

# Chartparsing in Continuous Speech Recognition

Schamai Safra, TIK/ETH Zurich

COST 249 meeting of 29 Feb/1 Mar 1996

## 1 Introduction

In the ARCOS-G project experiments are conducted in a new approach to Continuous Speech Recognition where basic elements are detected by (traditional) statistical means like HMMs or ANNs, but then, putting the resulting basic element hypothesis together into the desired solution is done by means of chart parsing according to linguistic rules, e.g. lexicon, grammar, pronunciation rules etc.

Problems to solve are among others to adapt the classical methods of chart parsing – which are based on a given sequence of certain elements – to the much more vague input typical for speech recognition: a lattice of uncertain hypotheses, with gaps, overlaps and missing links.

In the following, after an informal introduction to chartparsing is given, some problems of applying chartparsing to CSR and possible ways to solve these are shown.

## 2 Parsing of Natural Language

“Parsing” is well known to us from computing. Given the syntax (Grammar) of some programming language, e.g.

```
ProgramModule =  
  'MODULE' Ident ';' .  
  {Import} Block Ident '.' .  
Import =  
  [ 'FROM' Ident ]  
  'IMPORT' IdentList ';' .  
Block = ...
```

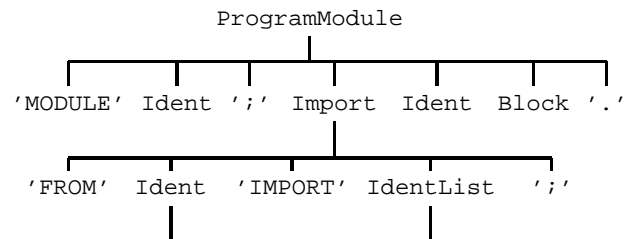
the aim of parsing any symbol sequence claiming to be a program of that language is either finding a syntax tree explaining the syntactic structure of the program according

to the language, or else, rejecting the symbol sequence as undervivable from that language.

For instance, given the following program,

```
MODULE hallo;  
FROM InOut IMPORT  
  WriteString,WriteLn;  
BEGIN  
  WriteString("Hallo World");  
  WriteLn;  
END hallo.
```

the syntactic structure the parser produces in this case could look like this (with many parts left out):



Now, programming languages, and in general formal languages, are designed to be unambiguous, i.e. that any symbol sequence has either one (a program) or zero (not a program) derivations from the syntax.

Not so for *natural* languages: If parsing should be applied to natural Languages (assuming we have a Grammar describing a non-trivial subset of some natural language), we have to consider ambiguities, since natural languages are ambiguous. A schoolbook example is the sentence

He saw the man with the telescope.

where it is not clear whether “the telescope” is the instrument of the observation or an object observed together

with “the man”. The problem is increased by orders of magnitude if we not only consider the ambiguity in one word sequence but in a lattice of word hypotheses as they occur in speech recognition:

??? saw/sore ??? man ??? telescope/tell us  
cope.

The danger of course is that ambiguity will prohibitively increase the parsing effort. On the other hand, we can observe that while there are several overall syntactic structures to one given input, these structures are almost entirely built from identical sub-structures. This is where chart parsing comes into play. Chart parsing is designed to parse ambiguous inputs in that it parses each sub-structure only once, even if it belongs to several structures.

To explain the working of chart parsing, let me use a parable, where a researcher community tries to solve a problem (the overall structure of the input), which can be done by solving different sets of sub-problems (sub-structures) first. The research community proceeds according to the following rules:

- Don't repeat successes:
  - Make your (partial) results public.
  - Search for published results first before trying to solve a (sub-)problem yourself.
- Don't repeat failures:
  - Publish what you are planning to do before you start to solve a (sub-)problem.
  - Don't try to solve a (sub-)problem if someone else already plans to do it.

Chart parsing saves efforts in the same two ways: the chart is a place where partial results but also intentions are stored and retrieved from. Therefore, sub-structures, once found, can be reused for building different alternative structures, but also, if one alternative failed, it is never tried again, since the intention to find this alternative is also marked in the chart and prohibits a second trial.

