Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems

S.J. Young
N.H.Russell
J.H.S Thornton
Cambridge University Engineering Department

July 31, 1989

Abstract

This paper describes a simple but powerful abstract model in which connected word recognition is viewed as a process of passing tokens around a transition network. The advantages of this unifying view are many. The various apparently different connected word algorithms can be represented within the same conceptual framework simply by changing the network topology, the application of grammatical constraints is straightforward, and perhaps most importantly, the entire structure is independent of the actual underlying pattern matching technology. To illustrate the power of this conceptual model, the paper concludes by describing some work done under the UK Alvey-sponsored VODIS Project in which the Token Passing paradigm enabled the One Pass algorithm to be straightforwardly extended to include the generation of multiple alternatives and context free syntactic constraints.

1 Introduction

In a recent paper, Godin and Lockwood [1989] investigated the One Pass (OP) [Vintsyuk 1971, Bridle *et al* 1982] and Level Building (LB) [Myers & Rabiner 1981] connected word algorithms in some depth and eventually showed that when syntax constraints were applied, the two algorithms were essentially identical. Godin and Lockwood's paper is based upon the conventional formulation in which within-template matching is viewed as finding a minimum cost path through a matrix of local distances whereas between-template matching is viewed as extending a path from one template to the next. The problem with this viewpoint is that it is rather specific to DTW technology and although the OP and LB algorithms are fundamentally similar, the actual code used to implement them is nevertheless different. Thus, for example, given a Level Building DTW-based connected speech recogniser, it will not, in general, be entirely straightforward to convert it to a One Pass Hidden Markov Model (HMM) recogniser. More crucially, since within-template and between-template matching are treated differently, extending the higher level processing to include, say, more complex grammatical constraints is not trivial.

This paper describes a simple conceptual model of connected word[1] recognition based

[1]the term *word* here denotes the basic recognition unit used which may be a word but could equally well be a diphone, phoneme, etc.

on *Token Passing* within a transition network structure. With this model, which is equally applicable to DTW and HMM recognition, it will be shown that the OP and LB algorithms are just simple topological variants. Furthermore, because the model is so simple, it is easy to extend to include the generation of alternative solutions and more complex high level control mechanisms.

The paper is organised as follows. The general Token Passing model and its relationship to the various existing algorithms is described in section 2. In section 3, some extensions of the model to include context free grammar constraints and the generation of alternatives are described. Finally, in section 4, the application of the model in a working laboratory system is presented along with some experimental results on the use of multiple alternatives and various modes of grammatical constraint.

2 The Token Passing Model

2.1 Isolated Word Recognition

Consider first isolated word recognition. In the Token Passing model, each vocabulary word is represented by a *word model* which is a finite state network of the form shown in Figure 1. Associated with each pair of connected states i and j in a word model is a *transition cost* p_{ij}, and associated with each state j is a *local cost function* $d_j(t)$. The unknown speech signal is assumed to be in the usual form of a sequence of vectors $\mathbf{x}_1 \ldots \mathbf{x}_T$ and occupation of state j at time t implies that the cost of matching the associated speech vector $\mathbf{x}_t$ is $d_j(t)$.

Each possible sequence of states through the model $\mathbf{i} = i_0, i_1, \ldots, i_T$, where i_0 is the initial state, represents one possible alignment of the model with the speech signal. The total cost $S(\mathbf{i})$ of this alignment is given by

$$S(\mathbf{i}) = \sum_{\tau=1}^{T} (p_{i_{\tau-1},i_\tau} + d_{i_\tau}(\tau)) \tag{1}$$

In speech recognition, the *similarity* between a model and the unknown speech is assumed to be inversely proportional to the minimum cost alignment. Let $s_j(t)$ be the minimum cost alignment between the segment $\mathbf{x}_1 \ldots \mathbf{x}_t$ of the unknown speech and the model starting in state i_0 and ending in state j. By the principle of optimality, this cost can be computed by the recursion

$$s_j(t) = \min_i \{s_i(t-1) + p_{ij}\} + d_j(t) \tag{2}$$

thereby giving an efficient way of computing the required minimum value of $S(\mathbf{i})$, that is

$$S_{min}(\mathbf{i}) = \min_j \{s_j(T)\} \tag{3}$$

where j is any allowable final state of the word model.

All of the above is entirely straightforward and well-known. Indeed, the similarity between this model and a hidden Markov model is particularly self-evident. However, the distinction is maintained here purposefully to emphasise that token passing is not restricted to HMM's. This is discussed further in section 2.2

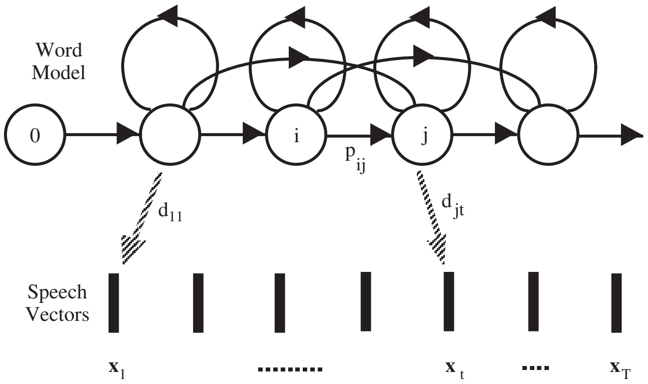


Figure 1: Structure of a Word Model

Equation 2 would normally be evaluated by representing $s_j(t)$ as a matrix as shown in Figure 2, and calculating the matrix elements column by column. This style of evaluation is the basis of the Godin and Lockwood paper. The central point of this paper, however, is that an alternative style of evaluation based on token passing leads to much simpler and more powerful generalisation.

Let each state of a word model be capable of holding a moveable *token* where each token, for the moment at least, holds a single alignment cost so that at time t, the token in state j holds the value $s_j(t)$. In terms of these tokens, the algorithm implied by equation 2 can be reformulated (somewhat informally) as follows:

Initialisation:
  Each model initial state holds a token with value 0;
  All other states hold a token with value ∞

Algorithm:
  for t:= 1 **to** T **do**
    for each state i **do**
      Pass a copy of the token in state i to all connecting
      states j, incrementing its s value by $p_{ij} + d_j(t)$;
    end;
    Discard the original tokens;
    for each state i **do**
      find the token in state i with the smallest s
      value and discard the rest
    end;
  end;

Termination:
  Examine all final states, the token with the smallest s value
  gives the required minimum matching score.

