Language Understanding: a Procedural Perspective *

Ruth Kempson
Department of Linguistics
SOAS, University of London
e-mail: rk@soas.ac.uk

Wilfried Meyer Viol
Department of Computing
Imperial College, University of London
e-mail: wm3@doc.ic.ac.uk

Dov Gabbay
Department of Computing
Imperial College, University of London
e-mail: dg@doc.ic.ac.uk

Abstract

In this paper we introduce the data structures and process structure of an incremental parser which reflects the underspecified nature of the information encoded in the NL string. We show how this parser can deal with so called crossover phenomena in a way other systems cannot. The the parser is formulated within the LDS framework and constructs parse trees with the help of a modal logic for finite trees. Wh-expressions are interpreted as constituents with underspecified tree location.
The dynamics of the parser is formulated in terms of transition system in which the transitions between the parser states are effected by deduction rules.

1 Introduction

In this paper we will introduce the data structures and the process structure of a natural language parser which incrementally creates a (set of) labelled logical form(s), possible interpretation(s), while traveling through a NL string in a left-to-right, word-by-word fashion. The underlying aim is to model the process of understanding reflecting at each step the partial nature of the information encoded in the string and the ways in which this information is enriched by mechanisms which fix some particular interpretation. We describe an application for the model dealing with the interaction of *Wh* expressions and anaphora.
The compilation of logical formulas interpreting NL strings is formalized within an LDS framework ([Gab90]) in which *labels* guide the parser in the goal-directed process of constructing an *ordered tree of declarative units*. Declarative units consist of pairs of *sequences of labels* followed by a *content formula*. The *formula*

*This research was supported by the UK Engineering and Physical Sciences Research Council under grant reference GR/K67397,

of a declarative unit represents the *content* of the words supplied in the course of a parse. The *labels* annotate this content with linguistic *features* and *control* information guiding the direction of the parse process. Declarative units are represented as finite sets of formulas. In the course of a parse the content and number of these *feature sets* grows incrementally. Moreover, these sets implicitly label an *Ordered Tree Structure*, the syntactic tree of the representation under development: that is, one set of feature specifications can be *daughter* or *sister* to another set. This tree structure is formalized within a *Modal Logic of Finite Trees* (eg. [BlaMey94], [Kra95]) in which modalities refer to relations within trees. For instance, $\langle d \rangle \phi$ holds at a tree node if ϕ holds at a daughter node.
Dynamically, the formal model is the parser as a *transition system*. The *states* of the transition system correspond to *state descriptions* of the parser. The *transitions* of the system correspond to *inference steps between state descriptions*. In this sense we represent parsing as *deduction*. The execution of the transitions is delegated to so-called goal-directed *tasks*. A task contains a record of what has been achieved on the way to the goal of the task, and what remains to be done. Passing along the string these tasks start by being *declared* (nothing has been achieved yet) and incrementally move to being *satisfied* (nothing remains to be done). In this sense our parser resembles the one of [Mil94]. At every point in the parsing process the information gathered up to that point is captured in a sequence of *task states*. A task may spawn new (sub)tasks. The *goal* of the entire procedure is the satisfaction of the top level task with goal type t, at the last word of a string. The steps involved in the satisfaction of such tasks include steps of type deduction (modus ponens), but, unlike categorial grammars (eg [Mor94]), such familiar deductive steps form but one type of transition from task state to task state. At the conclusion of a successful parse the satisfied tasks can be seen to label a (parse) tree.

2 Data Structures

The object of our parser is the compilation of logical formulas. This determines the data structures required. There are four of them: *Content Formulas*, *Declarative Units*, *Task States*, and *Parse States*. **Content Formulas** essentially represent the contribution to *content* of the words in a string. The language of these formulas is a term logical one with variable binding term operators like ϵ and τ to accommodate quantification. They constitute the *formula part* of declarative units. An example

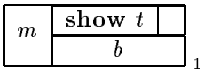
$$(a) \qquad Man(\epsilon x Man(x)) \wedge Walk(\epsilon x Man(x))$$

representing "a man walks". For more information about variable binding term operators and motivation for their use in linguistics, see [Mey95], [MeyKKG96]. The logical formula to be compiled for a given string consists of a labelled content formula of semantic type t. The labelled formulas, **declarative units** (DU's), are constructed in the course of the parse. So at any moment in the process we

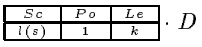
must be able to describe the *partial declarative units* present. This description consists of a set of DU-formulas. An example where a is the content formula above

$$(b) \qquad \{Tn(m), Ty(t), Tense(p), Fo(a)\}$$

The goal of the parsing process is to construct a declarative unit of type t using all the words of the string. The construction of a declarative unit is delegated to a *(sub)task*. A (sub)task can be in one of three **task states**: *Declared*, the task is declared but no material has been collected to complete the task. *In progress*, DU-formulas have been collected but not enough to construct a labelled formula of the required type. And finally *Satisfied*, when enough material has been collected to construct a formula of the required type. An example of a satisfied task state for b the declarative unit above



A **Parse State**, finally, is then completely described by a sequence of task states D, the position of the reading head in the string, and a pointer to the task presently under consideration.



The following paragraphs describe these data structures in more detail

2.1 Content Formulas

Definition 1 Language
Terms and Formulas of the language L_C for a non-empty set C of quantifier operators are built from individual variables in V, meta variables M, predicate variables in $\mathcal{P}$, individual constants in A, and predicate constants in P as follows

1. the set T_C of L_C-*terms* is defined by
   (a) all elements of A or M are in T_C
   (b) if $x \in V$, $a \in A$, $c \in C$ and $\phi[a/x]$ in $FORM_C$, then $(cx, \phi) \in T_C$.
2. the set Λ_C of *lambda terms* of L_C is defined by
   if $X \in V$, $a \in A$, $\phi[a/X] \in FORM_C \cup T_C$ then $\lambda X.\phi \in \Lambda_C$.
   If $X \in \mathcal{P}$, $a \in A$, $\phi[a/X] \in FORM_C \cup T_C$ then $\lambda X.\phi \in \Lambda_C$,
3. the set $FORM_C$ of L_C-*formulas* is defined by
   (a) if $t_1, \ldots, t_n \in T_C$, and P an n-place predicate of L. then $Pt_1 \ldots t_n \in FORM_C$

(b) if $\phi, \phi' \in FORM_C$ then $\phi \# \phi' \in FORM_C$, where $\# \in \{\wedge, \vee, \leftrightarrow, \rightarrow\}$.