### 3 Chart-Parsing Basics

The following is a very informal, cartoon-like introduction to chartparsing, with all the formulas and many details left out.

Let's assume the following toy grammar for illustration:

```
NounPhrase = Art [ Color ] Noun.
Art = 'a' | 'the'.
Color = 'red' | 'green' | 'blue'.
Noun = 'house' | 'car'.
```

#### Chart

The chart is a directed graph, built from nodes and labeled edges. Nodes represent time points and edges represent either atoms (or terminals), i.e. the smallest parts from which to build the solution, the leaves of the parse tree, or more complex constituents (or non-terminals).

The starting situation before the parsing is a chart, initialized with just all known atoms.



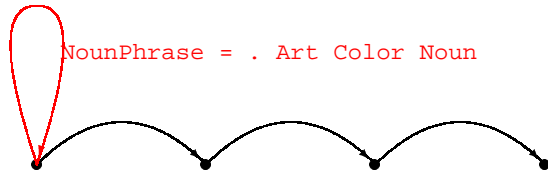
#### Non-Terminal Edges

Every non-terminal edge is labeled with the production rule of the constituent it stands for, e.g. `NounPhrase = Art Color . Noun`. There is a dot in the right hand side of the production rule separating the subconstituents already found from the subconstituents yet to be found. A dot to the very left of all subconstituents signifies a pure hypothesis, i.e., none of the subconstituents was found yet. The top hypothesis is an example of a pure hypothesis. A dot to the very left signifies a completely found constituent. An edge representing a completely found constituent is called 'inactive', edges with still some subconstituents to be found are called 'active'.

An edge spans only those atoms that belong to its found subconstituents. For instance a pure hypothesis starts and ends in the same node as it covers no subconstituents yet.

## Top Hypothesis

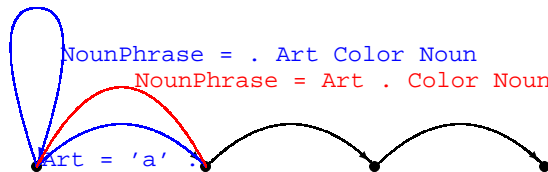
In order to trigger the search, the “Top Hypothesis”, an edge representing the intention to find the top constituent, has to be inserted into the chart. (Usually, the top hypothesis will be “Sentence”, but in our simple example it is NounPhrase.).



After insertion of the top hypothesis, new edges can be created by applying two basic operations, the “Rule Invocation” (hypothetization) and the “Fundamental rule”:

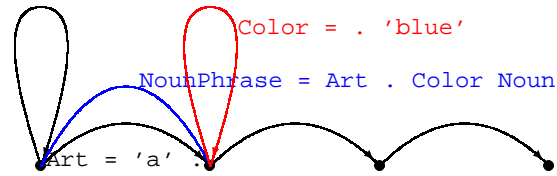
## Fundamental Rule (Expansion)

If an active edge (e.g. `NounPhrase = . Art Color Noun`) ends in a node where an inactive edge starts (e.g. `Art = 'a' .`), and this inactive edge corresponds to the next subconstituent which is needed to expand the active edge, then the expansion is performed by creating a new edge, starting at the active edge’s starting node and ending at the inactive edge’s ending node, labelled with the expanded rule (e.g. `NounPhrase = Art . Color Noun`).



## Rule Invocation

An active edge triggers the insertion of pure hypothesis edges at its end node, corresponding to the next subconstituent required in order to expand the triggering edge. In our example, the edge “`NounPhrase = Art . Color Noun`” triggers the creation of a pure hypothesis edge “`Color = . 'blue' .`”.



## Agenda

We see that the insertion of one edge into the chart can trigger the production of many other edges, which in turn have to be inserted into the chart. In order not to forget any edges (and thus to leave parts of the search space unexplored), edges are not inserted directly into the chart but are first inserted in a so-called *agenda*. The parsing process consists of taking one edge out of the agenda, inserting it into the chart while storing all newly created edges into the agenda, and so forth. The process is started by inserting the top hypothesis edge into the agenda and it stops as soon as the agenda becomes empty. This completes an exhaustive search through the whole search space.