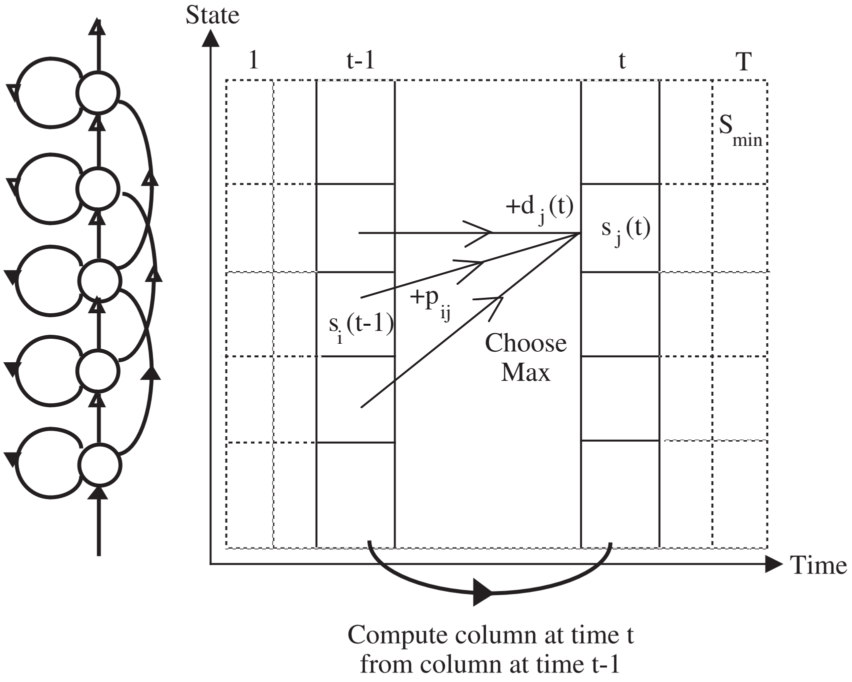


Figure 2: Matrix of Partial Alignment Costs

This token passing algorithm is illustrated in Figure 3. It corresponds to a time synchronous search strategy in which the alignments represented by the tokens are each advanced by one frame of the input for every cycle of the main loop. Notice here that the possibility of having multiple initial and final states has been allowed, this is not necessary for isolated word recognition but will be needed later.

2.2 Relation to DTW and HMM-based Pattern Matching

As noted above, the correspondence between the token passing model and hidden Markov models is obvious and direct. For a HMM with transition probabilities a_{ij} and output probabilities $b_j(\mathbf{x})$, if each transition cost p_{ij} is set equal to $-\log a_{ij}$ and each local cost $d_j(t)$ is set equal to $-\log b_j(\mathbf{x})$ then equation 2 becomes

$$s_j(t) = \max_i \{s_i(t-1) + \log a_{ij}\} + \log b_j(\mathbf{x}) \tag{4}$$

and $s_j(t)$ now denotes the maximum log probability of the model being in state j after generating the sequence $\mathbf{x}_1 \ldots \mathbf{x}_T$. Equation 4 is the standard Viterbi decoder equation for a HMM [Levinson *et al* 1983].

For DTW, each state of the word model is associated with a frame of the reference template and a topology similar to that shown in Figure 1 is assumed. In this case,the transition costs p_{ij} are set equal to fixed penalties of H for the case $i = j$ and V for the case $i = j - 2$. As shown in Figure 4, H denotes an additive horizontal penalty and V denotes an additive vertical penalty designed to penalise paths through the distance

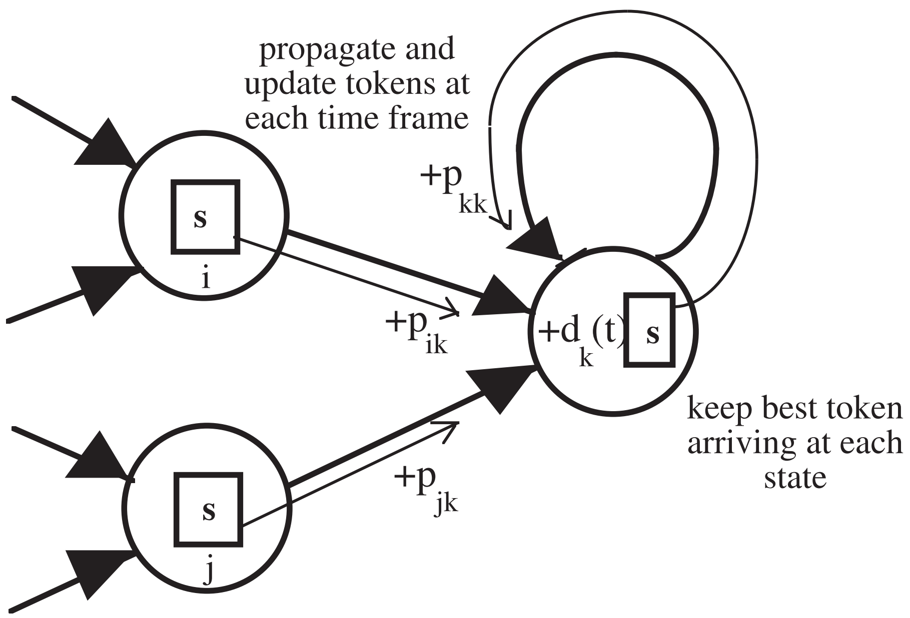


Figure 3: Basic Token Passing Algorithm

matrix which stray too far away from the diagonal. Each local cost $d_j(t)$ is set equal to $d(j,t)$ the local distance between frame j of the reference template and frame t of the unknown. Equation 2 now becomes

$$s_j(t) = \min_i \left\{ \begin{array}{ll} s_j(t-1) & +H \\ s_{j-1}(t-1) & +0 \\ s_{j-2}(t-1) & +V \end{array} \right\} + d(t,j) \tag{5}$$

and $s_j(t)$ now denotes the partial minimum accumulated distance between the first t frames of the unknown and the first j frames of the reference. Equation 5 is the standard asymmetric Dynamic Programming decision rule used in one form or another by all connected word algorithms.

The fact that both DTW and HMM-based recognition share similar formulae has been noted by others [Bridle 1984, Juang 1984]. DTW is effectively a special case of HMM recognition.

2.3 Connected Word Recognition

Given the token passing framework outlined in section 2.1, the extension to connected word recognition is trivial. The individual word models are simply connected together into a looped composite model as shown in Figure 5. The same token passing algorithm still applies although it is now effectively implementing the One-Pass (OP) algorithm. The transition costs associated with the external arcs of the looped model can be used to reflect the relative frequencies of each word occurring if this is known, or made equal otherwise.

