Indefinites as Epsilon Terms: A Labelled Deduction Account *

Wilfried Meyer Viol
Department of Computing
Imperial College
University of London
email: wm3@doc.ic.ac.uk

Rodger Kibble
Department of Linguistics
School of Oriental and African Studies
University of London
email: rodger@semantics.soas.ac.uk

Ruth Kempson
Department of Linguistics
School of Oriental and African Studies
University of London
email: kempson@mailbox.ulcc.ac.uk

Dov Gabbay
Department of Computing
Imperial College
University of London
email: dg@doc.ic.ac.uk

Abstract

This paper gives an account of indefinites as epsilon terms within a model of utterance processing as a task of labelled deduction, in which partially specified inputs are progressively resolved during the interpretation process. The formal tools used are labelled deduction (Gabbay forthcoming), the epsilon calculus (Meyer-Viol 1995), and a tree-logic (LOFT - Blackburn & Meyer-Viol 1994) which characterises the structure as it is incrementally built. The system has been partly implemented in SWI-Prolog as part of a longer-term project and the model is a state transition system which defines the steps which license movement from state to state.

1 Introduction

It is a well known problem of on-line parsing that quantified NPs, in particular indefinites, may be interpreted as giving rise to logical forms in which they have a scope very different from that indicated by the surface sequence of expressions in which they occur (a problem which cannot be reduced to positing referential uses of indefinites cf. Farkas 1981, Abusch 1994). Various computational and formal systems have tackled this ambiguity problem, either, as is familiar, by positing processes of restructuring or storage (Montague 1974, Cooper 1983, May 1985, Morrill 1994, Pereira 1990), or by building underspecified structures including unscoped or partially scoped representations of quantified NPs (Alshawi & Crouch 1992, Reyle 1993, Pereira & Pollack 1991). In all of these cases, there is implicit recognition that the determination of scope choice is not incremental, but can only be

*This research was supported by the UK Engineering and Physical Sciences Research Council under grant reference GR/K67397, "A Labelled Deductive System for Natural Language Understanding"

defined once the total structure is complete. In this paper we propose that the interpretation of indefinite NPs involves an anaphoric-like dependency, in which the indefinite is lexically projected as involving a dependent name for which the *anchor* of the dependency has to be chosen on-line. The dependency is represented by an indexing on the name, the index indicating the expression to which the dependent element is to be anchored.The anchor dictates the mode of combination of the parts and the construction of the resulting dependent term.

The account is set within a model of utterance interpretation as a structure-building operation assigning logical forms to a sentence through a mixed process of labelled type deduction and other operations. This model is part of a general programme developing a deductive framework for natural language understanding (cf. Gabbay & Kempson 1992, Finger et al 1996, Kempson et al forthcoming). The formal tools we use are labelled deduction to define type deduction and other mixed modes of inference (Gabbay forthcoming), the epsilon calculus as the selected logical language (Meyer-Viol 1995), and the modal tree logic LOFT for modelling the architecture within which the setting out of premises for type deduction takes place (Blackburn & Meyer-Viol 1994). Utterance processing is taken to be a goal-driven inferential task, in which the goal is to establish a logical formula of type $\mathbf{t}$ using information as incrementally provided on a left-right basis by a given input string.

Within this system, all NPs project onto formulas labelled with type e, with epsilon and tau terms for quantified NPs of existential and universal force respectively. Context dependence is reconstructed as a syntactic choice defined over the configurations built up during the interpretation process allowing content to be initially underspecified. All determiners lexically define such partial specifications of content, projecting specialised metavariables which are resolved in various ways. Anaphora resolution, for example, is assumed to involve an initial metavariable which is associated with a very free choice mechanism, restricted only by a locality constraint precluding representations which are too local to the point in the structure in which the metavariable is itself projected. Indefinites, on the other hand, project a metavariable corresponding to an incomplete epsilon term; a choice mechanism selects as the anchoring index some other representation projected in the interpretation process; and the elimination rules then progressively compute the formula in which the effect of this choice of anchoring is made explicit.