## Search Strategy

Depending on where newly created edges are put into the agenda respectively how the next edge to be taken from the agenda is selected, a *search strategy* can be determined. The most simple queue or stack processing produce a width-first or depth-first strategy, more complex strategies can consider rule weightings (prior scoring) and/or terminal scores (posterior scoring). Of course, strategy is only relevant to non-exhaustive search: A good strategy means an increased chance to find the desired parse much before the whole search space was examined (e.g. if resource limitations force us to stop prematurely).

## 4 Are Linguistic Rules useful for CSR?

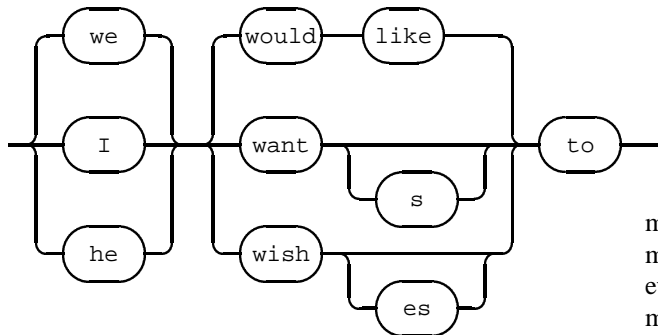
Even if linguistic rules and rule based approaches are not “in” in continuous speech recognition, they are nevertheless obviously used in some areas of CSR.

### Hard-wired Probabilistic Network

For instance, a 'Hard-wired Probabilistic Network' can be built from submodels according to a grammar. The Grammar

```
Request = Subj V Obj .
Subj    = 'I' | 'we' | 'he' .
V       = 'want' ['s'] | 'wish' ['s'] |
          'would' 'like' .
Obj     = 'to' ... .
```

can be (automatically) converted into the following intermediate recognition network:



Each word can then be replaced by a detailed probabilistic word model (e.g. HMM), possibly itself composed of subword models.

A hard-wired recognition network like this corresponds to a slightly over-productive language model in that it allows non-grammatical solutions like "I wishes to ...", but otherwise every path through the network is syntactically correct and (almost) no parsing is needed, once the path that corresponds the utterance is selected.

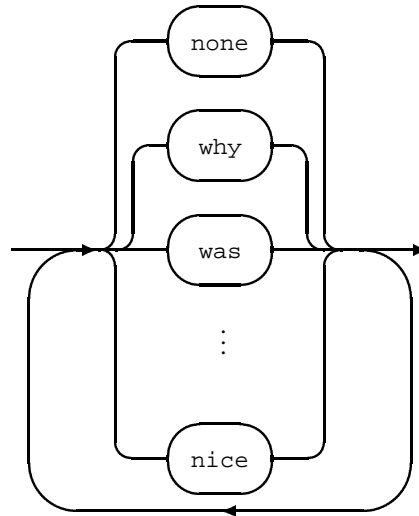
This kind of recognition network is well-suited for modelling short phrases from words or modelling words from subword units.

### 'Word Sequence' Model

State-of-the-Art CSR use probabilistic language models. Spoken in the language of rule based modelling, underlying the recognition network is the most simple, 'minimal' grammar:

```
Utterance = Word { Word } .
Word      = Silence | Noise | 'none'
          | 'why' | 'was' | 'nice' ...
```

resulting in the recognition network



Unlike the hard-wired network, the word sequence model describes an 'infinite' search space. Of course, this model alone is not sufficient to restrict the search space as every word sequence is allowed. Instead, the search space must be dynamically constrained by an overlaid probabilistic language model (bigram etc.) and dynamic programming with pruning and other heuristic search strategies has to be applied. The result of the search is usually a word lattice (e.g. N-best search). Here, rule based parsing is required as post-processing to select the appropriate path out of the lattice.

## 5 Chartparsing as search mechanism

Despite the recent successes of recognition approaches with probabilistic language models, still some weaknesses can be pointed out:

- Probabilistic language models are not optimal to describe linguistic regularities. (As an extreme case, a lexicon can also be thought of as a set of linguistic rules: Each word consists of a sequence of phones, sometimes also with alternative paths in case of pronunciation variants. Imagine now that the lexicon would be described as a bigram, i.e. telling only the probability of phones following one another.)