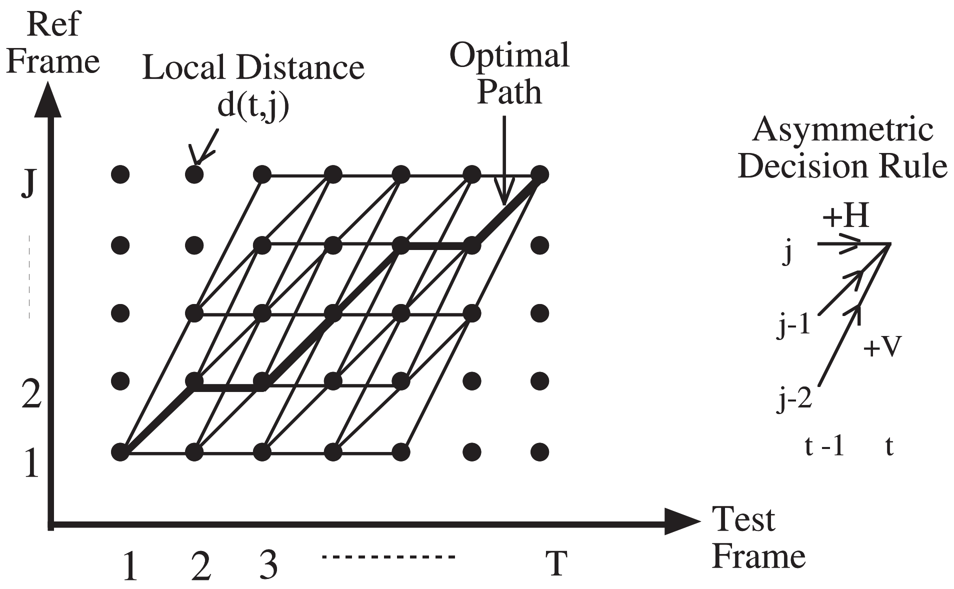


Figure 4: DTW Recognition as a Path Search Problem

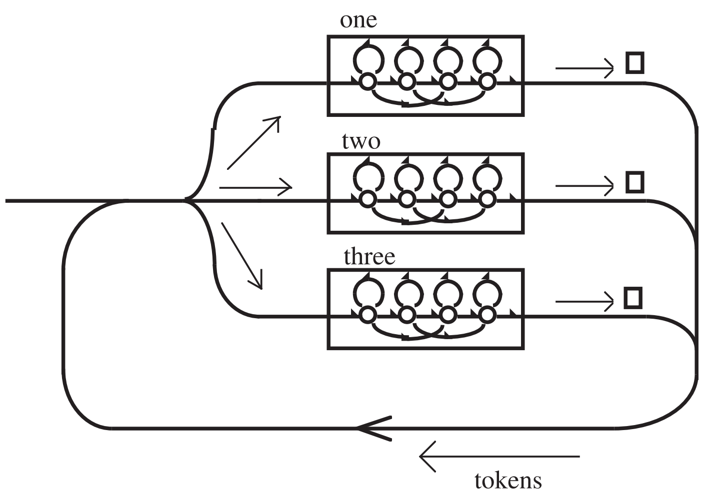


Figure 5: Connected Word Recognition by Token Passing

In practice, of course, for connected word recognition, we wish to know the identity of the best matching word sequence not just the overall alignment cost. To cater for this, the basic token passing algorithm is extended as follows. The tokens themselves are now assumed to hold a *path identifier* as well as the partial alignment cost s. This path identifier is simply a pointer to a record of word boundary information which will be called a *Word Link Record (WLR)*. At each time t, the following steps are taken *in addition to* those listed in the basic algorithm above

> **for each** token propagated via an external arc at time t **do**
>   create a new WLR containing
>   <token contents, t, identity of emitting word model>;
>   change the path identifier of the propagating token to
>   point to this new record
> **end**

This extension to the algorithm is illustrated in Figure 6. As can be seen, its effect is that during token propagation, potential word boundaries are recorded in a linked list structure such that on completion at time T, the path identifier held in the token with the minimum alignment cost can be used to trace back through the linked list to find the best matching word sequence and the corresponding word boundary locations.

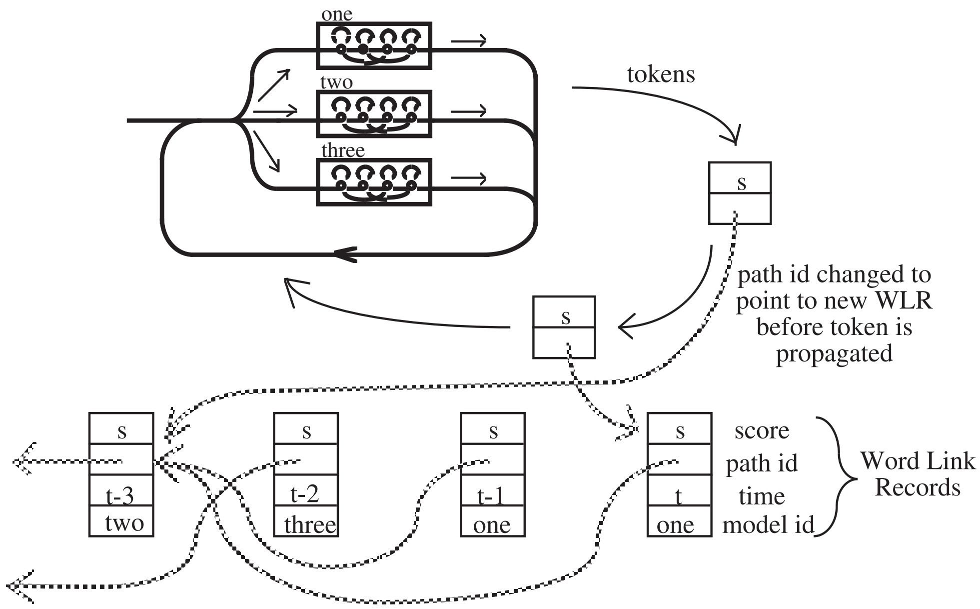


Figure 6: Recording Word Boundary Information

2.4 Relation to Existing Connected Word Algorithms

As already noted, the basic looped composite model shown in Figure 5 and reproduced again in Figure 7(a) corresponds directly to the One Pass (OP) algorithm. The Level Building (LB) algorithm is implemented by connecting the models in a left to right sequence as shown in Figure 7(b). Comparing this with the equivalent OP algorithm it is clear that the OP algorithm is much more efficient since both time and space complexity are proportional to the number of word model instances. Also, it is interesting to note that the LB algorithm as originally defined by Myers calculates all 1-length sequence matches, then all 2-length sequence matches, and so on, whereas the token passing implementation shown in Figure 7b calculates all levels in parallel. This implies that the nested *loops* in Myers' algorithm can be reordered; Godin and Lockwood [1989] show that this is indeed possible.

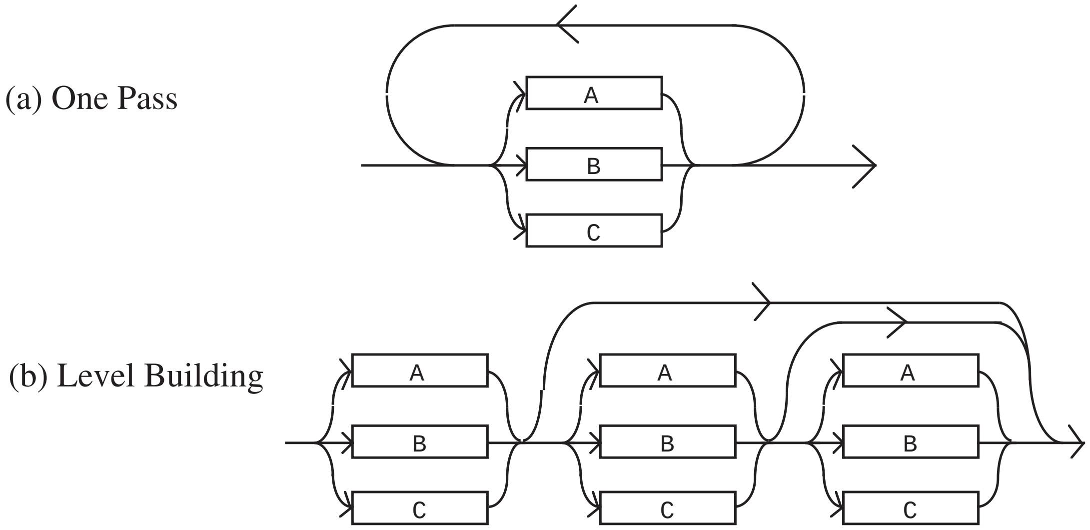
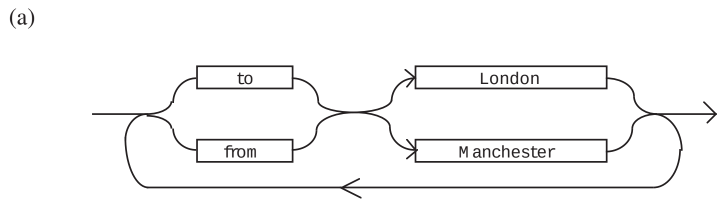


Figure 7: The One Pass and Level Building Topologies

Figures 8(a) and 8(b) illustrate that the application of finite state (or regular grammar) syntactic constraints in the token passing scheme is achieved trivially by connecting the word models together in such a way that only the required word sequences are allowed. Once syntax constraints are applied, the distinction between the OP and LB algorithms disappears since the choice of topology now depends more on the requirements of syntax than on preferences for one algorithm over another. Thus, although the syntax in Figure 8(a) could be said to be OP-like whereas that in Figure 8(b) could be said to be LB-like, the real distinction is whether the more precise constraint offered by (b) is worth the extra computation and memory implied by the 4-fold increase in word model instances. The OP and LB algorithms can therefore be viewed as different special cases of the Token Passing algorithm with Finite State syntax constraints.



