $\chi' \in \mathbb{N}^*$, and $\chi i \in D_\tau$ if $\chi j \in D_\tau$ for some $j \in \mathbb{N}$ with $i < j$.[5]

Definition 3.2 A *finite labeled tree*, τ, is a quadruple of the form $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau, \mathit{label}_\tau \rangle$ where the triple $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is a finite (ordered) tree defined in the usual sense,[6] i.e. up to an isomorphism $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is the natural (tree) interpretation of some tree domain D_τ,[7] and where label_τ is the *labeling function (of τ)*, i.e. a function from N_τ into some set of *labels*, L_τ.

Note that, if $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau, \mathit{label}_\tau \rangle$, is a finite labeled tree, a tree domain, D_τ, whose natural (tree) interpretation is isomorphic to $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is uniquely determined. Note further that Definition 3.2 demands $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ to be "only" isomorphic to the natural (tree) interpretation of some tree domain, D_τ, and not to be necessarily a tree domain itself. Such a definition allows us to have two finite labeled trees with disjoint set of nodes, which, of course, is not the case for the corresponding tree domains whose natural (tree) interpretations are isomorphic to the underlying (non-labeled) trees. This possibility is exploited within the next definition.

Definition 3.3 For any finite string $\alpha \in \Sigma^*$ for some finite alphabet Σ, a *multi-rooted tree (over α)* is a finite tuple of the form $\langle \tau_r \rangle_{r<R}$ for some $R \in \mathbb{N}$, where for each $r < R$, τ_r is a finite labeled tree $\langle N_r, \triangleleft_r^*, \prec_r, \mathit{label}_r \rangle$ such that for each $r, s < R$ with $r \neq s$ it holds that $N_r \cap N_s = \emptyset$, and such that for each $r < R$ there are some $k(r) \in \mathbb{N}$, $\alpha_0^{(r)}, \dots \alpha_{k(r)}^{(r)} \in \Sigma^*$ and $w_0^{(r)}, \dots w_{k(r)+1}^{(r)} \in \Sigma^*$ for which the yield of τ_r is of the form $\alpha_0^{(r)} \cdots \alpha_{k(r)}^{(r)}$, and for which α is of the form $w_0^{(r)}\alpha_0^{(r)}, \dots w_{k(r)}^{(r)}\alpha_{k(r)}^{(r)}w_{k(r)+1}^{(r)}$.

Straightforwardly, adding a “super root” immediately dominating the tree components of a multi-rooted tree, $\langle \tau_r \rangle_{r<R}$, gives rise to a tree in the sense of Definition 3.2 again.

3.2 Finding a partition of a given STMT model into subgraphs

Given a totally anchored AG, G, consisting of exactly one timeline such that every two nodes with the same time reference are identical (i.e., G is an STMT model),

[5]For each set M, M^* is the Kleene closure of M, including ϵ, the empty string.

[6]N_τ is the finite, nonempty set of *nodes*, and $\triangleleft_\tau^*$ and $\prec_\tau$ are the respective binary relations of *dominance* and *precedence* on N_τ. Thus, $\triangleleft_\tau^*$ is the reflexive-transitive closure of $\triangleleft_\tau \subseteq N_\tau \times N_\tau$, the relation of *immediate dominance* on N_τ.

[7]In other words, up to an isomorphism N_τ is a tree domain such that for all $\chi, \psi \in N_\tau$ it holds that $\chi \triangleleft_\tau \psi$ iff $\psi = \chi i$ for some $i \in \mathbb{N}$, and $\chi \prec_\tau \psi$ iff $\chi = \omega i \chi'$ and $\psi = \omega j \psi'$ for some $\omega, \chi', \psi' \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$ with $i < j$.

the considerations of this subsection concern the aim of finding a partition of G into subgraphs each of which being without time-crossing arcs in the sense of Definition 2.4. This aim is motivated by the observation that time-crossing of two edges is usually caused by interference of two different layers of annotation, while single layers consist of edges in matching form, i.e. time-crossing arc free. A corresponding partition is not necessarily unique and there are, of course, several possible algorithms in order to calculate a particular instance of such a partition. Heuristics may help to decide which algorithm is to be preferred in terms of its computational properties. As an example, we here present an algorithm which can be seen as being a representative of a "left-to-right, top-down" strategy (cf. Figure 3). The perspective motivating this "tree-traversing" metaphor results from our final aim to reconstruct the given AG as a multi-rooted tree by using the corresponding partition. In the next section we will show, that such a reconstruction can be done "on the fly," when calculating the partition according to the presented algorithm.