2 The ϵ-calculus: overview and applications

The ϵ-calculus gives us a quantifier free language which has all the expressivity of a first-order language (plus some more). It gives us a way to represent all noun phrases as entities of type e. Instead of quantified formulas the ϵ-calculus uses quantified terms to achieve the same purpose. These terms are constructed with *variable binding term operators* (Costa 1980), each such operator ν corresponding to a quantifier. The building principle of these terms is: whenever ϕ is a formula of the language, then $\nu x \phi$ is a *term* in which all occurrences of x free in ϕ are bound by the operator νx. In this paper we will use only two such operators: epsilon ϵ, and tau τ. These give rise to ϵ-terms and τ-terms

- The term $\epsilon x\phi$ denotes some arbitrary d in the domain which has the property ϕ, if there are any such objects, and an arbitrary d *tout court* if there are no such objects. Given this choice of d, it is evident that $\phi[\epsilon x\phi/x]$ is true precisely if $\exists x\phi$ is. So we have the equivalence $\exists x\phi \leftrightarrow \phi[\epsilon x\phi/x]$
- We interpret the term $\tau x\phi$ as shorthand for $\epsilon x\neg\phi$, i.e. it denotes an arbitrary object d s.t. d does *not* have the property ϕ if there is such an object, otherwise an arbitrary d *tout court*. The following equivalence holds: $\forall x\phi \leftrightarrow \phi[\tau x\phi/x]$

Variable binding term operators can be introduced for all (generalized) quantifiers. For instance, in "most men walk" the term $a = \text{most}x(\text{man}(x) \rightarrow \text{walk}(x))$ would denote a walking man if indeed most men walk and some arbitrary non-walker otherwise. So walk(a) holds exactly iff most men walk. [1] And in "No man walks" the term $b = \tau x(\text{man}(x) \rightarrow \neg\text{walk}(x))$ would give the right denotation. That is, man(b) $\rightarrow \neg$walk (b) holds iff no man walks.

In the quantified formula $Qx\phi$, the binding structure of the variable x in the matrix ϕ is completely described by the triple $\langle q, x, \phi\rangle$, where q denotes the *mode* of quantification of x and ϕ denotes the *scope* of that quantification. In the calculus with variable binding term operators this description is substituted, as it were, for x thus eliminating the need for the quantifier.

In the ϵ-calculus quantifier scope interactions are exhibited as term dependencies. For example, the two readings of *Every nurse visits a patient* translate as follows into the quantifier free calculus.

Narrow Scope Object $Nu(b) \rightarrow (Vis(b, a_b) \wedge Pat(a_b))$ where

1. $a_y = \epsilon x(Vis(y, x) \wedge Pat(x))$
2. $b = \tau y(Nu(y) \rightarrow (Vis(y, a_y) \wedge Pat(a_y)))$
3. $a_b = \epsilon x(Vis(b, x) \wedge Pat(x))$

Here the patient is construed as a term depending on the subject of 'visits' i.e. 'every nurse'. This dependence is reflected in the fact that the τ-term for nurse (b) occurs inside the final representation of patient (a_b)

Wide Scope Object $(Nu(b) \rightarrow (Vis(b_a, a)) \wedge Pat(a))$ where

1. $a = \epsilon x Pat(x)$
2. $b_x = \tau y(Nu(y) \rightarrow Vis(y, x))$
3. $b_a = \tau y(Nu(y) \rightarrow Vis(y, \epsilon x Pat(x))$

[1] As it stands the use of the implication sign might be confusing, though note that we do get the right interpretation. In general, but not in this paper, we are considering to let operators like *most* bind variables in pairs of formulas. 'Most men', would start its life as $\langle \mu, x, Men(x)\rangle$ without a semantic interpretation and get its definitive shape $\mu x(Men(x), Walk(x))$ by some rule like the one for τ-terms presented in this paper.

In this case, the patient does not depend on anything. In fact, in this calculus $\epsilon x Pat(x)$ occurs inside the representation of the term for every nurse.

In the narrow scope reading the term derived from *a patient*, $\epsilon x(Vis(y,x) \wedge Pat(x))$ has the variable y free, i.e. the value of of the term $\epsilon x(Vis(y,x) \wedge Pat(x))$ covaries with or depends on the value of y according to the extension of the *visit* relation (seen as a set of ordered pairs). In the wide scope reading where the patient is the same for each nurse the term $a = \epsilon x Pat(x)$ is independent, i.e. contains no free variables [2].

Construction of the Terms

The parsing process is essentially a two stage affair (although in practice the stages may be interleaved, as in the current prototype implementation). In the first stage a 'parse tree' is constructed in which the leaves are decorated with functions and arguments to these functions. The second phase of the parse then consists in iteratively applying functions to arguments among sister nodes and depositing the result on the mother node.
After the first phase of the parse, quantified noun phrases in the string have resulted in leaves on which the *quantificational mode* of the NP is specified (indefinite, universal, etc.), the *restrictor* and scope information, couched in terms of *dependence*. These pieces of information suffice for the second stage of the parsing process to *algorithmically* compute the definitive shape of the quantified terms.