Notice that λ may bind variables within the scope of elements of C but not vice versa. Moreover in elements of T_C and $FORM_C$ there occur no free variables.

Example 1 We will represent the content of a word like *some* by

$$\lambda P(\epsilon x, P(x))$$

It is a function requiring an instance of (the type of) P (for instance, $\lambda y \cdot Man(y)$) to become a complete object (of type e), i.e., *epsilon* terms.

$$(\epsilon x, Man(x))$$

2.2 Declarative Units

Declarative units consists of pairs of *sequences of labels* followed by a *content formula*.

$$\underbrace{\langle l_1, \ldots l_n \rangle}_{\text{Sequence of labels}} \quad : \quad \underbrace{\phi}_{\text{Content Formula}}$$

These declarative units constitute the components of the representation to be constructed. The *formula* of a declarative unit is the side representing the *content* of the words supplied in the course of a parse. The *labels* annotate this content with linguistic *features* and *control* information guiding the direction of the parse process.
In the course of a parse declarative units are built. In order to be able to describe growth and construction of these units, they will be represented as finite sets of *atomic DU-formulas*

$$\{C_1(l_1), \ldots, C_n(l_n), Fo(\phi)\}.$$

The *label categories* $C_1, \ldots C_n$ can be seen as *feature dimensions*. $C(l)$ states the association of feature dimension C with feature value l. In fact, the formula $C_i(l)$ is an atomic formula interpreted on a model over $\mathcal{L}$. We recognize two special feature dimensions:

- The *type* dimension Ty with values from $\langle Lab_{Ty}(e, cn, t), \rightarrow, \cdot \rangle$ consisting of types over e, cn and t. This feature determines the combinatorial properties of the declarative unit.
- The *Tree Node* dimension Tn with values chosen from the node set Lab_{Tn} of an *ordered linked tree* $\mathcal{T} = \langle Lab_{Tn}, \leq, <, L \rangle$ (where $\leq$ is the *dominance* relation, $<$ the *precedence* relation, and L the *link* relation). On these trees the relations R_u, R_d, R_l, R_r, and R_L are defined: *mother-of, daughter-of, left-sister-of, right-sister-of and linked-to relations*. We will also consider the reflexive and transitive closures R^*, $i \in \{u, d, l, r, L\}$ of these relations. These relations are interpreted by *modalities* in our language in the standard way.

Definition 2 (Language of Declarative Units) The language of *declarative units* is a first order language with *Non-logical Vocabulary*:

1. a denumerable number of sorted constants from Lab for $i \leq n$, where $\mathcal{L}_i = \langle Lab_i, R^1, \ldots, R_i^m \rangle$ structures the set of *feature values* in Lab,
2. *monadic predicates* Fo ('*Fo*rmula'), Ty ('*Ty*pe'), Tn ('*T*ree *n*ode'), C_i, $i \leq n$ and identity '= ',
3. *modalities* $\langle u \rangle$ (up),$\langle d \rangle$ (down) ,$\langle l \rangle$ (left), $\langle r \rangle$ (right), $\langle L \rangle$ (link) and their starred versions $\langle d \rangle^*, \langle u \rangle^*, \langle l \rangle^*, \langle r \rangle^*$. We will use the abbreviation $\langle \# \rangle^n$, $n \in I\!N$, for
$$\underbrace{\langle \# \rangle \ldots \langle \# \rangle}_{n \text{ times}}$$

Formulas:

1. If $j \in L_C$ then $Fo(j)$ is an (atomic) DU-formula.
   If $k \in Lab_1$ then $Ty(k)$ is an (atomic) DU-formula.
   If $k \in Lab_2$ then $Tn(k)$ is an (atomic) DU-formula.
   If $k \in Lab$, $2 < i \leq n$, then $C_i(k)$ is an (atomic) DU-formula.
   If t, t' are variables or individual constants, then $t = t'$ is an (atomic) DU-formula.
2. If ϕ and ψ are DU-formulas and x, then $\phi \# \psi$ is a DU-formula for $\# \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
   If x is a variable and ϕ a DU-formula, then $\forall x \phi$ and $\exists x \phi$ are DU-formulas.
   If M is a modality and ϕ a DU-formula, then $M\phi$ is a DU-formula.

This language we use to formulate properties of declarative units and their rules.

- Declarative units are related to each other through their *tree node feature*: If $Tn(n)$ is an element of a partial declarative unit DU_i, we may represent as follows:

$$\begin{array}{ccccc} \text{Tree Node } n & & & & \text{labelled formula} \\ Tn(n) & \longleftarrow & i & \longrightarrow & \{C_1(l) \ldots C_k(l')\} \end{array}$$

  That is, unit i is located at node n. We have the modal entity of a *decorated tree node*. If $m, n \in Lab_{Tn}$ such that $m \leq n$ and $Tn(m) \in DU_1, Tn(n) \in DU_2$ then $DU_1 \leq DU_2$. This we can call *absolute addressing*. But there are also *relative* ways of expressing tree relations. For instance, allowing declarative units to contain non-atomic *addresses*, if $Tn(m) \in DU_1$, $\langle l \rangle Tn(m) \in DU_2$, and $\langle u \rangle^* Tn(m) \in DU_3$, then $DU_1 \leq DU_3$ and $DU_1 < DU_2$.

$$DU_1 = \{Tn(m) \ldots C_k(l')\} \quad < \quad DU_2 = \{\langle l \rangle Tn(m) \ldots C'_k(l')\}$$

- A set of declarative units structured as a tree represents an *application* tree: declarative units at daughters combine to give a declarative unit at the mother. For instance, applications of the rule of Modus Ponens.

$$\frac{\langle e, \ldots \rangle : \phi \qquad\qquad \langle e \rightarrow t, \ldots \rangle : \psi}{\langle t, \ldots \rangle : \psi(\phi)}$$

  for declarative units get the *meta description*

$$\frac{\langle d \rangle (Ty(e) \wedge Fo(\phi)) \wedge \langle d \rangle (Ty(e \rightarrow t) \wedge Fo(\psi))}{Ty(t) \wedge Fo(\psi(\phi))}$$

  An item has Type Feature t and Form Feature $\psi(\phi)$ if it has daughters with Type Features $e \rightarrow t$ and t and Form Features ψ and ϕ respectively.