For the rest of this section let $G = \langle N, A, \tau \rangle$ be a totally anchored AG over a label set L and a single timeline $\langle T, \leq_T \rangle$ such that every two nodes with the same time reference are identical. W.l.o.g. we may assume T is of the form $\{0, 1, \ldots, K-1\}$ for some $K \in \mathbb{N}$ such that τ is surjective, i.e., in particular the order on T is induced by the canonical order on $\mathbb{N}$. Since we are concerned with a single timeline and a totally anchored graph, we can also identify the set of vertices, N, with T. Thus, the cardinality of N is K. We construct the "upper part" of the corresponding adjacency matrix, $\langle A_{p,q} \rangle_{0 \leq p < q < K}$, where each entry $A_{p,q}$ consists of all arcs $a = \langle p, q, l \rangle \in A$ for some $l \in L$. The algorithm which provides us with a partition of A, Part A, is the following:

```
partition $A$
1  $r \leftarrow 0$
2  construct $A_r$                          % construct first partition set
3  Part $A \leftarrow \langle A_r \rangle$                  % initialize the list of partition sets
4  UNTIL $A_r = \emptyset$ REPEAT
5    $r \leftarrow r + 1$
6    construct $A_r$                        % construct next (potential) partition
7    IF $A_r \neq \emptyset$ THEN
8      Part $A \leftarrow$ Part $A \cdot \langle A_r \rangle$   % append next partition set to list
9    FI
10 RETURN Part $A$
```

Here the subprocedure **construct** A_r is defined as:

```
construct $A_r$
1  $A_r \leftarrow \emptyset$
2  IF $K > 0$ THEN[8]
3    explore $A_{0,K-1}$
4  FI
```

[8]Note that, because of (i) of Definition 2.1, $K > 1$ if N is nonempty.

For $0 \leq p < q < K$, **explore** $A_{p,q}$ is defined as follows:

```
explore $A_{p,q}$
IF $A_{p,q} \neq \emptyset$ THEN
  choose $a \in A_{p,q}$
  $A_r \leftarrow A_r \cup \{a\}$                  % add $a$ to partition set $A_r$
  $A_{p,q} \leftarrow A_{p,q} \setminus \{a\}$     % remove $a$ from set $A_{p,q}$
FI
$i \leftarrow 0$
$j \leftarrow 1$
WHILE $p+i < q-j$ and $A_{p+i,q-j} = \emptyset$ REPEAT % searching a leftmost ‘‘child’’
  IF $p+i < q-j-1$ THEN              % exploring ‘‘top-down’’ interval $[p+i,q-j]$
    $j \leftarrow j+1$               % reducing right interval value by $1$
  ELSE
    $i \leftarrow i+1$               % increasing left interval value by $1$
    $j \leftarrow 0$                 % right interval value back to $q$
  FI
$i_{p,q} \leftarrow i$
$j_{p,q} \leftarrow j$
IF $p + i_{p,q} < q - j_{p,q}$ THEN   % there is no ‘‘child’’ at all iff $p+i_{p,q}=q-j_{p,q}$
  explore $A_{p+i_{p,q},q-j_{p,q}}$   % exploring leftmost ‘‘child’’ -- nonemptiness
                                      of $A_{p+i_{p,q},q-j_{p,q}}$ is guaranteed by WHILE-loop
  IF $j_{p,q} > 0$ THEN
    explore $A_{q-j_{p,q},q}$         % searching for right ‘‘sibling’’ of leftmost
                                      ‘‘child’’
  FI
FI
```

Fact 3.4 *For $p, q < K$ with $p < q$, $i_{p,q}$ and $j_{p,q}$ (depending on $i_{p,q}$) are minimal, i.e., $A_{p+i,q-j} = \emptyset$ for all $j < q-(p+i)$ in case $i < i_{p,q}$, and for all $j < j_{p,q}$ in case $i = i_{p,q}$.*