In the first stage of parsing "every nurse visits a patient" the noun phrases 'a patient' and 'every nurse' end up as an 'unscoped' proto ϵ-terms $\langle \epsilon, x, Pat(x) \rangle$ and $\langle \tau, x, Nu(x) \rangle$ respectively. *At this point the terms are not given a semantics as they stand*, but the parse tree contains enough information to algorithmically compute the eventual shape of the terms which do get an interpretation. The reason for this delay of interpretation is that restrictors of the terms may have to be extended in the course of the computation ("a patient who smiles,...."), or that the restrictor of the term still has to 'recombine' with the nuclear scope: terms starting their life as, for instance, $\langle \tau, x, \phi(x) \rangle$, $\langle$'no', $x, \phi(x) \rangle$ or $\langle$'most', $x, \phi(x) \rangle$ have to be restructured so as to incorporate the nuclear scope. For instance, in processing the sentence "every nurse visits a patient' the proto term $\langle \tau, x, Nu(x) \rangle$ is not interpreted as the term $\tau x Nu(x)$ but if the dependence information states a dependence of 'a patient' on 'every nurse' (i.e., narrow scope object), then the term giving 'a patient' its definitive semantic shape is constructed by the following Narrow Scope rule.

Definition 1 (Narrow Scope Object) Suppose ϕ free of λ, and L some sequence, possibly empty, of *lambda v*'s.

1. $\lambda x L \phi[x/y] \ + \epsilon y \psi = L(\phi[\epsilon y(\phi \wedge \psi)/y] \wedge \psi[\epsilon y(\phi \wedge \psi)/y])$
2. $\lambda x L \phi[x/y] \ + \tau y \psi = L(\psi[(\tau y(\psi \rightarrow \phi))/y] \rightarrow \phi[\tau y(\psi \rightarrow \phi)/y])$

[2] Note that the restrictor $\phi(x)$ of an epsilon term $\epsilon x \phi(x)$ may represent both the natural language *content*, of the quantified NP — the common noun 'patient' in "a patient" — and the *scope* of that NP in the entire clause. Within the confines of these proceedings we cannot elaborate on these complications.

That is, we use function application "with a twist": in applying we 'recombine' the restrictor and nuclear scope. Here we derive the object narrow scope reading of our example sentence

1. **Visit + a Patient**:

$$\lambda u \lambda y Vis(y,u) \ + \ \epsilon x Pat(x)) \ = \ \lambda y(Vis(y,a_y) \wedge Pat(a_y))^3$$

2. **Visit a Patient + every Nurse**

$$\lambda y(Vis(y,a_y) \wedge Pat(a_y)) \ + \ \tau y Nu(y) \ = \ Nu(b) \rightarrow (Vis(b,a_b) \wedge Pat(a_b))$$

where

(a) $a_y = \epsilon x(Vis(y,x) \wedge Pat(x))$
(b) $b = \tau y(Nu(y) \rightarrow (Vis(y,a_y) \wedge Pat(a_y)))$
(c) $a_b = \epsilon x(Vis(b,x) \wedge Pat(x))$

If parsing "every nurse visits a patient" results in the proto term $\langle \epsilon, xPat(x) \rangle$ with the information that there is no dependence (or, alternatively, a dependence on the time index/ world of evaluation [4]) then the Wide Scope application rule is called for.

Definition 2 (Wide Scope Object) Suppose ϕ free of λ, and L some sequence, possibly empty, of *lambda v*'s. Let $X(x,\chi)$ be the *smallest subformula* of χ containing all free occurrences in χ of the variable x.