- We will use the following modal principles more or less implicitly (here $\# \in \{u, d, l, r, L\}$)

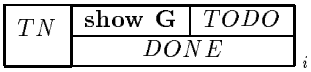
  1. $\langle \# \rangle^{-1}[\#]\phi \rightarrow \phi$ where $\langle u \rangle = \langle d \rangle^{-1}$, $\langle l \rangle = \langle r \rangle^{-1}$: mother-of is the converse of daughter-of and left-of the converse of right-of.
  2. $\langle \# \rangle^{*}\phi \leftrightarrow (\phi \vee \langle \# \rangle \langle \# \rangle^{*}\phi)$. For instance, stating that $\langle d \rangle^{*}\phi$ holds at a tree node n is equivalent to stating that ϕ holds at n or $\langle d \rangle^{*}\phi$ holds at a daughter of n.

2.3 Task States

In the course of a parse both the set of (partial) declarative units and the set of features connected to a particular (partial) declarative unit grows. The *Complete Declarative Units* (*Du's*) are *maximally consistent* subsets of $\mathcal{A}$. A *Du* is finite (modulo identity) and has the form

$$\{C_1(l_1), \ldots, C_n(l_n), Fo(\phi)\}$$

The growth of the structure of (partial) declarative units corresponding to an NL string is described in terms of transitions between *Task States*. With each declarative unit there corresponds one task. A Task State is a description of the state of a task. A task is completely described by the following four feature dimensions: **Goal (G)**. Values on this dimension are the *semantic types* in the label set Ty. This feature indicates which semantic object is under construction. **Tree Node (TN)**. Values are elements of the label set Tn. The 'top-node' in Tn will be denoted by 1. This feature fixes the location of the task in question within a tree structure. **Discrepancy (TODO)**. Values are (finite sequences of) DU-formulas. This dimensions tells us what has to be found/constructed before the goal object can be constructed.**Result (DONE)**. Values are lists, sequences, of DU-formulas. These values will be the partial declarative units of the Incremental Model. We will represent the task state $TS(i)$ by



We can distinguish three kinds of task states: **Task Declarations**: Nothing has yet been achieved with respect to a goal G, G is in TODO, DONE is empty. **Tasks in Progress**: both DONE and TODO are non empty. If G is the goal and if things are set up right, then $DONE, TODO \Rightarrow G$. **Satisified Tasks** There is nothing left to be done. TODO is empty *Soundness* of the deductive system amounts to the fact that the goal G can be computed, derived, from α in case TODO is empty.

2.4 Parse States

A parse state

$$\langle \boxed{\begin{array}{|c|c|c|}\hline Sc & Po & Le \\ \hline s & p & l \\ \hline\end{array}} (S) \rangle,$$

finally, consists of a *bookkeeping device* $\boxed{\begin{array}{|c|c|c|}\hline Sc & Po & Le \\ \hline s & p & l \\ \hline\end{array}}$ and a sequence S of task states. In the bookkeeping device, the value of *pointer* Po, p, is a natural number $p \leq l$. This value gives the task state in S currently under consideration. s Gives the value of the *string counter* Sc which represents the location of the reading head in the string. Finally, the length of the string of tasks state S is l, the value of Le.

3 Dynamics

The *dynamics* of the parse process consists of reaching a final parse state starting from the initial state where the *transitions* in the process are licensed, driven, by the words in a string. Concretely, the dynamics of the parsing process, is the dynamics of *demand satisfaction*: given the tree modalities, the presence of a modal formula of the form $\langle d \rangle \phi$ as TODO gives an *unsatisfied demand*, a so-called *defect*[1]: it requires action. The *task states* of LDS_{NL} can be seen as sets of DU-formulas together with demands to be dealt with. The parsing process then essentially consists of attempts to use the words of a string to satisfy the overall demand to construct an entity of the truth-value type t. The course from initial parse state to a final one is guided by *Transition Rules*.

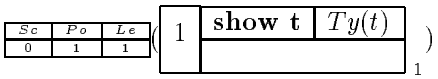
[1] We know this from completeness proofs in modal logic of the so-called *step-by-step* variety. In contrast to the situation in modal logic, however, satisfying nodes (maximally consistent sets) need to be eventually *licensed* by the words in a given string. Consistency alone is not enough.

3.1 Basic Transition Rules

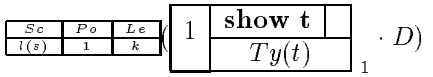
In the following the symbols $X, Y, Z, \ldots$ will range over individual DU-formulas, the symbols $\alpha, \beta, \ldots$ will range over (possibly empty) sequences of such formulas, D, $D', \ldots$ will range over (possibly empty) sequences of tasks, and $w_i, w_{i+1}, \ldots$ will range over words.

The start of a parsing sequence is a single task state, the *Axiom state*. The last element of such a sequence is the *Goal* state. The number of task states in a parse state grows by applications of the Subgoal- and Adjunction rules. Tasks become satisfied by applications of the Scanning and Thinning.

1. **Axiom**



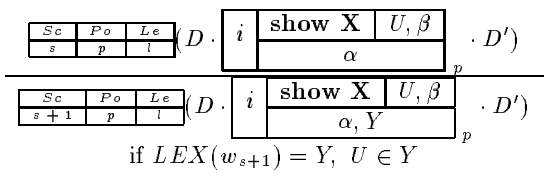
Goal



where s is a string with length $l(s)$ and all elements of D are *satisfied* task states.
In the Axiom state nothing has been done yet and it is required to construct an object of type t; in the Goal state this requirement has been fulfilled.

2. **String processing**. Essentially, only the words occurring in the string allow us to add information in the DONE compartment of a task state. This is formalized in the Scanning rule. When a formula occurs both in the DONE and in the TODO compartment, then it may be removed from the latter. This is the way TODO is emptied. The Thinning rule expresses this.