Remark. Note that for $p, q < K$ with $p < q$, the WHILE-loop for potentially finding minimal i and minimal j, depending on i, within the subprocedure **explore** $A_{p,q}$ is a "left-to-right, top-down" search along the "rows" left-to and below matrix entry $A_{p,q}$ (cf. Figure 3). In particular, we have $A_{p+i,q-j} = \emptyset$ if $i < i_{p,q}$ and $q-(p+i_{p,q}) \leq j < q-(p+i)$. Hence **explore** $A_{p,p+i_{p,q}}$ would yield no contribution to the partition set A_r, if it were part of the subprocedure **explore** $A_{p,q}$.

Proposition 3.5 *The time complexity of* **partition** A *is in $O(K^2 m)$, m being the cardinality of A.*

Proposition 3.6 *Let $k \in \mathbb{N}$. Then for $K = 2k$, the label set $L = \{l_0, l_1, \ldots, l_{k-1}\}$ and the arc set $A = \{\langle p, p+k, l_p\rangle \mid 0 \leq p < k\}$ the time complexity of* **partition** A *is at least in $O(K^2 m)$ (as before, m being the cardinality of A, i.e., $m = k$ in this case).*

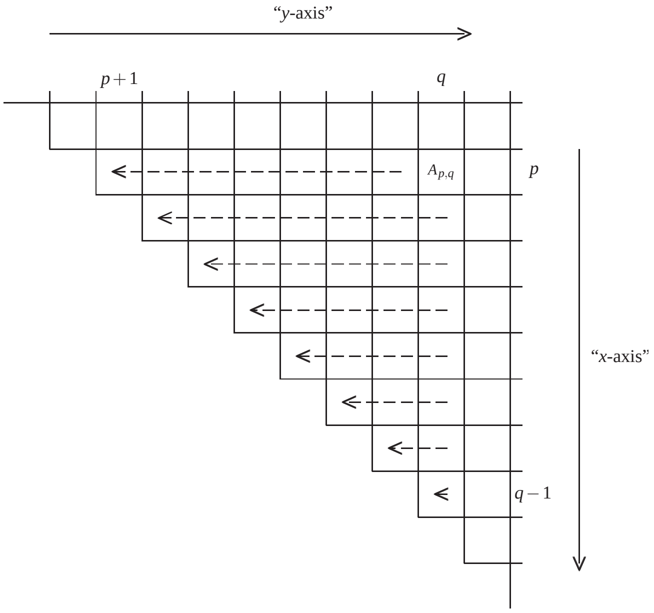


Figure 3: Potential search space for i and j within WHILE-loop of **explore** $A_{p,q}$.

As mentioned above, in order to find an appropriate partition of A, the algorithm given above only represents one of several more general possibilities depending on the strategy pursued. In fact, applying the algorithm to the example above yields a partition into three arc sets which might "contra-intuitively" partition the second, third and fifth layer, depending on which arc is chosen first from the arc subsets $A_{0,1}$, $A_{1,3}$, $A_{2,3}$ and $A_{4,5}$ according to line 2 of the subprocedure **explore**. If the second layer is not partitioned, the fifth will be, and vice versa. This is due to the fact that both $a_2^{(2)}$ and $a_2^{(5)}$ definitely belong to the set A_0, while $a_3^{(2)}$ and $a_1^{(5)}$ never will do so at the same time.

Example 3.7 (continued) Applying **partition** to A_{ex} yields a partition into three sets, A_0, A_1 and A_2, of the form

$$A_0 = \{x_0\,,\,x_1\,,\,a_2^{(2)}\,,\,x_2\,,\,a_2^{(5)}\,,\,x_3\}\,,\;A_1 = \{x'_0\,,\,x'_1\,,\,x'_2\,,\,x'_3\}\;\text{ and }\;A_2 = \{a_1^{(6)}\,,\,a_1^{(4)}\}$$

with $x_i, x'_i \in X_i$ such that $x_i \neq x'_i$ for $0 \leq i \leq 3$, where

$$X_0 = \{a_1^{(2)}, a_1^{(3)}\}\,,\;X_1 = \{a_1^{(1)}, a_2^{(3)}\}\,,\;X_2 = \{a_3^{(2)}, a_1^{(5)}\}\;\text{ and }\;X_3 = \{a_3^{(5)}, a_1^{(7)}\}.$$

Note that the situation of "algorithmic arbitrariness" immediately changes if we have further access to "external" information which can be used to guide the selection from an arc subset $A_{p,q}$, like hierarchical structure and ontological information in the sense of Bird and Liberman (2001, Section 3.2f), e.g., in terms of type and speaker as in an EXMARaLDA Basic Transcription Model itself based on a pure STMT model (cf. Schmidt 2005 and Figure 4 here).