1. $\lambda x L\phi[x/y] \ + \epsilon y\psi = L(\phi[\epsilon y\psi/y] \wedge \psi[\epsilon y\psi/y])$

2. $\lambda x L\phi[x/y] \ + \tau y\psi = L\phi[\xi/X(y,\phi)])$
where $\xi = \psi[(\tau y(\psi \rightarrow X(y,\phi))/y] \rightarrow X(y,\phi)[\tau y(\psi \rightarrow X(y,\phi))/y]$

1. **Visit + a Patient**:

$$\lambda u \lambda v Vis(v,u) \ + \ \epsilon x Pat(x) \ = \ \lambda y(Vis(y,a) \wedge Pat(a))^5$$

2. Visit a Patient + every Nurse

$$\lambda y(Vis(y,a) \wedge Pat(a)) + \tau y Nu(y) \ = \ (Nu(b) \rightarrow (Vis(b_a,a)) \wedge Pat(a))$$

where

(a) $a = \epsilon x Pat(x)$
(b) $X(y, \lambda y(Vis(y,a) \wedge Pat(a))) = Vis(y,a)$

[3] This is equivalent to $\lambda y(\exists x(Vis(y,x) \wedge Pat(x)))$

[4] There is an unresolved issue here concerning whether the world should be represented in a (two-sorted) object language or whether the world *labels* the formula; cf Finger & Gabbay 1993).

[5] This derives $\lambda v(\exists x Pat(x) \wedge Vis(v, \epsilon x Pat(x))))$.

(c) $b_x = \tau y(Nu(y) \rightarrow Vis(y,x))$

This semantic notion of dependency can be conceptually separated from the syntactic notion of scope (see e.g. Farkas , forthcoming). We make use of the notion of dependency in the way scope relations are determined as a by-product of the sequential parsing process. As was mentioned above quantified NPs are interpreted initially as unscoped 'proto-terms' but there is the option to fix dependencies on variables so as to place constraints on the eventual scope relations. For instance in the above sentence processing *every nurse* results in the proto-term $\langle \tau,\ y,\ Nu(y)\rangle$. At the point where *patient* is encountered there are two options:

2. (a) $\langle \epsilon,\ x,\ Pat(x)\rangle$
   (b) $\langle \epsilon,\ x,\ dep(y),\ Pat(x)\rangle$

Option (2a) triggers the combination rule resulting in the wide scope reading for the indefinite in (1) and option (2b) triggers the rule which results in the narrow scope reading. If we express the dependency of the variable x in (2a) on the world/temporal index directly in the representation language, then indefinites can be given a unitary characterisation as $\langle \epsilon, x, dep(y), Pat(x)\rangle$ with y to be chosen much like an anaphoric choice. Like anaphoric resolution this choice is primarily on-line but to account for (marked) cases where scope does not correspond to left-right order we need to allow dependency on a meta-variable which is instantiated later: In the first parsing stage of "A nurse visits every patient" 'A nurse' starts its life as the proto term

$$\langle \epsilon,\ x,\ dep(\mathbf{u}),\ Nu(x)\rangle.$$

At the end of that stage, the dependence has been resolved to

$$\langle \epsilon,\ x,\ dep(\mathbf{a}),\ Nu(x)\rangle.$$

where $\mathbf{a} = \langle \tau, x, Pat(x)\rangle$. The second stage of the parse then uses the dependence information to construct the semantic representation as below definition 1.
In choosing the source of dependence in these terms there are several options. We may choose the constructed term (if already in its definitive shape), or we may choose the variable associated with it (if we are dealing with a quantified NP). An altogether different possibility is to make a proto term dependent on the location in the *tree* where the anchor is located, or even on the identifier of the *task* it is associated with. All these options are currently being explored. In the implementation described below we initially select a task identifier as the anchor, which is eventually replaced by the term which is constructed when that task is satisfied.

3 Construction algorithm and implementation

The interpretation process is formalised in a LDS framework in which type-logical formulae as labels guide the parser in the goal-directed process of constructing labelled formulas

or *declarative units* (*DUs*). The goal of the parser is the deductive task **show(t)** (derive a formula of type **t**); this is achieved by generating a succession of subtasks **show(e), show(e→t)** etc. In this section we define declarative units and *task states* and describe the rules which license transitions between states, with some remarks concerning implementation considerations.

3.1 Declarative Units

The *formula* of a declarative unit is the side representing the content of the words supplied in the course of a parse. In the fragment used in this paper a formula is either a term of type e or a lambda-expression. The *labels* annotate this content with linguistic features and control information. These include:

(i) the logical types expressed as type-logical formulas: e, t are types and for any types a, b; $a \rightarrow b$ is a type.

(ii) Tree node identifiers: these identifiers and other properties of the nodes are described by LOFT (Logic of Finite Trees), a logic for a propositional modal language which allows nodes of a tree to be defined in terms of the relations that hold between them. (Blackburn & Meyer-Viol 1994). In the system described in this paper we only employ the modality $\langle d \rangle$ or "down": $\langle d \rangle P$ holds at tree-node i iff P holds at a daughter of i.