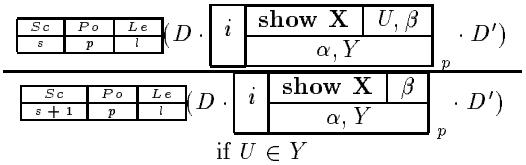
   (a) **Scanning**



if $LEX(w_{s+1}) = Y,\ U \in Y$

The expression $LEX(w_{s+1}) = Y$ refers to the lexical entry for the word w_{j+1}. This is a set of DU-formulas possibly containing U; if this set contains the *trigger* in the TODO box, then it may be added to the DONE box. In that case, the *Thinning* rule allows us to remove the demand from the TODO box.

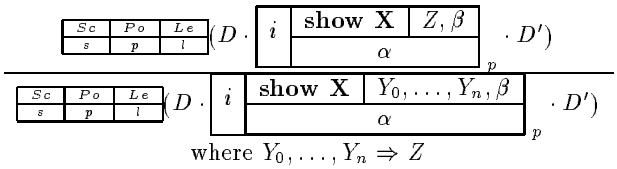
   (b) **Thinning**



This is the only rule allowing us to remove elements from TODO; we can only end up with satisfied tasks (empty TODO's) by fulfilling the requirements.

1. **Mode of Combination**. The following two rules deal with the processing of information available at a specific task node. The first rule, Introduction, analyses a requirement into subrequirements as specified by syntactic or logical rules present. The second rule, Elimination, combines — or rewrites — information that has been acquired, again according to the rules present.

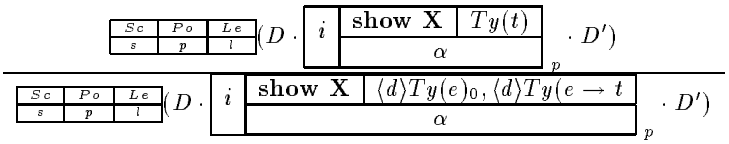
   (a) **Introduction**



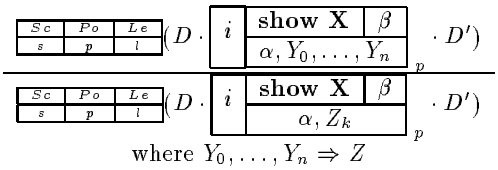
Here $Y_0, \ldots, Y_n \Rightarrow Z$ is some mode of combination: a requirement (Z) is analysed into subrequirements ($Y_1, \ldots, Y_n$). For instance for function application we have

$$\langle d \rangle Ty(X), \langle d \rangle Ty(X \rightarrow Y) \Rightarrow Ty(Y)$$

So, Introduction gives



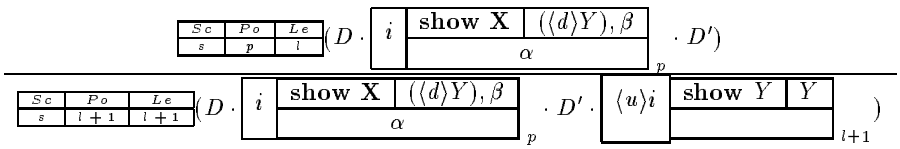
   (b) **Elimination**



The inverse of Introduction. This inverse relation guarantees that an empty TODO compartment corresponds to a DONE compartment which can derive the goal.

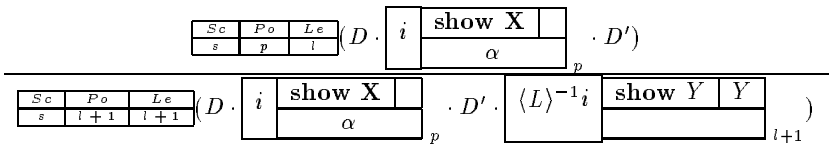
2. **Tree Extending Processes**. The following rules deal with the creation of the linked tree structure. Two rules deal with the building up of structure: the Subgoal rule which creates the trees proper, and the Adjunction rule which creates *linkages*. When a requirement at node $Tn(i)$ involves an existential modality, e.g. $\langle d\rangle Ty(e)$, "there is a daughter of type e", then the Subgoal rule creates such a daughter — at node $\langle u\rangle Tn(i)$ — with the requirement $Ty(e)$. The Adjunction rule only connects the top node of a new tree (still to be developed) — at node $\langle L\rangle^{-1}Tn(i)$ — to a *satisfied* task state — at node $Tn(i)$: adjuncts are 'superfluous' with respect to demand satisfaction. Having created new structure, the Completion rule tells us how to propagate the information through the tree once the requirements have been fulfilled.

   (a) **Subgoal**

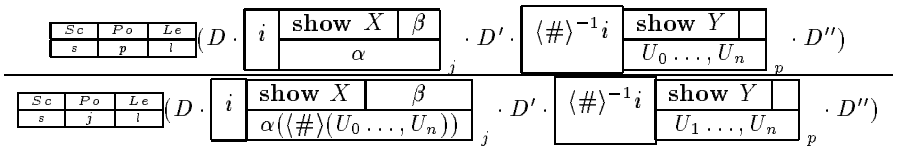


   Here a subtask is constructed in order to satisfy some demands in TODO of task i. When there are no demands then structures can be coordinated through *adjunction*.

   (b) **Y-Adjunction**



   (a) **Completion**



The completion rule combines both structures created by *Subgoal* and those created by *Y-Adjunction*.
This concludes the statement of the *basic transition rules*. Each of these rules

rewrites (concludes from) a single premise to a conclusion modulo some side conditions. Notice that rewriting can only take place at task states highlighted by the task pointer. This means that there may be rules required to shift this pointer. These rules won't be described here.

Remark 1 (Lexical Rules) We give a taste of the way lexical rules are formulated in this set-up. In the Lexicon we associate with a word w a set Y of DU-formulas

$$Lex(w) = Y$$

If the TODO feature of the task state at the pointer has value X, which we will abbreviate as $Tod(X)$, and $X \in Y$, then the information Y of w may be used (see the Scanning rule). Every word is entered relative to a context consisting of a *parse state*, that is a sequence of task states and a pointer, Po, a natural number $Po := n$ pointing to the n'th element of the sequence of length $Le := l$ and a string counter $Sc = j$. For instance,

1. If $Tod(Ty(x))$,
   $Lex(w_{j+1}) = Y, Ty(x) \in Y$,
   then **Scan**.

For instance, $x = e$, $Y = \{Ty(e), Fo(\mathbf{John})\}$. The task under consideration requires an item of type e, and the word to be processed, *John*, is of that type.