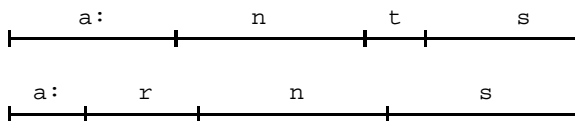
- Parsing is needed anyway to select a solution out of the word graph and in order to 'understand' the utterance. A rule-based language model is therefore there in any case.
- Solutions discarded by the recognizer don't reach the post-processing parser anymore. However, the decision criteria to discard that solution don't use the full linguistic knowledge available.
- Going through the recognition network with dynamic programming, several paths have to be kept active at each time frame. But often, those paths are active that contain similar parts. Since a probabilistic model has no knowledge of 'constituents', it cannot exploit this similarity and effort is wasted to do the same work several times.

Here, the idea to use chartparsing for CSR comes in:

- Use Chart-Parsing as search mechanism.
- Operate it on a network of phonetic units.
- Use *one* language model for both dynamic search restriction *and* linguistic 'understanding'.
- Partial hypotheses — phones, morphemes, syntactic constituents – can be found once and used many times.
- Scheduling mechanism could be used to reconsult the recognizer when other phonetic units are needed.

## 6 The Boundary Hypothesis Problem

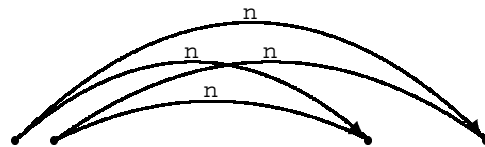
Consider a traditional probabilistic recognizer – say an HMM recognizer – analyze a piece of signal, resulting in the following two well-scored paths. (The ticks mark segmentation borders.)



Of course, both paths are well-scored because they are phonetically similar, and accordingly, equal or similar phonemes appear in both paths. However, the recognizer regards the corresponding 'units' in both paths as different:

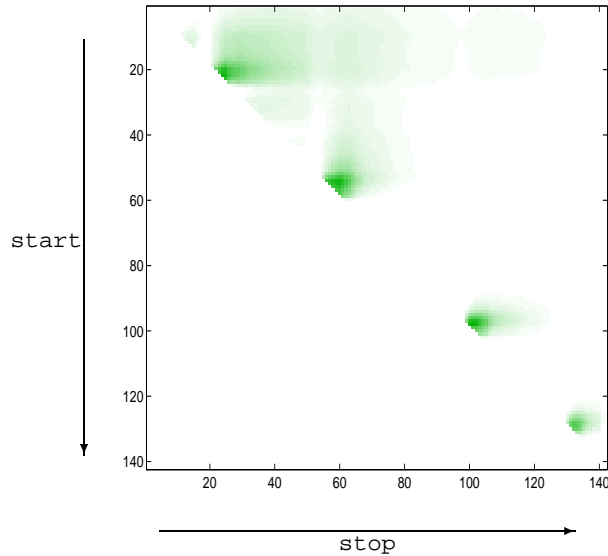
- The segmentation is different because it depends on the neighbouring hypotheses
- The scoring is different because it depends on the segmentation and – in case of context-specific phonetic unit models – it depends on the neighbouring units.

If we are to use chartparsing as search mechanism and choose the nodes of the chart to represent points in time (i.e. segment borders), then differently segmented units correspond to different hypotheses. Take the above example, where several 'n'-hypotheses with slightly different segmentations are likely to get high scores:



But isn't it actually "the same" 'n', regardless of what the neighbours happen to be? For the chart parser to exploit its abilities (find once – use many times), it would be crucial to regard all these 'n's as one unit and reuse it in many contexts.