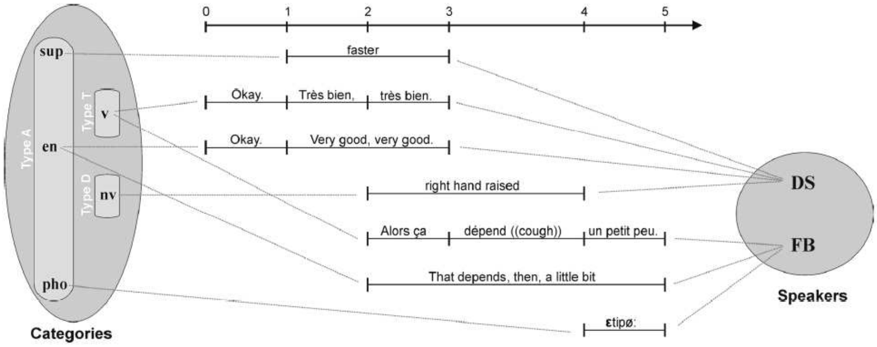


Figure 4: Example of an EXMARaLDA Basic Transcription Model (cf. Schmidt 2005).

3.3 Building a multi-rooted tree from an STMT model - "Reversing the chart construction"

Consistently taking over all formal prerequisites, we, in particular, let $G = \langle N, A, \tau \rangle$ be the totally anchored AG over a label set L and a single timeline $\langle T, \leq_T \rangle$ from the previous section. In order to build—on the fly—a multi-rooted tree, MultTree A, from Part A, the partition of A, created by applying **partition** to A, we slightly adapt and extend the algorithm.[9]

[9] Modified lines are indicated by a prime following the line number (x'). Additional lines are indicated by a subsequent line number counting (x.1, x.2 etc.).

partition A

```
% including the construction of a multi-rooted tree, MultTree A, from Part A
1      $r \leftarrow 0$
2'     construct $A_r, T_r$                 % construct first partition set and
                                              corresponding tree
3      Part $A \leftarrow \langle A_r \rangle$             % initialize list of partition sets
3.1    MultTree $A \leftarrow \langle T_r \rangle$         % initialize multi-rooted tree (list)
4      UNTIL $A_r = \emptyset$ REPEAT
5        $r \leftarrow r + 1$
6        construct $A_r, T_r$               % construct next (potential) partition
                                              set and corresponding tree
7        IF $A_r \neq \emptyset$ THEN
8          Part $A \leftarrow$ Part $A \cdot \langle A_r \rangle$        % append next partition set to list
8.1        MultTree $A \leftarrow$ MultTree $A \cdot \langle T_r \rangle$ % append next tree to multi-rooted
                                              tree (list)
9        FI
10     RETURN Part $A$
10.1   RETURN MultTree $A$
```

The subprocedure **construct** A_r, T_r is “just” an extension of **construct** A_r.

construct A_r, T_r

```
1      $A_r \leftarrow \emptyset$
1.1    $T_r \leftarrow \emptyset$
2      IF $K > 0$ THEN[10]
2.1      $\chi_{0,K-1} \leftarrow \epsilon$                % define (essential part of) root address of $T_r$
2.2      $k_{0,K-1} \leftarrow \epsilon$                % dummy for line 4.1 of procedure explore $A_{0,K-1}$
2.3      real root$_{0,K-1} \leftarrow$ FALSE     % potentially no arc from node 0 to node K-1
3        explore $A_{0,K-1}$
4      FI
```

For $0 \leq p < q < K$, the modified subprocedure **explore** $A_{p,q}$ is now given by

[10] Recall that, because of (i) of Definition 2.1, $K > 1$ if N is nonempty.