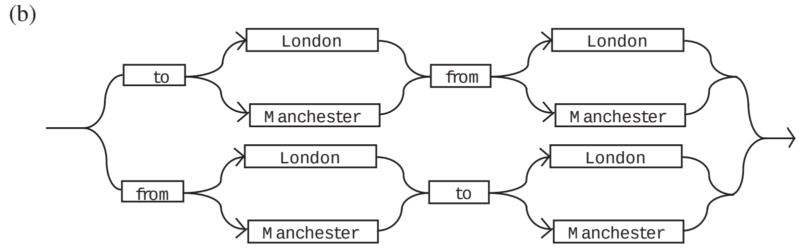


Figure 8: Styles of Syntax Control

2.5 Word Model Abstraction

In the preceding sections, it has been shown that the token passing approach allows both the OP and LB connected word algorithms to be implemented using a single algorithm simply by choosing an appropriate network topology. Furthermore, the individual word models apply equally well to both Dynamic Time Warp and Hidden Markov Model technologies.

A crucial aspect of the token passing view is that the way the individual word models contribute to the overall recognition process is determined entirely by the flow of tokens into them and all the book-keeping necessary to record word boundary positions and word identities depends only on the flow of tokens out of them. Thus, it is clear that any internal processing carried out within a word model can be separated from the between-model processing. Hence, it is possible to view word models as abstract units which take tokens as input and produce tokens as output. In the following section, this abstraction will be used to simplify the description of more complex connected word algorithms where the actual operation of the word models is represented simply by a procedure *step_word_models(t)* which means that one cycle of the main loop of the algorithm given in section 2.1 is executed.

In the token passing scheme, each token effectively represents a minimum cost partial alignment of a sequence of word models against that part of the unknown speech which has currently been seen. The propagation of a token from one word model to another corresponds to adding that further model to the sequence, and viewed like this it is clear that controlling the flow of tokens between word models corresponds to the implementation of grammatical constraints.

Thus, as shown in Figure 9, treating word models as abstract pattern matching units enables a standard interface to be defined between the high level processing components

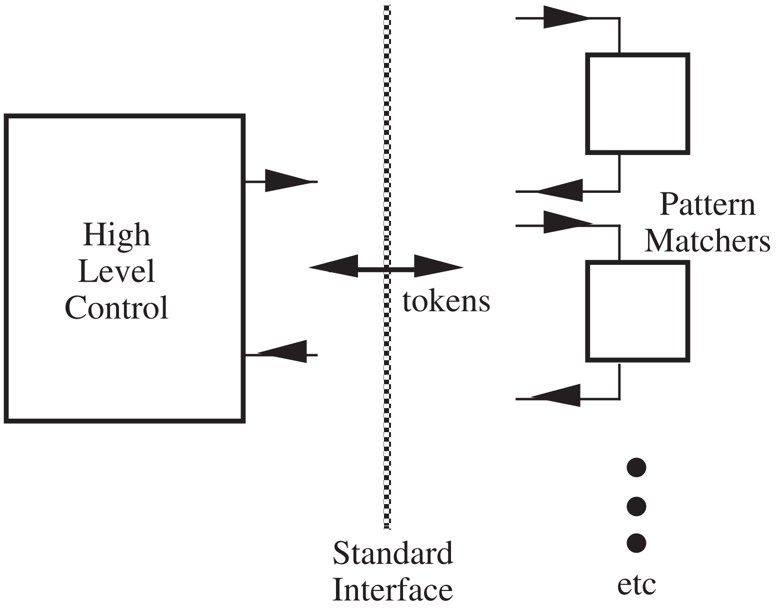


Figure 9: Separation of High Level Control from Low Level Pattern Matching

(syntax constraints, parsing, dialogue control, etc) of a system and the low level pattern matching. The existence of this standard interface allows a speech recognition architecture to be built which is essentially independent of the underlying pattern matching technology used since it does not matter how the *step_word_models* procedure is actually implemented. Externally, all that is required is that at time t each word model consumes the token at its input representing the best match between some sequence of models and the segment of input speech from 1 to $t-1$, and then outputs a token representing the best match between some sequence of models ending with that model and the segment of input speech from 1 to t. As an example, this approach was taken in the Alvey VODIS project [Young 1986, Young *et al* 1988]. The initial VODIS architecture was built around word models based on DTW matching and implemented in hardware but follow-on work has concentrated on HMM-based word models with minimal change to the overall system.

3 Extensions to the Model

In any continuous speech recognition system, proper use of syntactic constraints is vital to achieving acceptable recognition performance. These constraints are usually applied *a priori* by limiting the connections between word models as described above. The net result of applying syntactic knowledge in this way is that the system computes the best matching sequence of word models that conform to the regular grammar implied by the finite state network of word model connections. For certain types of application where the input language is small and users can be trained to adhere to it, this basic scheme works very well. However, where more ambitious grammars are needed or where users cannot be trained to speak within the prescribed syntactic forms (such as in a database inquiry system for use by the general public), there are problems.

A regular grammar covering a large input language may fail to provide sufficiently tight constraints. This is particularly true when the input language is actually specified in terms of context free rules (which are much more intuitive to the application designer) and then compiled automatically into an approximately equivalent finite state network. Since any finite state (i.e. regular) grammar whose language includes all the strings of some desired context free language must in general also include many strings not in the context free language, compilation will inevitably imply a weakening of constraints. Thus, there may be some advantage in being able to apply context free constraints directly within connected word algorithms.

However, systems which rely on strong *a priori* syntactic constraints are not robust against speakers who stray outside of the prescribed grammar. An alternative approach is therefore to extend the basic connected word algorithm to generate a lattice of word alternatives rather than the single best matching sequence. Syntactic knowledge can then be applied in an *a posteriori* parsing phase rather than by *a priori* constraints giving more flexibility in the design of error recovery strategies.

In this section, the implementation of direct context free constraints and the generation of alternatives will be described. Although these two mechanisms appear to be mutually exclusive, this is not, in fact, the case. Generating multiple alternatives in combination with strong *a priori* syntactic constraints simply means that all the alternatives will be grammatically correct. Subsequent semantic/pragmatic processing may, however, still be able to benefit from the availability of alternatives. Furthermore, with multiple alternatives, it is important to be able to focus the generation process on those parts of the utterance that are most likely to contain important information. This can be done easily in conjunction with a direct context free constraint mechanism since each non-terminal of the grammar dominates a definite substring of terminals. Hence, a specification of the number of alternatives to allow can be associated with each non-terminal. There is no corresponding entity in a finite state grammar making it much more difficult to control the generation alternatives.

In fact, the optimal combination of the two mechanisms seems to be to use a weakened set of context free rules for the *a priori* constraints, generate multiple alternatives and then use the full set of rules for *a posteriori* parsing. This is discussed further in the Section 4.