(iii) Features specifying other syntactic information such as Case, +Q for questionhood etc.

Declarative units are represented as finite sets of formulas $\{Lab_1(l_1), \ldots, Lab_n(l_n), Form(\phi)\}$. Each such set is called a *DU-formula*.

3.2 Task States

A **Task state** is a description of the state of a task. The four feature dimensions of a task state are

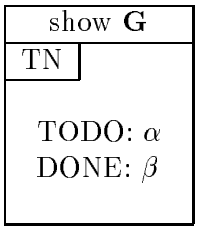
Goal (G) Values on this dimension are the semantic types in the label set *Ty*.

Tree Node (TN) This feature fixes the location of the task in question within a tree structure.

Discrepancy (TODO) Values are (finite sequences of) DU-formulas. This dimension 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 (partial) declarative units.

We will represent a task state either graphically by



or as the string $\{TN(n),\ Goal(G),\ TODO(\alpha),\ DONE(\beta)\}$

In the rules given below, tasks are identifed by the tree node identifier TN since there is a one-to-one correspondence between tasks and nodes. In the prototype implementation of this system task states are recorded in the Prolog database as unit clauses of the form `task(Tn, show(Goal), ToDo, Done)` where `ToDo` and `Done` are lists. This not only simplifies the programming task of accessing and updating task states but according to the particular Prolog implementation may allow fast direct access to any task state via the mechanism of indexing clauses for a predicate by their first argument (see e.g. Shieber et al 1994). We can distinguish three kinds of task states:

Task Declaration Nothing has yet been achieved with respect to the goal G:
$\{Goal(G), Tn(TN), ToDo(G), Done(\emptyset)\}$

Tasks in Progress In the middle of a task: $\{Goal(G), Tn(TN), ToDo(\beta), Done(\alpha)\}$

If things are set up right, then $\alpha\beta \Rightarrow G$ by some rule of combination.

Satisfied Task $\{Goal(G), Tn(TN), ToDo(\emptyset), Done(\alpha)\}$

Soundness of the deductive system amounts to the fact that the goal G can be computed, derived, from α in case ToDo is empty.

3.3 Parse States and Dynamics: The Basic Transition Rules

A parse state consists of a pair : a bookkeeping device which gives a value for the parsing pointer identifying the current task and the string counter which represents the location of the current word in the input string; and a sequence S of task states (to avoid cluttering the definitions, we generally omit details of the bookkeeping device in what follows). The dynamics of the parse process consists of reaching a final parse state or *Goal State* starting from a begin state, the *Axiom State*:

(1) AXIOM: $\{Goal(t), Tn(TN), Todo(Ty(t)), Done(\emptyset)\}$

(2) GOAL : $(\{Goal(t), Tn(TN), Todo(\emptyset), Done(Ty(t), \ldots)\}$. D)
where Elements of D are task states with fully specified DU formulas.

As was stated in section **2**, the parsing process consists of two conceptually separate phases, namely constructing a function-argument tree (in the process recognising a

grammatical sentence) and iteratively applying functions to arguments to derive a logical form. In our system the "structure-building" rules which construct the initial parse tree are **Introduction**, **Subordination** and **Scanning**, while the "interpretive" rules are **Completion** and **Elimination**. These rules are defined below.

The symbols X,Y,Z,.. will range over individual DU-formulas, the symbols α, β,... will range over (possibly empty) sequences of such formulas, D, D',... will range over (possibly empty) sequences of tasks, and w_i, w_{i+1},... will range over words.

(3) SCANNING For some string position s and tree node i:

$$\frac{(D.\{Tn(i),\ Goal(X),\ TODO(U,\beta),\ Done(\alpha)\}.D')}{(D.\{Tn(i),\ Goal(X),\ TODO(\beta),\ Done(\alpha,Y)\}.D')}$$

if LEX(w_{s+1}) = Y, UϵY ; set $s := s+1$

The side condition specifies that Y is the lexical entry retrieved for the current word in the string, with type-specification U. This rule licenses the introduction of material from the lexicon as long as it satisfies the conditions specified in TODO. These conditions are established by the rule of INTRODUCTION.