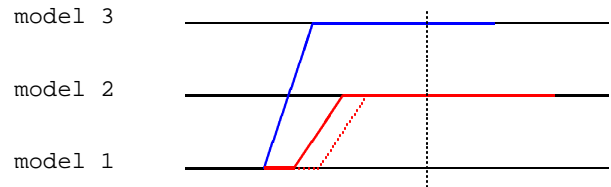
To illustrate the situation in a different way, an utterance (it says "mama papa") is analyzed in an unusual way: for each possible segment within the utterance, scores are assigned for how well the segment matches the 'a' phoneme model (HMM). The result is shown in the graphics below, where each point represents one possible segment and the colour intensity of a point at coordinates (start, stop) corresponds to the found score  $s_a(\text{start}, \text{stop})$  of a segment between the time points 'start' and 'stop'. (in the oral representation, the phonemes 'm' and 'p' are also shown in different colors.)



As expected, we see that there are 4 highly scored “coloured spots” corresponding to the 4 occurrences of the phoneme ‘a’ in the utterance, however these spots are “smeared”, meaning uncertain boundaries.

Even if we selected only a few candidate boundaries (only a few points from each “coloured spot”) we had still too many combinations to parse through. Look at just one symbol sequence but two boundary candidates between subsequent symbols. We still would get  $2^n$  different hypotheses with the same symbol sequence!

In a closer look at the segmentation problem we can see that some border hypotheses might be eliminated at an early stage. The optimal boundary (e.g. the one selected by the HMM decoding) is a function of the context, but only a narrow context (the adjacent neighbour might be sufficient) is needed to determine the optimal boundary. (Think of how big a piece of utterance you have to hear in order to determine a segment border manually!) A clever basic element detector could exploit this fact to eliminate some of the superfluous boundary hypotheses. For instance, in an N-best viterbi decoder, if two paths correspond to the same symbol sequence, it is often because the active path branches at some place but the two resulting paths then rejoin soon afterwards. (See the sketch below.) A short trace-back from the point of rejoining would suffice to detect the redundancy and to abandon the superfluous branch again.



## 7 Chart-Parsing Revised

The basic problem of using labeled speech segments as terminals of a chart parser remain, even if some superfluous segment borders (and hence segments) can be eliminated already in the decoding phase: if sub-structures once found are to be reused in different contexts, then several decoder hypotheses, all interpreting a certain spot of the utterance as the same phone, but slightly differing in segmentation (different dots within the same “coloured spot”), should be regarded as one unit when treated by the parser. However, the contextual influence on the score has to be considered as well.

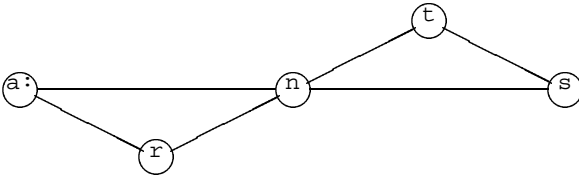
The following approach would take into account both requirements:

- A (yet to be invented) recognizer delivers only one basic element hypothesis per “coloured spot”.
- Each such hypothesis gets an ‘inherent’, context-independent score (e.g. an average over a range of boundaries).
- Basic element hypotheses that might be adjacent get connected through so called “links”. Basic Elements and the links between them form a network replacing the traditional N-best lattice.
- A link represents an “adjacency hypothesis” and carries its own score.
- Since a link puts two basic elements into each others context, the optimal border for that context can be locally determined. The link’s score can contain a correction term, helping to calculate the joint hypothesis score from the inherent scores of both participating basic elements.

This new basic element structure has to be reflected in a new interpretation of the chart:

- Basic elements become nodes of the chart.
- Links between (possibly) adjacent elements become the edges.

The following sketch shows an initialized chart with the new structure, taking the basic elements from the introductory example:



We can see that, for instance, there is only one 'n'-node in the chart, but nevertheless, both postulated paths (and two more) exist.

The scoring is integrated into the chart parsing process in a natural way: When applying the fundamental rule, the score of the new edge is calculated from the score of the parts and the score of the link involved.

## 8 Unsolved problems

The suggested ideas still have to be worked out in detail and answers to several questions are still open, e.g.:

- How can a 'coloured spot' recognizer be realized?
- What are the appropriate 'inherent' basic element scores, how should links be scored?
- How should the score arithmetic look like, i.e. what is the formula to be used during fundamental rule application, when calculating the score of the new edge from the score of the parts and the score of the link.