explore $A_{p,q}$

```
1      IF $A_{p,q} \neq \emptyset$ THEN
2        choose $a \in A_{p,q}$
3        $A_r \leftarrow A_r \cup \{a\}$                        % add a to partition set $A_r$
4        $A_{p,q} \leftarrow A_{p,q} \setminus \{a\}$                  % remove a from set $A_{p,q}$
4.1      $\chi_{p,q} \leftarrow \chi_{p,q} \cdot k_{p,q}$                    % define (essential part of) new
                                                   node address for $T_r$
4.2      $T_r \leftarrow T_r \cup \{\langle\langle \chi_{p,q}, r\rangle, label(a)\rangle\}$        % add to $T_r$ a corresponding new
                                                   node labeled by label(a)[11]
4.3      real root$_{p,q}$ $\leftarrow$ TRUE                     % no dummy node, cf. line 4.10, 11.1
4.4      $k \leftarrow 0$                                % potential next daughter to find
                                                   is leftmost child
4.5      IF $p + 1 = q$ THEN
4.6        $T_r \leftarrow T_r \cup \{\langle\langle \chi_{p,q} \cdot k, r\rangle, \langle p, p+1\rangle\rangle\}$  % yield of $T_r$ comprises (arc cover-
                                                   ing) interval [p,p+1]
4.7      FI
4.8    ELSE
4.9      IF $p = 0$ and $q = K - 1$ THEN
4.10       $T_r \leftarrow T_r \cup \{\langle\langle \epsilon, r\rangle, dummy\text{-}label_r\rangle\}$  % dummy root covering potential non-
                                                   time-crossing, non-inclusive arcs
                                                   (cf. $T_0$ and $T_1$ from Example 3.9)
4.11       $k \leftarrow 0$
4.12     ELSE
4.13       $k \leftarrow k_{p,q}$
4.14     FI
5      FI
6      $i \leftarrow 0$
7      $j \leftarrow 1$
```

[11]For each arc $a = \langle p, q, l\rangle \in A$ for some $p, q \in N$ and $l \in L$, we take $label(a)$ to denote its label l.

```
8      WHILE $p+i < q-j$ and $A_{p+i,q-j} = \emptyset$ REPEAT % searching leftmost ''child''
9        IF $p+i < q-j-1$ THEN
10         $j \leftarrow j+1$
11       ELSE
11.1       IF real $\mathrm{root}_{p,q}$ = TRUE THEN
11.2         $T_r \leftarrow T_r \cup \{\langle\langle\chi_{p,q}\cdot k, r\rangle, \langle p+i, p+i+1\rangle\rangle\}$ % yield of $T_r$ comprises (arc
                                                                   covering) interval [p+i,p+i+1]
11.3         $k \leftarrow k+1$
11.4       FI
12         $i \leftarrow i+1$
13         $j \leftarrow 0$
14       FI
15     $i_{p,q} \leftarrow i$
16     $j_{p,q} \leftarrow j$
17     IF $p+i_{p,q} < q-j_{p,q}$ THEN
17.1     $\chi_{p+i_{p,q},q-j_{p,q}} \leftarrow \chi_{p,q}$
17.2     $k_{p+i_{p,q},q-j_{p,q}} \leftarrow k$
17.3     real $\mathrm{root}_{p+i_{p,q},q-j_{p,q}} \leftarrow$ real $\mathrm{root}_{p,q}$
18       explore $A_{p+i_{p,q},q-j_{p,q}}$
19       IF $j_{p,q} > 0$ THEN
19.1       $\chi_{p+i_{p,q},q-j_{p,q}} \leftarrow \chi_{p,q}$
19.2       $k_{q-j_{p,q},q} \leftarrow k_{p+i_{p,q},q-j_{p,q}} + 1$
19.3       real $\mathrm{root}_{q-j_{p,q},q} \leftarrow$ real $\mathrm{root}_{p,q}$
20         explore $A_{q-j_{p,q},q}$                          % searching for right ''sibling'' of
                                                            leftmost ''child''
21       FI
22     FI
```

Remark. The previously stated results on the time complexity bounds of **partition** A are not affected (cf. Proposition 3.5 and 3.6).