(4) INTRODUCTION

$$\frac{(D.\{Tn(i),\ Goal(X),\ TODO(Z,\beta),\ Done(\alpha)\}.D')}{(D.\{Tn(i),\ Goal(X),\ TODO(\langle d\rangle Y_0,\ldots,\langle d\rangle Y_n,\beta),\ Done(\alpha)\}.D')}$$

where Y_0,....$Y_n \Rightarrow Z$

The relation $\Rightarrow$ stands for some mode of combination; in the current system we employ Modus Ponens over the type system, restricted to combinations of the types which occur in the grammar of the language being parsed (namely `t` plus the types of lexical items).

This rule licenses the introduction of a set of new subgoals as long as there is some mode of combination reducing the subgoals to the original one. Introduction itself sets up the need for a rule transferring the requirement on the daughter to a TODO at which the requirement has to be met:

(5) SUBORDINATION

$$\frac{(D.\{Tn(i),\ Goal(X),\ TODO((\langle d\rangle Ty(x)),\beta),\ Done(\alpha)\}.D')}{\begin{array}{l}(D.\{Tn(i),\ Goal(X),\ TODO((\langle d\rangle Ty(x)),\beta),\ Done(\alpha)\}.D'.\\ \quad \{Tn(k),\ Goal(x),\ TODO(Ty(x))),\ Done(\emptyset)\})\end{array}}$$

where k stands in the daughter relation to i

Subordination is one of a pair of rules which license the transition from a task to be carried out on a daughter to the creation of a task corresponding to that daughter. Like Introduction, it relies on a converse rule, COMPLETION, which defines the transition back to the mother:

(6) COMPLETION

$$\frac{\begin{array}{l}(D.\{Tn(i),\ Goal(X),\ TODO((\langle d\rangle Y, \beta),\ Done(\alpha)\}.D'. \\ \quad \{Tn(k),\ Goal(Y),\ TODO(\emptyset),\ Done(U_0, \ldots, U_n)\}.D'')\end{array}}{\begin{array}{l}(D.\{Tn(i),\ Goal(X),\ TODO((\beta),\ Done(\alpha, \langle d\rangle(U_0, \ldots, U_n)\}.D'. \\ \quad \{Tn(k),\ Goal(Y),\ TODO(\emptyset),\ Done(U_0, \ldots, U_n\}.D'')\end{array}}$$

COMPLETION then feeds ELIMINATION, the rule twinned with INTRODUCTION:

(7) ELIMINATION

$$\frac{(D.\{Tn(i),\ Goal(X),\ TODO(\beta),\ Done(\alpha,\ \langle d\rangle Y_0, \ldots, \langle d\rangle Y_n)\}.D')}{(D.\{Tn(i),\ Goal(X),\ TODO(\beta),\ Done(\alpha, Z_k)\}.D')}$$

where $Y_0, \ldots, Y_n \Rightarrow Z$

This rule effects the converse of Introduction, so that type specifications of subtasks are progressively derived.The rule schema $Y_0, \ldots, Y_n \Rightarrow Z$ is realised by a generalised Modus Ponens, where type-deduction is combined with the variant of function application described in section **2**:

(8) $$\frac{\langle d\rangle(Ty(a) \wedge Form(\phi)) \wedge \langle d\rangle(Ty(a \rightarrow b) \wedge Form(\psi))}{Ty(b) \wedge Form(\psi(\phi))}$$

The above rules between them define the state space of the parser, and various options are available for traversing that space. In the prototype program we have chosen to interleave the structure-building and interpretive rules, with a simple recursive definition shown below, whereby the rules are invoked in turn until the `goalstate` predicate succeeds. (This predicate attempts to call the `task` predicate with the goal `show(t)`, empty `ToDo` and with `ty(t)` as an element of `Done`). The order in which rules are applied is dictated in part by efficiency considerations. For instance, the **Scanning** rule is always called to check whether the type of the current word in the string matches the current task specification before invoking **Introduction** and **Subordination** to generate new tasks; this minimizes unwanted backtracking.

```
parse:-goalstate.
parse:-eliminate,parse.
parse:-complete,parse.
parse:-scan,parse.
parse:-subord,parse.
parse:-intro,parse.
parse:-backtrack,parse.
```