3.1 Direct Context Free Grammar Constraints

For the direct context free grammar constraint mechanism described here, it is assumed that the allowed input language is defined by a set of extended BNF production rules [Wirth 1976]. In order to apply context free grammar constraints within the token passing scheme, these grammar rules are compiled into a set of linked syntax networks of the form illustrated by Figure 10.

The nodes of each syntax network are of three types: links, terminals, and non-terminals. Link nodes are used to store tokens and are the points where recognition decisions are recorded. Terminal nodes correspond to word models and non-terminal nodes refer to separate *sub-syntax* networks representing the RHS of the corresponding grammar rule. Neither word models nor subsyntax networks are shared and hence there is a unique instance of a word model or sub-syntax network for every reference to a terminal

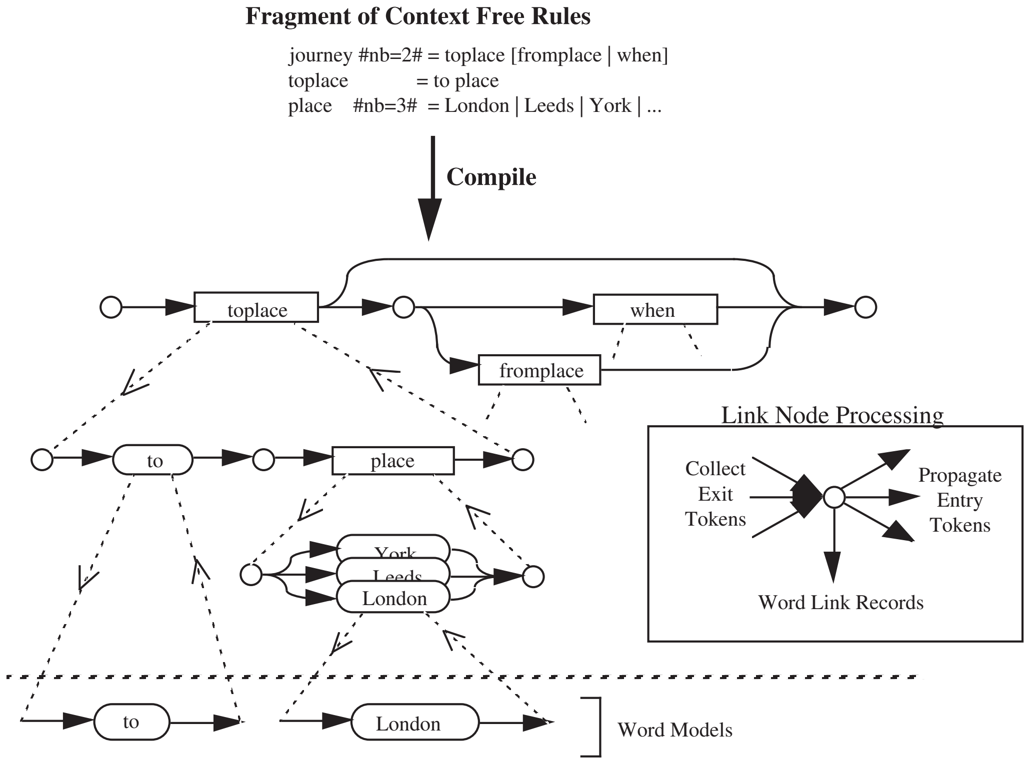


Figure 10: Implementing Direct Context Free Constraints

or non-terminal in the rules. Notice that this implies that recursive rules, whose expansion would require infinitely many instances, cannot be allowed. The extended BNF formalism for the grammar rules, however, allows loops and branches to be included and these can be handled within the constraint scheme without any difficulty so there is little real loss of expressive power.

The three types of node are combined in such a way that every arc connects either a terminal or a non-terminal to a link node, or *vice versa*. Each syntax network has exactly one entry, one exit and zero or more internal link nodes. Every terminal and non-terminal node will have exactly one arc leading into it, whereas each link node may have any number. Link nodes can thus be viewed as filters which remove all but the best (ie lowest cost) tokens passing through them.

Given the above representation of the context free grammar constraints, implementation of connected word recognition using the token passing framework is relatively straightforward. The basic idea is that tokens propagate through the networks just as in the finite state case, however, when a token enters a non-terminal node it is transferred down to the entry node of the corresponding sub-syntax, and when a token enters a terminal node it is transferred to the entry node of the corresponding word model. Similarly, when tokens reach the exit of word models and sub-syntaxes, they are transferred back up to the node which *owns* them. This flow of tokens is illustrated in Figure 10. The dotted line across this figure corresponds to the Standard Interface shown earlier in Figure 9. The flow of tokens across this interface will be such that the sequence of word models visited by any token will conform to the context free grammar rules.

In more detail, the algorithm for connected word recognition with context free grammar rule constraints is as follows:

Initialisation:
    Store a token with cost $s = 0$ in the entry node of the *top_level* syntax;
    Store a token with cost $s = \infty$ in all other link nodes

Algorithm:
    for t:= 1 **to** T **do**
        Propagate_entry_tokens(*top_level*);
        Copy tokens from all terminal nodes into the entry nodes
        of the corresponding word models;
        Step_word_models(t);
        Copy the tokens from exit nodes of all word models back to
        the corresponding terminal nodes;
        Propagate_exit_tokens(*top_level*)
    end;

Termination:
    The token stored in the exit node of the *top_level* syntax
    gives the required best matching word model sequence.

The two procedures which handle token propagation *Propagate_entry_tokens* and *Propagate_exit_tokens* are recursively defined as follows:

```
Propagate_entry_tokens(s: syntax):
    for each link node l in syntax s do
        for each node n following l do
            Copy token in l into n;
            if n is non-terminal then
                Copy token in n to entry node of
                corresponding sub-syntax;
                Propagate_entry_tokens(n)
            end
        end;
        Store a token with cost s = ∞ in l
    end;

Propagate_exit_tokens(s: syntax):
    for each link node l in syntax s do
        for each node n preceding l do
            if n is non-terminal then
                Propagate_exit_tokens(n);
                Copy token in exit node of sub-syntax n back to
                corresponding non-terminal node
            end;
            Filter_tokens(n, l)
        end;
        Record_decisions(l)
    end;
```

If the context free grammar is *stochastic* then each arc of the syntax networks will carry a transition cost exactly the same as within the word models. The token propagation procedures must then be modified to add these costs to the tokens as they are propagated.

The procedure *Filter_tokens* is called once for each node n preceding a given link l. It simply examines the token in l and if the cost of the token in l exceeds that of the token in n then the token in l is replaced by the token in n. The procedure *Record_decisions* is called once for each link l after all token filtering has been completed. It simply creates a WLR for the token in l (see Section 2.3) and changes the path identifier of that token to point to the new WLR.

On completion of the above algorithm, the exit node of the top level syntax will hold a token whose path identifier points to a WLR corresponding to the last word in the recognised sequence. Tracing back from this WLR to preceding WLR's via their path identifiers then yields the actual recognised word sequence.

3.2 Generating Multiple Alternatives

Obtaining multiple alternative word matches in the basic Token Passing scheme is achieved by recording the N-best (i.e. lowest cost) tokens emitted at each syntactically distinct word boundary instead of just the best. Note, however, that only the best token is actually propagated as before. Since the total cost at each potential word boundary is recorded in the Word Link Records, the cost for each individual word n can easily be determined by subtracting the cost at word boundary $n-1$ from the cost at word boundary n. On completion of the recognition processing, the WLR's can be converted to a lattice of alternative word matches and then processed by a *Chart* parser [Winograd 1983]. This process is illustrated in Figure 11.