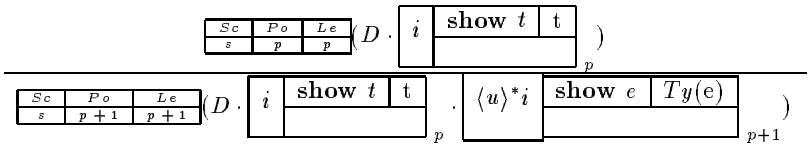
2. If $Tod(Ty(x))$,
   $Lex(w_{j+1}) = Y, Ty(y \to x) \in Y$,
   then a) apply **Introd.**: $Tod(\langle d\rangle Ty(y \to x), (\langle d\rangle Ty(y))$,
   b) apply **Subgoal**,
   c) apply **Scan**.

For instance, $x = e \to t$, $y = e$, $Y = \{ty(e \to (e \to t)), Fo(\mathbf{Love})\}$. The task under consideration requires a VP $(e \to t)$, but gets a transitive verb $(e \to (e \to t))$. $e \to t$ is split in $e \to (e \to t)$ and e, and scanning then leaves $Tod(\langle d\rangle(ty(e)))$.

3.2 Wh-Rules

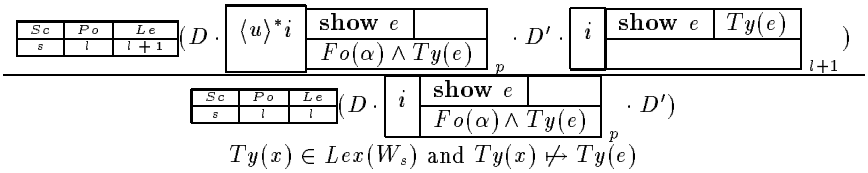
Besides the basic transition rules our system includes special purpose rules, for instance, for dealing with *wh* expressions. For sentence initial *wh* the premise parse state consists of the Axiom; for relative clause *wh*, first *Adjunction* creates an Axiom state linked to the head, next *Gap Adjunction* is applied to the fresh axiom.

Gap Adjunction

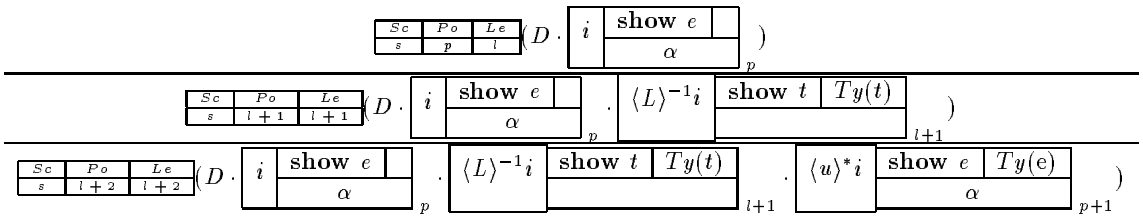


Now there is a node of type e 'dangling' at an unspecified location below the adjoint clause. This is our 'dynamic' representation of long distance dependencies. The kind of Wh word determines the subsequent actions through the lexicon. For instance, if Wh= *who*, then the Wh phrase consists of this single word and $Fo(Wh) \wedge Ty(e)$ is put in DONE. On the other hand, if Wh= *which*, then we are dealing with a proper Wh-phrase and Introduction splits $Ty(e)$ in $\langle d \rangle Ty(cn \rightarrow e)$ and $\langle d \rangle Ty(cn)$ and places $Fo(Wh) \wedge Ty(cn \rightarrow e)$ in DONE of the $\langle d \rangle Ty(cn \rightarrow e)$ daughter.
The *wh* constituent with underspecified location becomes fixed at the location of the *gap*. This is the import of the following rule.

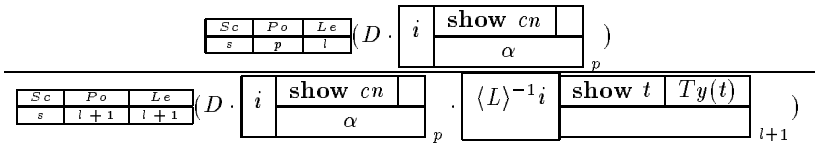
Gap Resolution



Here $Ty(x) \not\rightarrow Ty(e)$ means that the type of the current word is such that it cannot reduce to type e: at location $Tn(i)$ we find the *gap*.
So the characterisation of a DU-formula as unfixed relative to some node i, an option lexically encoded for English *Wh*, allows initial positions in a t-task to under-specify their tree relation. *Resolution* of such underdetermined specification takes place locally at some node i whose TODO specification is satisfied not by the specification of the next lexical input but by the presented floating constituent.
The adjunction and Wh-rules may apply to satisfied e- and satisfied cn-tasks. These correspond respectively to *non restrictive* and *restrictive* relative clause strategies. Both applications are driven by lexical specifications.
In case of *unrestricted relative clauses* we have the following sequence of rule applications:



First t-adjunction is applied and then Gap adjunction. Notice that the dangling constituent is 'loaded' with the information of the satisfied e-task that constitutes the head of the relative clause.
For *restricted relative clauses* we proceed as follows:



Here, (due to the type clash), no information of the head is loaded into the dangling component. When the gap is resolved and the restricted relative clause finished, then the Completion rule 'merges' the head with the relative clause. This requires a principle like

$$Fo(\lambda y \phi) \wedge Ty(cn), \langle L \rangle (Fo(\beta[Wh/y]) \wedge Ty(t)) \Rightarrow Fo(\lambda y(\phi \wedge \beta)$$

This principle allows the Elimination rule, after Completion, to merge, for instance, $Fo(\lambda y Man(y)) \wedge Ty(cn)$ and $Fo(Walks(WH)) \wedge Ty(t)$ to $Fo(\lambda y(Man(y) \wedge walks(y))) \wedge Ty(cn)$.