Proposition 3.8 *For each T_r appearing as a component in MultTree A, let L_r denote the (label) set $L \cup \{\textit{dummy-label}_r\} \cup \{\langle p, p+1\rangle \,|\, 0 < p < K-1\}$. Then the set of first components of the first components of the elements of T_r, $\{\chi \in \mathbb{N}^* \,|\, \langle\langle\chi, r\rangle, l\rangle \in T_r$ for some $l \in L_r\}$, constitutes a tree domain. In this sense T_r can straightforwardly be interpreted as a finite labeled tree. All non-terminal nodes of T_r, except for the root node, are necessarily labeled by arc labels from L. The root node is labeled by dummy-label$_r$ in case $A_{0,K-1}$ was empty, while processing* **explore** *$A_{0,K-1}$. Otherwise it is also labeled by an element from L. Each leaf is labeled by $\langle p, p+1\rangle$ for some $p < K-1$.*

Example 3.9 (continued) Applying the modified algorithm **partition** to A_{ex}, in particular yields a multi-rooted tree MultTree $A_{\mathrm{ex}} = \langle T_0, T_1, T_2\rangle$ with

$$\begin{aligned}T_0 = \{&\langle\langle\,\epsilon\,,\,0\,\rangle\,,\,\textit{dummy-label}_0\,\rangle\,,\,\langle\langle\,0\,,\,0\,\rangle\,,\,\textit{label}(x_0)\,\rangle\,,\,\langle\langle\,00\,,\,0\,\rangle\,,\,\langle\,0\,,\,1\,\rangle\,\rangle\,,\\ &\langle\langle\,1\,,\,0\,\rangle\,,\,\textit{label}(x_1)\,\rangle\,,\,\langle\langle\,10\,,\,0\,\rangle\,,\,\textit{label}(a_2^{(2)})\,\rangle\,,\,\langle\langle\,100\,,\,0\,\rangle\,,\,\langle\,1\,,\,2\,\rangle\,\rangle\,,\\ &\langle\langle\,11\,,\,0\,\rangle\,,\,\textit{label}(x_2)\,\rangle\,,\,\langle\langle\,110\,,\,0\,\rangle\,,\,\langle\,2\,,\,3\,\rangle\,\rangle\,,\,\langle\langle\,2\,,\,0\,\rangle\,,\,\textit{label}(a_2^{(5)})\,\rangle\,,\\ &\langle\langle\,20\,,\,0\,\rangle\,,\,\langle\,3\,,\,4\,\rangle\,\rangle\,,\,\langle\langle\,3\,,\,0\,\rangle\,,\,\textit{label}(x_3)\,\rangle\,\rangle\,,\,\langle\langle\,30\,,\,0\,\rangle\,,\,\langle\,4\,,\,5\,\rangle\,\rangle\,\}\end{aligned}$$

$$\begin{aligned}T_1 = \{&\langle\langle\,\epsilon\,,\,1\,\rangle\,,\,\textit{dummy-label}_1\,\rangle\,,\,\langle\langle\,0\,,\,1\,\rangle\,,\,\textit{label}(x_0')\,\rangle\,,\,\langle\langle\,00\,,\,1\,\rangle\,,\,\langle\,0\,,\,1\,\rangle\,\rangle\,,\\ &\langle\langle\,1\,,\,1\,\rangle\,,\,\textit{label}(x_1')\,\rangle\,,\,\langle\langle\,10\,,\,1\,\rangle\,,\,\langle\,1\,,\,2\,\rangle\,\rangle\,,\,\langle\langle\,11\,,\,1\,\rangle\,,\,\textit{label}(x_2')\,\rangle\,,\\ &\langle\langle\,110\,,\,1\,\rangle\,,\,\langle\,2\,,\,3\,\rangle\,\rangle\,,\,\langle\langle\,2\,,\,1\,\rangle\,,\,\textit{label}(x_3')\,\rangle\,\rangle\,,\,\langle\langle\,20\,,\,1\,\rangle\,,\,\langle\,4\,,\,5\,\rangle\,\rangle\}\end{aligned}$$