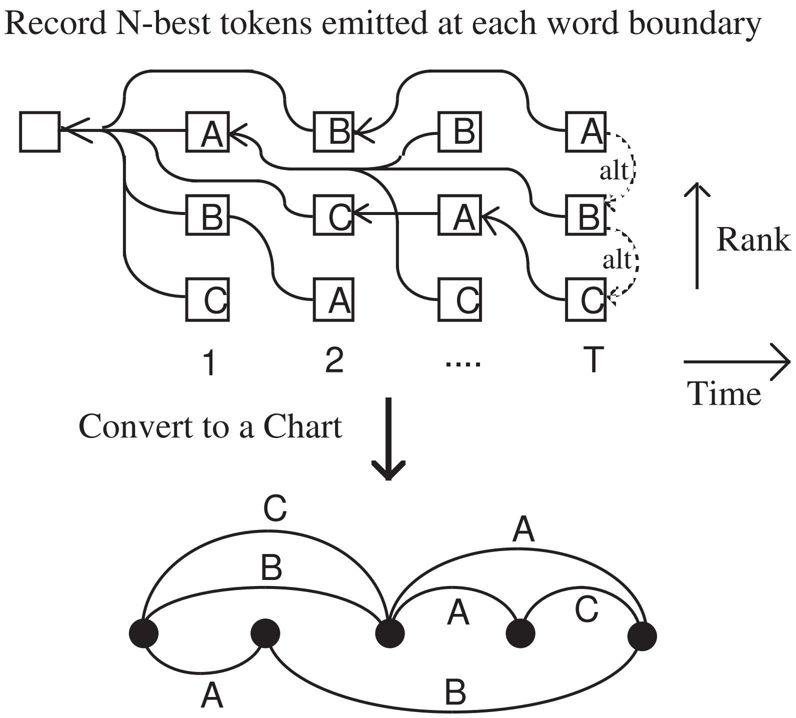


Figure 11: Recording Multiple Word Matches

To generate multiple alternatives with the system of direct context free constraints described above, it is only necessary to modify the *Filter_tokens* and *Record_decisions* procedures. The *Filter_tokens* procedure now maintains a list of up to N tokens for each link node l. For each terminal or non-terminal node n preceding l, the token in n is only added to the list for l if there are not already N lower cost tokens stored in l. After all processing by *Filter_tokens* for link node l has been completed, the *Record_decisions* procedure creates a WLR for each token stored in l linking them together by an additional *alt* field. It then discards all tokens except the best and changes the path identifier of that best token to point to the corresponding WLR. The number of tokens saved at a link node can be set individually for each corresponding rule enabling the generation of alternatives to be focussed on the semantically rich regions of the input. Thus, for example, in Figure 10 the parameter *nb* attached to the non-terminals *journey* and *place* indicate that the best two alternatives for a journey and the best three alternatives for a place should be saved.

The Word Lattices generated by the above procedures do not necessarily include the N-best matching sequences since the Token Passing system (in common with the conventional One Pass algorithm) is such that a word match is available for every possible end-point but not every possible start-point. This is an intrinsic feature of the algorithm which is essential to its efficiency. When tokens enter the same node, they compete and only the best is propagated. If the actual N-best matching sequences are required then word matches must be computed for every possible start and end point. The N-best sequences can then be found by dynamic programming [Young 1984]. Such a procedure is, however, computationally expensive. Although sub-optimal, the procedure described here is very cheap to compute and, in practice, does appear to generate *useful* alternatives.

4 An Example Application

The preceding sections have described the Token Passing approach to the design of connected word recognition systems and have shown how it can be extended to generate multiple alternatives. In this section, an application of Token Passing in the design of a Voice Operated Database Inquiry System (VODIS) will be briefly described and some experimental results presented to indicate the relative advantages of the various possible combinations of *a priori* and *a posteriori* syntactic processing.

4.1 The VODIS Project

The UK Alvey-sponsored VODIS project was a three year collaborative venture between British Telecom, Logica and Cambridge University Engineering Department. The project was not concerned with recognition algorithm development *per se*, rather it was concerned with those aspects of voice-based system design which arose from its intended use as a telephone-based conversational question/answer system for the general public. The example application area tackled was that of Train Timetable Inquiries.

Figure 12 shows a block diagram of the final VODIS architecture. The central control of the VODIS system resides in a frame-based Dialogue Controller (DC) [Young & Proctor 1989]. The DC operates in cycles asking the user questions and processing the replies until his or her query is fully understood.

The system's knowledge of syntax and semantics is in the form of Context Free Grammar rules stored in the Rule Database. At the start of each recognition cycle, the DC activates two distinct contextually relevant rule subsets in the Rule Database. One of these subsets is used to apply *a priori* syntactic constraints to the subsequent pattern matching and the other is used for the *a posteriori* parsing.

The actual speech recognition takes place at two levels: word level and phrase level. At the word level, the input speech is matched frame synchronously with the word models using the DTW variant of the basic Token Passing scheme described in section 2. To handle interword pauses, each word model has been augmented to include an optional preceding looped silence state. Also, each word model has a fixed cost per frame *Wildcard* connected in parallel with it to place an upper limit on the cost of a match. At the phrase level, the flow of tokens between word models is controlled by transition networks built automatically from the grammar and multiple alternatives are recorded as described in section 3.

Once the input speech has been consumed, the recorded word boundary information in the WLR lists is converted to a Chart of word alternatives and processed by a Chart Parser. This Chart Parser is a modified form of the standard Bottom-Up algorithm