4 Application of the Model - Crossover: A Dynamic Perspective

Before we can discuss the application of this framework to the treatment of crossover phenomena we need a characterisation of underspecificity and its resolution in context. We define contextual resolution to be the selection of some suitable syntactic choice, given what is independently made available. Anaphora resolution, for example, involves a choice relative to the constraints the particular lexical item imposes. Pronouns are defined as meta variables of the form u_{pro}

$$Lex(he) = \{Ty(e), Fo(u_{\text{pro}}), Gender(male), \langle u \rangle Ty(t), ...\}$$

Determiners also may project such meta variables, eg the definite and demonstrative determiners. The instantiation of the meta variable 'u_{pro}' must be a formula in some *satisfied task* or a formula which has been derived elsewhere in the discourse sequence. This holds for all anaphoric expressions. The process is presumed to be on-line, and subject to a locality restriction specific to pronominals (following Pollard and Sag 1994). [2] The novelty of our account of *wh* lies in the characterisation of initial *wh* expressions as projecting an unfixed constituent, and in the sensitivity of anaphora resolution to the progressive accumulation of information. A benefit of looking at *wh* phenomena from this perspective is that phenomena classified as mysteries are predicted - eg the interaction of *wh* construal and anaphora resolution called "crossover". The restriction, provisionally described, is that a pronominal cannot be construed as bound by the *wh* expression if the gap which the *wh* is to bind is to the right of the pronominal:

[2] Exceptions to the on-line restriction are invariably language- and construction-particular, and must be specifically encoded.

(1) *Who$_i$ does Joan think that he$_i$ worries e$_i$ is sick?

(2) *Who$_i$ does Joan think that his$_i$ mother worries e$_i$ is sick?

(3) *Whose$_i$ exam results$_j$ was he$_i$ certain e$_j$ would be better than anyone else's?

(4) Who$_i$ does John think e$_i$ worries his$_i$ mother is sick?

(5) Who$_i$ does Joan think e$_i$ worries that he$_i$ is sick?

(6) Whose$_i$ exam results$_j$ e$_j$ were so striking that he$_i$ was suspected of cheating?

This restriction is context-sensitive, being apparently rescinded in some circumstances. Within the GB paradigm, the phenomenon has been argued ([LasSto91]) to subdivide into at least three different phenomena. "Strong crossover" is the restriction which prohibits a pronoun from being interpreted as dependent on a c-commanding operator if it both precedes and c-commands the trace bound by that operator, said to be due to the name- like properties of the empty-category (a principle C effect) - (1). "Weak crossover" is a restriction that precludes a pronoun from being bound by some c-commanding operator in the presence of a following trace, when the pronoun does not c-command that trace - (2). This distinction is not made in the categorial grammar account ([Dow92]). There, crossover arises from the interaction between two different types of quantifying in process, in both cases discharging some initial assumption constructed for the purpose of preserving local semantic compositionality. *Wh* binding involves a regular quantifying operation, with a constructed assumption locally combining with other premises and then being discharged at some arbitrary point in the compilation of content. The output of this rule, being of the form $\lambda x.\psi$, combines with the higher type *wh* expression to yield a propositional formula with the variable associated with the *wh* suitably binding the abstracted position. Anaphora resolution is seen ([Hep90]) as a specialised variant of quantifying in, a process which discharges an assumption corresponding to the pronominal in the presence of some antecedent, without any alteration in the resulting type assignment. The precluded order is predicted to be underivable because though the gap assumption could be discharged in the presence of the pronominal, this would leave nothing for the *wh* to apply to, and vacuous application is debarred.

The problem in both accounts is that the phenomenon is predicted in virtue of the configurational relation between the parts, yet weak crossover effects display context-sensitivity. Nonrestrictive relative clause constructions, parasitic gap constructions, and *easy to please* constructions, all fail to display weak crossover effects:

(7) John, who$_i$ his$_i$ mother had regularly ignored e$_i$ fell ill during the exam period.

(8) *John, who$_i$ Sue thinks he$_i$ doesn't believe e$_i$ is sick, fell ill during the exam.

(9) Which$_i$ of the team did the judge put away e$_i$ without his$_i$ mother being able to see?

(10) Sam$_i$ is all too easy for his$_i$ mother to ignore e$_i$.

The categorial account does not extend to these data; and the Principles and Parameter account requires a number of additional principles ([LasSto91]).[3]

Data such as (7), (9),(10) are said to constitute a discrete phenomenon of "weakest crossover" in which the empty category is a pronominal epithet and not a "true variable". This division into a third category is however problematic, as demonstrated in detail by [Pos93]; and in any case, as Postal reminds us, there are data which fall within the strong crossover classification such as (3), (6) which nevertheless display exactly the same context-sensitivity as weak crossover phenomena:

(11) John, whose$_i$ mother$_j$ I told him$_i$ e$_j$ had refused the operation, was very upset.

(12) John, whose$_i$ exam results$_j$ he$_i$ had been certain e$_j$ would be better than anyone else's, failed dismally.

There is also puzzling variation across languages, which remain intransigent, given this mode of explanation.

Looked at from the dynamics of processing, the data are unproblematic. Formulas characterised as holding at some node $\langle u \rangle^* m$ do not have a fixed position in the configuration, and do not have an identified position from which to project the information contained in that formula. Without any such identification, as in *wh* questions, the information contained therein will not be visible for the purpose of pronominal resolution, unless it is identified independently of its position. The effect of Wh resolution when it applies is indeed to determine this position within the configuration. Hence the primary crossover restriction $*wh$ ...pronoun...gap (for both weak, strong, and extended strong crossover data)[4].

This restriction does not apply invariably in relative clauses, because of the different information made available at the outset of processing the different

[3]The asymmetry of context-sensitivity as between strong crossover and weak crossover environments is particularly puzzling for the Categorial analysis of [Dow92]. On this analysis, principle B is but a pragmatic effect which discourages co-referential readings in a given locally restricted environment (cf. [Hep90] who explicitly excludes any account of principle B effects from the account of anaphoric binding). If the circumstances in which a weak crossover effect fails to hold is to be explained as a pragmatic phenomenon of incidental co-reference, then there is no reason to preclude a similar effect of pragmatic coreference in example 8). In particular, there is no basis for invoking any locality restriction, since the surface sequence of expressions provides no basis for predicting that such a restriction might be applicable.

[4]The lack of crossover effect in parasitic gaps or *easy to please* constructions is predicted to be unproblematic, since in these cases, the previous expression (the true gap or the subject of *easy*) will provide the antecedent for the pronoun.

types of structure. In processing a question, no information is available at all about the *wh* expression:

(13) Who did you see?