$$\begin{aligned}T_2 = \{&\langle\langle\,\epsilon\,,\,2\rangle\,,\,\textit{dummy-label}_2\rangle\,,\,\langle\langle\,0\,,\,2\rangle\,,\,\textit{label}(a_1^{(6)})\,,\,\langle\langle\,00\,,\,2\rangle\,,\,\textit{label}(a_1^{(4)})\,,\\ &\langle\langle\,000\,,\,2\rangle\,,\,\langle\,2\,,\,3\rangle\rangle\,,\,\langle\langle\,001\,,\,2\rangle\,,\,\langle\,3\,,\,4\rangle\rangle\,,\,\langle\langle\,01\,,\,2\rangle\,,\,\langle\,4\,,\,5\rangle\,\rangle\,\}\end{aligned}$$

with $x_i, x_i' \in X_i$ such that $x_i \neq x_i'$ for $0 \leq i \leq 3$, where

$$X_0 = \{a_1^{(2)}, a_1^{(3)}\}\,,\ X_1 = \{a_1^{(1)}, a_2^{(3)}\}\,,\ X_2 = \{a_3^{(2)}, a_1^{(5)}\}\ \text{and}\ X_3 = \{a_3^{(5)}, a_1^{(7)}\}$$

(cf. Figure 5).

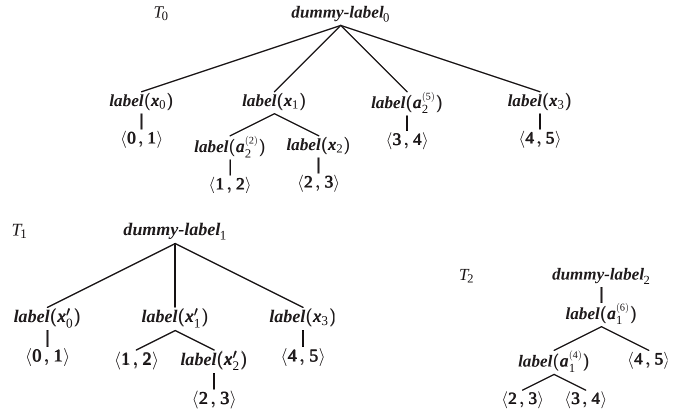


Figure 5: The tree components of the multi-rooted tree $\mathrm{MultTree}\,A_{\mathrm{ex}} = \langle T_0, T_1, T_2\rangle$.

4 Envoi

In this paper we have sketched the beginnings of a logical theory of annotation graphs. Along the way we have tried to emphasize the following points:

- Abstract logical framework with multilayer capabilities for linguistic annotations
- Compact logical representation
- Efficient MSO theory

- Subset of regular trees
- Internal definability of tree structure
- Partition of annotation layers

While the logical approach towards annotation models provides a unified format for the syntactic level it still has to be complemented with a component that serves to integrate syntactic with semantic structures. Primary candidates for this component are amalgamation techniques from model theory and the assembly of heterogeneous formal specifications via transformation systems. Care must be taken in this effort to preserve the nice complexity properties that are associated with finite graphs of bounded treewidth. On the other hand, annotation graphs offer a minimal formalization of typical transcription needs by means of acyclic graphs with fielded records on the edges. Semantic information is easily integrated into this minimal framework. It is for this reason that we believe that our general perspective on the formal properties of annotation graphs will retain its value if additional types of annotation are added to the current format of transcription schemes.

Acknowledgements

This research was partially supported by grant MO 386/3-4 of the German Research Foundation (DFG).

References

Bird, S. and Liberman, M. (2001). A formal framework for linguistic annotation. *Speech Communication*, 33:23–60.

Cotton, S. and Bird, S. (2000). An integrated framework for treebanks and multilayer annotations. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002),* Las Palmas de Gran Canaria, pages 1670–1677. European Language Resources Association.

Courcelle, B. (1990). The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75.

Courcelle, B. (2006). The monadic second-order logic of graphs XV: On a conjecture by D. Seese. *Journal of Applied Logic*, 4:79–114.

Robertson, N. and Seymour, P. D. (1986). Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322.

Schmidt, T. (2005). Time-based data models and the text encoding initiative's guidelines for transcription of speech. Arbeiten zur Mehrsprachigkeit, Folge B, Nr. 62 (Working Papers in Multilingualism, Series B, No. 62), Universität Hamburg, Hamburg.

Seese, D. (1991). The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53:169–195.