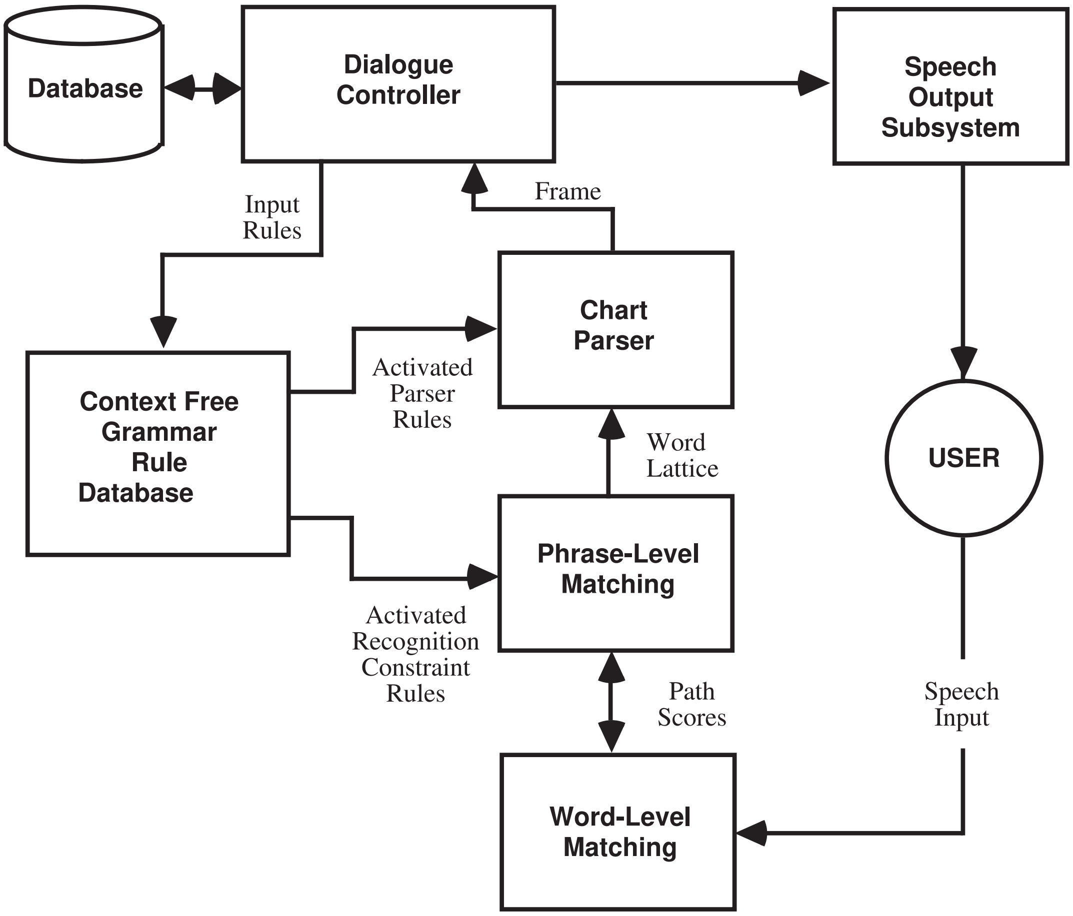


Figure 12: VODIS II System Architecture

[Winograd 1983] in which the concept of an *active edge* and an *agenda* has been dispensed with. Instead, inactive edges are built and vertices are processed in a fixed right to left order. When a complete parse is required (i.e. all syntactically valid interpretations are wanted), this modified algorithm is functionally identical to the standard algorithm but executes an order of magnitude faster and uses less memory.

When the full chart has been built, it will consist of a large number of arcs (or edges in Chart Parsing terminology) spanning segments of the input. The Parser initially assigns a cost to each terminal edge equal to the matching cost computed by the corresponding word model. Each higher level spanning edge is then assigned a score equal to the sum of the edges that it subsumes. That is, for an edge labelled with non-terminal X which subsumes a sequence of edges $Y_1 Y_2 \ldots Y_n$ corresponding to the syntax rule $X = Y_1 Y_2 \ldots Y_n$ [see Figure 13(a)] the cost S_X assigned to edge X is given by

$$S_X = \sum_{i=1}^{N} S_{Y_i} \tag{6}$$

In practice, the situation shown in Figure 13(b) is the more usual case where matches for only a subset of the constituents of X have been found. For cases such as this, a constant cost per input frame ρ is added for all frames for which there are no spanning arcs, that is, if only the edges Y_j to Y_k have a cost below the Wildcard threshold then

$$S_X = \sum_{i=j}^{k} S_{Y_i} + \rho \left(T(X) - \sum_{i=j}^{k} T(Y_i) \right) \tag{7}$$

where $T(A)$ denotes the length of the edge labelled with A.

The value of ρ is chosen such that it is greater than the average cost per input frame for a correct word match and less than that for an incorrect one. This method of dealing with unexplained segments of the input is one of many that have been investigated and, as well as being the simplest, it is also the most effective.

Once costs have been assigned to all edges in the chart, the lowest cost phrase structures built in the parsing process are extracted. In doing this, the parser does not insist that an edge spans the entire input. For example, in Figure 13(c) both W and X are possible syntactic interpretations of the input. The cost for X is computed using equation (6), that is $S_X = \sum_{i=1}^{5} S_{Y_i}$, whereas the cost for W is computed using equation 7 with $T(X)$ set equal to the total number of frames in the input M, that is $S_X = \sum_{i=2}^{4} S_{Y_i} + \rho \left(M - \sum_{i=2}^{4} T(Y_i) \right)$. This procedure ensures that syntactically valid segments of the input which match well but which cannot be made part of a fully spanning syntactic interpretation are not ignored.

Finally, semantic interpretation is performed by stripping the final parse tree of semantically irrelevant branches and then matching it against *slots* in the frame-based Dialogue Controller.

4.2 Effects on Performance of Syntactic Constraints

The separation of the rules in the VODIS system has allowed the effects on recognition performance of the balance between *a priori* and *a posteriori* processing to be investigated. The first version of VODIS and the one used for full-scale user trials [Cookson 1988]

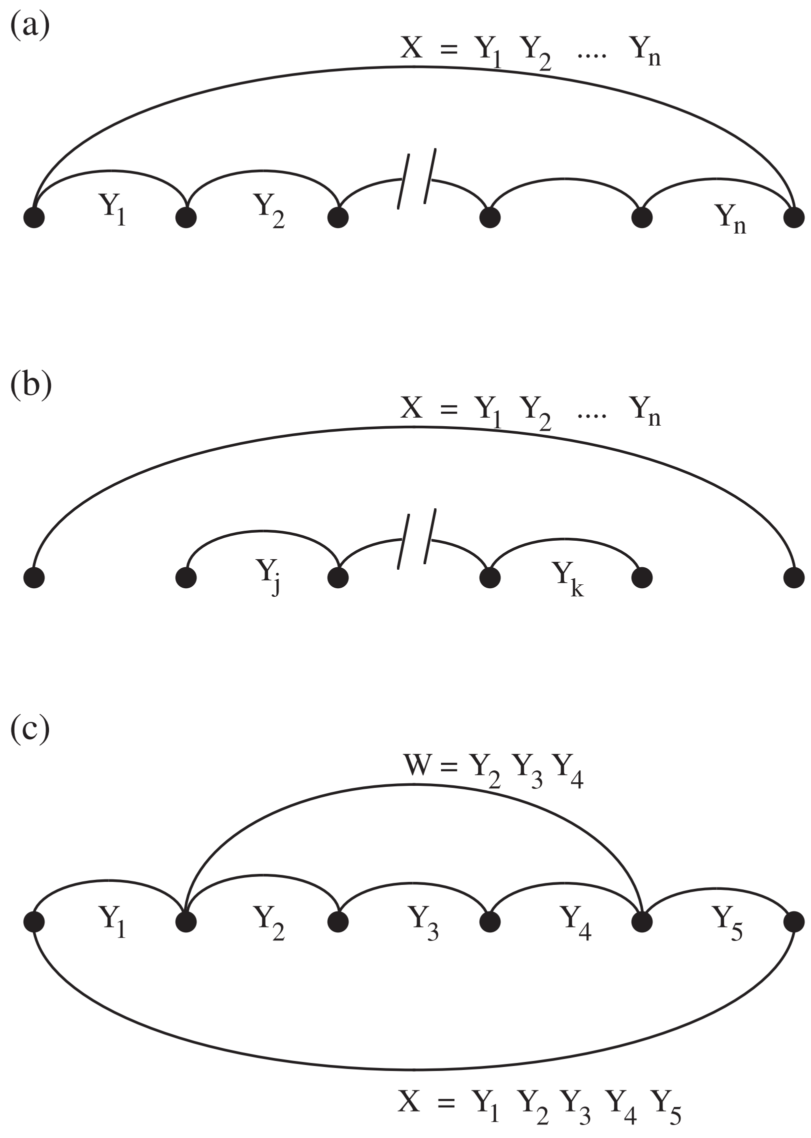


Figure 13: Example Edge Configurations in Chart

employed conventional finite state syntax constraints. This system worked very well with users who knew the preprogrammed syntax rules but performance dropped rapidly with naive untrained users and the system was forced to adapt rapidly to an isolated word style of interaction. The later version described above allows the rules used for *a priori* constraints to be weakened whilst still retaining a full set of grammar rules for the *a posteriori* parsing and semantic interpretation.