In processing a relative clause construed *non restrictively* information is made available from the outset about the nature of the *wh*-expression[5].

(14) John, who you saw, was a linguist.

Because such a *wh* expression is identified as 'John', it can count as an antecedent for the pronominal.
In restrictive relative construals however, where t-adjunction links a clause to a *cn*-task, no information is transferred from the initially completed *cn*-task to the linked structure (because of the type clash between the type *cn* and the type of the Wh variable), and so, as in questions, the DU-formula $Fo(Wh) \wedge Ty(e)$ remains invisible until after the identification of the tree node from which it is to project its content. So dependence of the pronominal on the Wh is impossible:[6]

(15) Every child who$_i$ his$_i$ brother ignored e$_i$ throughout his party was clearly ill at ease.

(16) At least one woman who$_i$ her$_i$ cousin refused to cooperate with e$_i$ reported the matter to the authorities.

The asymmetry between (17) and (18) is not covered by this process of identifying the *wh* relativiser; but it can be explained straightforwardly by the way information projected by the *wh* expression is transferred node by node through the emerging structure to its point of resolution.

(17) *John, who$_i$ Sue knows he$_i$ thinks e$_i$ will fail, did surprisingly well.

(18) John, who$_i$ Sue knows e$_i$ thinks he$_i$ will fail, did surprisingly well.

This successive transfer of information gives rise to the following prediction. With Wh identified independently by a given antecedent α in nonrestrictive construals of the *wh* element, the formula '$\langle d \rangle^*(Fo(\alpha) \wedge Ty(e))$' carried down will be

[5]Definite and definite determiners are the two primary types of determiner which license both restrictive and nonrestrictive construals. In both cases, adjunction is to a completed term of the formula language, either in the case of definites as an instantiated formula analogous to pronominals, or in the case of indefinites as a completed epsilon term (with no dependency needing to be compiled). Cf. [MeyKKG96].

[6]The data here are notoriously unclear (hence the term "weak crossover"), and linguists have differed in their decision as to how to classify these data. Some examples seem to display a weak crossover effect, in particular relatives clauses modifying a quantified determiner head as in (15-16), others do not. Judgments are much less clear with definites and generic construals, where there appears to be interference from the implication implicit in such construals that the resulting expression is independently identifiable: (i) The man who$_i$ his$_i$ mother systematically ignored e$_i$ made a good parent. (ii) The children who$_i$ their$_i$ mother systematically helps e$_i$ with their homework do well. In the face of this somewhat unclear data we adopt the view that restrictive relatives display crossover effects as in questions. Cf. Also footnote 7.

local everywhere along the chain of nodes intervening between its projection and its resolution. In consequence there will be locality clashes with any pronominal itself projecting Ty(e) onto a task-state along that chain. The restriction on identification of pronominals is that any DU-formula '$Fo(u_{\text{pro}}) \wedge Ty(e)$' within a given task **Show** t or **Show** e, could not be identified with any other DU-formula '$Fo(\alpha) \wedge Ty(e)$ of type e as also being a minor premise to the same task. If, then, the carried down $\langle d \rangle^*$ specification '$\langle d \rangle^*(Fo(\alpha) \wedge Ty(e))$' and the projection from the pronoun agree on the type specification, then the identification of u_{pro} as α is precluded. If, on the other hand, either the pronominal being projected or this specification is within a determiner, then the pronominal and the Wh element are not local w.r.t. each other. Hence the asymmetry between (8) and (7).

Since the DU-formula '$Fo(\alpha) \wedge Ty(e)$' projected from the *wh* expression is part of the information available at the intervening task states despite not yet having its combinatorial role fixed, this analysis further predicts that anaphora resolution in nonrestrictive relatives with the order *wh* pronoun will parallel non-relative sentences in which a name as antecedent is followed by a pronoun. (19) is predicted to parallel (20) (both *wh* and the definite NP precede the pronominal *his*). (12) parallels (21) (both *John's exam results* and *whose exam results* precede the pronominal). Similarly (22) and (23):

(19) The bastard who$_i$ his$_i$ mother worshipped.

(20) The bastard$_i$ worships his$_i$ mother.

(21) John$_i$'s exam results gave him$_i$ a nasty shock.

(22) *Joan behind whom$_i$ she$_i$ looked e$_i$ nervously, coughed.

(23) *Behind Joan$_i$ she$_i$ looked nervously.

This principle-B style of analysis will not apply at all to construals of relative sequences as adjuncts on the *cn*-task, because the consequence of building an adjunct structure on the cn task is that no information will be transferred into the building of the adjoined structure, so no potential antecedent is made available, and their construal is predicted to be controlled solely by the lack of availability of an antecedent - hence the lack of context-sensitivity in such cases, parallelling questions (modulo the caveat of footnote 6 as already discussed). [7]

[7] The account in the text presumes that the identification of pronominal occurring within a relative sequence will either make use of the information made available by the *wh* element, if any, or, if not, by some information which is identified independently from outside that structure. There is a third possibility. Given that the building of the adjoined structure for a relative sequence construed restrictively is an adjunction to a *cn*-task itself part of an unsatisfied *e*-task containing the projection of a determiner plus nominal, then the question arises whether the formula projected by that unsatisfied *e*-task can be used as the basis for assigning a value to a pronominal occurring within the relative sequence. The answer is that, in some languages, the answer seems to be yes. While English speakers are hesitant about many crossover judgments with restrictive relatives, particularly if the head is a definite determiner, French speakers display no hesitation over all weak crossover sequences, finding all to be fully acceptable. Some languages (and other dialects of English) accordingly seem to be less sensitive to the requirement that the term used for anaphora resolution be fully identified.

Striking confirmation of this procedural approach comes from Chinese, which provides a test case. Chinese has two relative strategies, one in which the entire relative precedes the Determiner plus nominal, the other in which the Determiner precedes the relative sequence with its nominal head following. The analysis proposed here predicts no possibility of anaphoric dependence between a pronominal within the relative in the first strategy, and allows anaphoric dependence as in English for the second. This prediction is correct. In the first strategy, with no identification that the sequence is a relative clause as in (24)-(25), there are only created TODO tasks associated with the initial expression, so even where it is the verb that is initial, there is no completed task to provide an antecedent for the pronominal:

(24) *Ta$_i$ muqin hushi e$_i$ de mei ge ren$_i$ dou mei kao jige.
he mother ignore rel every CL man all not test pass
Everyone who$_i$ his$_i$ mother ignored e$_i$ failed the test.

(25) *e$_i$ Hushi ta$_i$ muqin de mei ge ren$_i$ dou mei kao jige.
ignore he mother rel every CL man all not test pass
Everyone who$_i$ ignored his$_i$ mother failed the test.

In the second strategy, if the pronominal intervenes between the determiner and gap it will not allow dependence of the pronominal on the relativiser or the head, as in English.

(26) *Mei ge ta$_i$ muqin hushi de ren$_i$ dou mei kao jige.
every CL he mother ignore DE man all not test pass
Everyone who$_i$ his$_i$ failed the test.

But if the gap in the relative sequence precedes the pronominal then the coincidence of the disjunctive specification induced by the triggering of the need to build a relative clause and the TODO tasks created by the verb, ensure that the disjunction is resolved and the position can duly act as antecedent for a pronominal:

(27) Mei ge e$_i$ hushi ta$_i$ muqin de ren$_i$ dou mei kao jige.
every CL ignore he mother DE man all not test pass
Everyone who$_i$ ignored his$_i$ mother failed the test.

These data follow as an immediate consequence of the analysis proposed here. On the contrary, however, the crucial data (25), (27) are not characterisable as a crossover phenomenon on either GB or categorial analyses, and no unifying analysis is possible.

5 Conclusion

A distinguishing feature of this account of crossover has been the underspecification of constituent structure. The concept of structural underspecification

is close to the concept of functional uncertainty of Kaplan and Zaenen articulated within the LFG framework ([KapZae89]), but in that analysis the uncertainty is defined in terms of thematic roles which are structural primitives. The consequent transfer of content for the initial expression through the tree configuration is like the slash mechanism of HPSG ([PolSag94]), though in HPSG such transfer is defined as a mechanism of upward feature percolation. Like categorial grammar analyses, the basic mode of combination is type-deduction, and *wh* expressions are defined to ensure the combination of the content projected by the *wh* expression with expressions with which it is not contiguous (cf eg [Moo88], [Mor94]). However unlike categorial grammar accounts, this is not defined through some suitable type assignment, and no semantic mode of combination is defined between the *wh* expression and its sister expressions in the string. Seen in the most general terms, the account we have proposed presents the claim that the concept of context-dependence and its resolution is structurally defined, and that resolution of any underspecification of either denotation or structure is determined on a left-right basis that broadly follows the order in which the interpretation is presented. A surprising prediction follows immediately from this claim, distinguishing it from other frameworks: a sharp asymmetry is predicted between left-dislocation of *wh* expressions which occurs freely in language, and right-dislocated *wh* expressions which are predicted not to occur. Under this account, postposing of *wh* elements out of their structurally licensed position to the right periphery of a sentence is precluded. The lexical projection of a DU-formula with an underspecified tree node, with no further input to provide a means of resolution, would lead to inconsistency and predicted illformedness. Predicted is the non-occurrence of any question such as (28), with the *wh*-expression right-dislocated, in contrast to the wellformedness of its left-dislocated counterpart (29):

(28) *You think e_i saw Bill at the market who_i?

(29) Who_i do you think e_i saw Bill at the party ?

So far as we know, this prediction is not only correct across all languages, but cannot be made to follow directly from any other framework.

References

[BlaMey94] Blackburn, P. & Meyer Viol, W. Linguistics, logic and finite Trees. *Bulletin of the IPGL* 2:2-39,1994.

[Dow92] Dowty, D. 1992. "Variable-free' syntax, variable-binding syntax, the natural deduction Lambek calculus and the crossover constraint'. *Proceedings of the 1992 West Coast Conference on Formal Linguistics.*

[Gab90] Gabbay, D. 1990/fcmng. Labelled Deductive Systems. Oxford University Press.

[GabKem92] Gabbay, D. & Kempson, R. 1992. 'Natural-language content: a proof-theoretic perspective'. *Proceedings of 8th Amsterdam Semantics Colloquium*. Amsterdam.

[Hep90] Hepple, M. 1990. The Grammar of Processing of Order and Dependency: A Categorial Grammar. Ph.D Dissertation. University of Edinburgh.

[KapZae89] Kaplan, R. and Zaenen, A. 1989 'Long-distance dependencies, constituent structure, and functional uncertainty, in Baltin, M. and Kroch, A. (eds.) Alternative Conceptions of Phrase Structure, pp. 17- 42. University of Chicago Press. Chicago, London.

[KemMVGab] Kempson, R, Meyer-Viol, W., Gabbay, D. (forthcoming) 'Syntactic computation as labelled deduction: *wh* a case study' .Borsley, B. & Roberts, E. (eds.)*Syntactic Categories*. Academic Press.

[Kra95] Kracht, M. 1995. Syntactic codes and Grammar Refinement. *Journal of Language, Logic and Information* 4:41-60.

[LasSto91] Lasnik, H. & Stowell, T. 1991.'Weakest crossover' *Linguistic Inquiry* 22, 687-720.

[Mey95] Meyer Viol, W. 1995. Instantial Logic. Ph.D Dissertation, University of Utrecht.

[MeyKKG96] Meyer Viol, W., Kibble, R., Kempson, R., Gabbay, D. 1996/fcmng. Indefinites as Epsilon Terms: A Labelled deduction Account. *Proceeding of the Second International Workshop on Computational Semantics*. Tilburg.

[Mil94] Milward, D. 1994. Dynamic Dependency Grammar. *Linguistic and Philosophy*, 17:561-605.

[Moo88] Moortgat, M. 1988. Categorial Investigations. Fores, Dordrecht.

[Mor94] Morrill, G. 1994. Type-Logical Grammar. Kluwer, Dordrecht.

[PolSag94] Pollard, C. & Sag, I. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press. Chicago.

[Pos72] Postal. P.1972. *Cross-over Phenomena*. Holt, Rinehart & Winston.

[Pos93] Postal, P. 1993. 'Remarks on weak crossover effects'. *Linguistic Inquiry* 24, 539-56.