To investigate the effects of syntactic constraints, the performance of the system was measured with the following different constraint mechanisms

FState Finite State *a priori* constraints derived from the context free task grammar as used in the original VODIS system.

CF Context Free *a priori* constraints derived directly from the task grammar.

WeakCF Context free constraints derived from a set of weak rules, consistent with the task grammar, which enforce function word-content word ordering but no higher level structure. A *Wildcard* is placed between phrasal rules to account for unknown words.

Null No *a priori* constraints.

In all cases, the parsing and semantic interpretation were identical. In the FState case, no alternatives were generated and only fully spanning syntactic interpretations of the input were allowed. In all other cases, 3 alternatives were recorded at all semantically important nodes and 2 alternatives were recorded otherwise.

The test material consisted of 20 syntactic sentences (the S set) which conformed to the task grammar and 20 non-syntactic sentences (the NS set) which did not conform but were semantically similar. The latter NS set were extracted from transcripts of real British Rail enquiries, hence although they were grammatically *unusual*, they were nevertheless realistic. Note also that as well as being ungrammatical, 40% of the vocabulary used was unknown to the system.

The results are presented in Tables 1 and 2 for two speakers IC and ST who read each of the sentences as realistically as possible. The DTW recogniser was trained individually for each speaker. The performance is given in terms of *slot recognition rate* rather than word recognition rate since this is the only meaningful quantity which can be measured for this type of system. A full definition of a *slot* lies beyond the scope of this paper since it is intrinsically recursive and closely tied to our method of Dialogue Control implementation. However, a slot corresponds roughly to a single item of information in the utterance, hence, by way of example, the sentence "I want to leave from Peterborough at about nine to travel to York" has 5 slots:

1.	leave (	– as opposed to *arrive*
2.	about (	– as opposed to *before* or *after*
3.	time(digit(nine))))	– the time
4.	fromplace(Peterborough)	– departure place
5.	toplace(York)	– destination place

Spkr	Test	FState	CF	WeakCF	Null
IC	S	82	85	82	72
ST	S	93	92	92	90
IC	NS	10	34	45	41
ST	NS	18	31	54	51

Table 1: Percentage Slot Recognition Rate (Best alternative only)

Spkr	Test	CF	WeakCF	Null
IC	S	92	89	82
ST	S	94	97	97
IC	NS	38	49	48
ST	NS	33	57	54

Table 2: Percentage Slot Recognition Rate (Best 2 alternatives)

In all cases, recognition rate is calculated as $N_c/(N_t + N_i).100$ where N_c is the number of slots correctly recognised, N_t is the total number of slots and N_i is the number of insertion errors.

From Table 1 it can be seen that for the syntactic S set of sentences, there is little difference between conventional finite state (FState) constraints and direct context free (CF) constraints. For the non-syntactic NS set, however, there is a dramatic drop in performance for both, with the FState case being the worst. In this case, the better performance of the CF case is mainly due to the more sophisticated parsing made possible by the availability of alternatives. When the *a priori* constraints are weakened as in the WeakCF case, there is little change in performance on the S set but a considerable improvement in the NS case. For no constraints at all as in the Null case, performance is less sensitive to ungrammaticality but it always falls below that of the WeakCF case. Hence, the strategy of using weak *a priori* syntactic constraints in conjunction with full *a posteriori* parsing on a lattice of alternative word matches appears to offer the most robust strategy for this type of conversational speech system.

Table 2 shows the effects of including the second best global interpretation of the input in the slot matching algorithm. In this case, the same overall pattern of results emerges but there is a small but significant improvement in every case. Thus, depending on the design of the Dialogue Controller, the generation of semantic alternatives may also be useful.

5 Conclusions

This paper has described a simple conceptual model of connected speech recognition based on Token Passing. The advantages of this approach are firstly that it allows low level pattern matching to be abstracted in such a way that there is a clean interface between word level and phrase level processing. This has the great practical benefit of enabling the higher levels of a speech recognition system to be independent of the actual pattern matching technology used. The second main advantage of the Token Passing view is that it is very straightforward. In our view, most of the characteristics of connected word algorithms discussed by Godin and Lockwood [1989] are largely self-evident from the Token Passing viewpoint. Furthermore, reasoning about complex extensions is considerably simplified and as examples of these at the phrasal level, the incorporation of direct context free grammar constraints and semantically focussed multiple alternative generation have been described.

Finally, some performance issues with regard to the use of syntactic constraints have been presented. Using the VODIS system as a model, the relative effects of tight and weak *a priori* syntax constraints have been investigated. Where users are likely to use input forms whose syntax lies outside of the preprogrammed task grammar, performance drops dramatically with conventional tight *a priori* constraints. However, a combination of weak *a priori* syntactic constraints and full *a posteriori* parsing on a lattice of alternative word matches appears to be much more robust. The availability of semantic alternatives gives a smaller performance improvement but may still be worthwhile if the Dialogue Controller can be designed to exploit it.

Acknowledgement

The VODIS Project was funded by the UK Alvey Directorate and was a collaborative venture between British Telecom Research Laboratories, Logica UK and Cambridge University Engineering Department. The assistance and support of the cooperating partners is gratefully acknowledged.

References

[Bridle 1984] Bridle JS. *Stochastic Models and Template Matching: some important relationships between 2 apparently different techniques for ASR.* Proc IOA Autumn Conf, Vol 6, Pt 4.

[Bridle *et al* 1982] Bridle JS, Brown MD, Chamberlain RM. *An Algorithm for Connected Word Recognition.* Proc ICASSP, pp899-902, Paris, France.

[Cookson 1988] Cookson S. *Final Evaluation of VODIS.* Proc 7th FASE Symposium (Speech 88), pp1311-1320.

[Godin & Lockwood 1989] Godin C, Lockwood P. *DTW Schemes for Continuous Speech Recognition: a Unified View.* Computer Speech and Language, Vol 3, No2, pp169-198.

[Juang 1984] Juang B-H. *On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition - A Unified View.* AT and T Technical Journal, Vol 63, No 7, pp1213-1243.

[Levinson *et al* 1983] Levinson SE, Rabiner LR, Sondhi MM. *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition.* BSTJ, Vol 62, No 4, pp1035-1074.

[Myers & Rabiner 1981] Myers CS, Rabiner LR. *A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition.* IEEE Trans ASSP, Vol 29, No 2, pp284-296.

[Vintsyuk 1971] Vintsyuk TK. *Element-Wise Recognition of Continuous Speech Composed of Words from a Specified Dictionary.* Kibernetica, Vol 7, No 2, pp133-143.

[Winograd 1983] Winograd T. *Language as a Cognitive Process: Volume 1.* Addison-Wesley, pp116-127.

[Wirth 1976] Wirth N. *Algorithms+Data Structures = Programs.* Prentice-Hall, Series in Automatic Computation, Englewood Cliffs, New Jersey.

[Young 1984] Young SJ. *Generating Multiple Solutions from Connected Word DP Recognition Algorithms.* Proc IOA Autumn Conf, Vol 6, Pt 4, pp 351-354.

[Young 1986] Young SJ. *Designing a Conversational Speech Interface.* Proc IEE, Vol 133, Pt E, No 6.

[Young & Proctor 1989] Young SJ, Proctor CE. *The Design and Implementation of Dialogue Control in Voice Operated Database Inquiry Systems.* Computer Speech and Language, in press.

[Young *et al* 1988] Young SJ, Russell NH, Thornton JHS. *Speech Recognition in VODIS II.* Proc ICASSP, Vol 1, S10.9, pp441-445, New York.