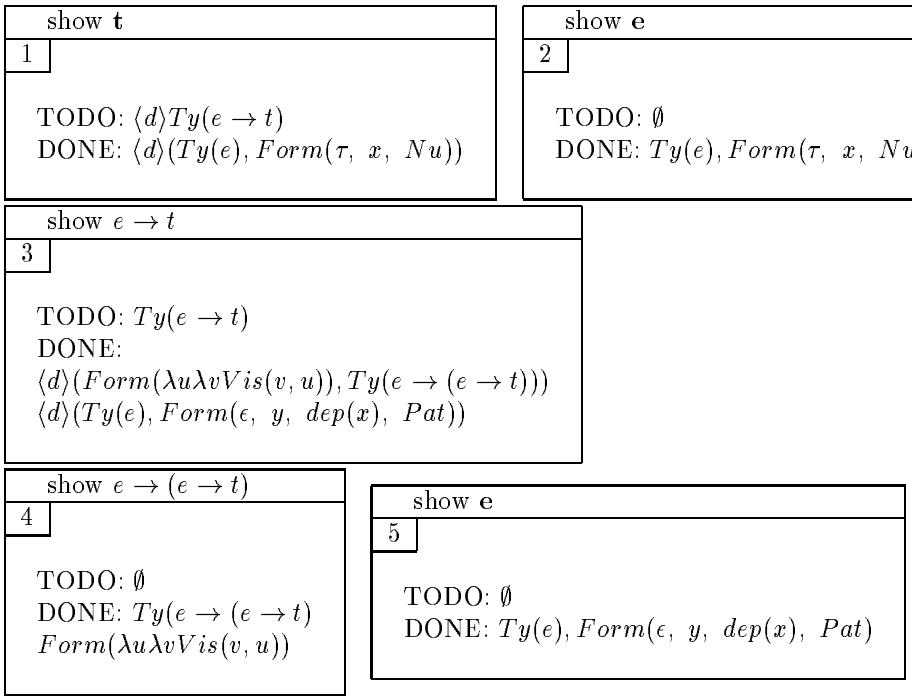
This inference regime results in a top-down parsing strategy, with backtracking over the **Introduction** rule until a task specification is generated which results in successful application of **Scanning**. An alternative approach would be to combine top-down and bottom-up techniques by making **Introduction** sensitive to the type(s) defined for the current word in the lexicon.

3.4 Example

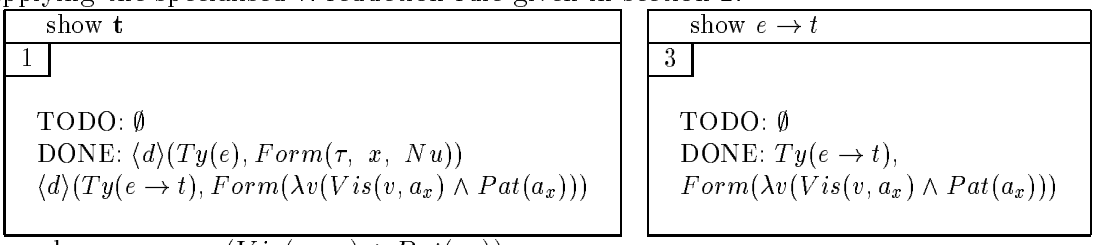
As an illustration we exhibit a sequence of snapshots of the parsing of *Every nurse visits a patient.*

Parse state 1: Axiom state `task(1, show(t), [ty(t)], [ ]).`

Parse state *i*: All words in the input string have been consumed. As a result of the interleaved inference regime **Elimination** and **Completion** have applied to the type `e` tasks.

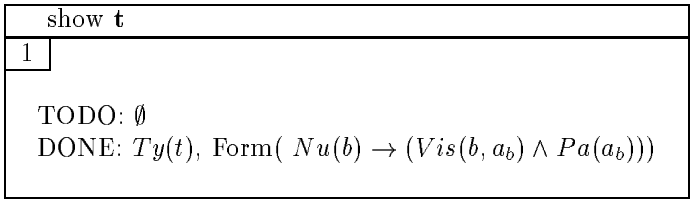


Parse state *i+2*: The rules of Elimination and Completion apply to tasks 1 and 3, applying the specialised λ-reduction rule given in section 2.



where $a_x = \epsilon y_x(Vis(v, y_x) \wedge Pat(y_x))$

Parse state *k*: The final step in deriving a formula of type **t** is application of Elimination to the top-level task Tn(1), again invoking the λ-reduction rule:



(. D')

where:

(i) $a(x) = \epsilon y(Vis(x, y) \wedge Pat(y))$

(ii) $b = \tau x(Nu(x) \rightarrow (Vis(x, a_x) \wedge Pa(a_x)))$

(iii) $a_b = \epsilon y(Vis(b, y) \wedge Pat(y))$

4 Conclusion

This LDS-NL model relates to categorial analyses manipulating labelled type deduction (eg Morrill 1994, Oehrle 1995, Joshi & Kulick 1995), though it is unlike these in addressing issues of underspecificity, and in its explicitly procedural perspective. It is close to semantic accounts of underspecificity (Alshawi & Crouch 1992, Reyle 1993, Farkas forthcoming), though unlike these formalisms, the process of resolving dependencies is assumed to be an on-line choice which is simultaneously context-dependent and nevertheless syntactic. So a semantics is defined only for the output of this process, not for the underspecified lexical input. The model is similar to the Candide system of Pereira & Pollack 1991 where the interpretation process divides into structure-building *conditional interpretation* rules and interpretive *discharge* rules, except that in the Candide system it is the tree-building operation which is deterministic, with decisions on quantifier scope and modifier attachment postponed to the semantic-pragmatic discharge phase.

The novelty of the present system lies in (a) the application of the epsilon calculus, and (b) its analysis of wide scope effects as a phenomenon falling together with anaphora. Its conceptual advantage over its competitors is that the indefinite is no longer seen as a determiner which is exceptional in virtue of having term-like properties; and its idiosyncratic freedom of scope choice follows directly from the anaphoric nature of the process resolving its lexical underspecification. Aspects of the research that are currently being developed are the precise nature of the relation between the concept of dependency implicit in the indexing and that defined within the epsilon calculus, the expansion of the system to incorporate dependencies on and between expressions denoting time, and the application to a broader range of quantifiers.

References

Abusch, D, 1994. 'The scope of indefinites' *Natural Language Semantics* 2.2, 83-135.
Alshawi, H & Crouch, R, 1992, 'Monotonic Semantic Interpretation' in *Proceedings ACL*
Blackburn, S. & Meyer-Viol, W. 1994 'Linguistics, logic and finite trees' *Bulletin of Interest Group in Pure and Applied Logics* Vol.2. No.1, 3-29.

Cooper, R. 1983. *Quantification and Syntactic Theory*. Reidel.
da Costa, N.C.A. 1980. 'A Model–Theoretic Approach to Variable Binding Term Operators, in A.I. Aruda, R. Chuaqui, N.C.A. da Costa (eds.), *Mathematical Logic in Latin America*, North Holland Publishing Company.
Farkas, D. 1981. Quantifier scope and syntactic islands.*CLS* 17, 59-66.
Farkas, D. forthcoming. Indexical Scope. in Sczabolsci, A. (ed.) *Ways of Scope-Taking*.
Finger, M. & Gabbay, D. 1993. Adding a temporal dimension to a logical system. *Journal of Logic, Language and Information* 1, 203-33
Finger, M., Kibble, R., Kempson, R. & Gabbay, D. 1996. 'Parsing natural language using LDS: A prototype' . Submitted to *Proceedings of WOLLIC '96*
Gabbay, D. 1990/fcmng. *Labelled Deductive Systems*. Oxford University Press.
Gabbay, D. & Kempson, R. 1992. 'Natural-language content: a proof-theoretic perspective' *Proceedings of 8th Amsterdam Semantics Colloquium*. Amsterdam.
Joshi, A. & Kulick S. 1995. 'Partial proof trees as building blocks for a categorial grammar' in Morrill, G. & Oehrle, R. (eds.) Proceedings of the Formal Grammar Conference, Barcelona July 1995.
Kempson. R. Meyer-Viol, W & Gabbay, D. forthcoming. 'Syntactic computation as labelled deduction'. In Borsley, R. & Roberts, I. (eds.) *Syntactic Categories*. Academic Press.
May, R. 1985. *Logical Form*. MIT Press.
Meyer Viol, W, 1995. Instantial Logic. PhD dissertation, Universiteit Utrecht. Milward, D. 1993. Dynamics. dependency grammar and incremental interpretation *COLING 14*. 1095-9
Montague, R. 1974. *Formal Philosophy*, ed. R.Thomason. Yale University Press.
Morrill, G. 1994. *Type-logical Grammar*. Kluwer.
Pereira, F. 1990. Categorial semantics and scoping. *Computational Linguistics* 16, 1-10.
Pereira, F. & Pollack, M. 1991 'Incremental interpretation'. *Artificial Intelligence* 50, 37-82.
Reyle, U, 1993. 'Dealing with Ambiguities by Underspecification' , *Journal of Semantics* 10.
Shieber , S, Schabes, Y & FCN Pereira, 1994. Principles and Implementation of Deductive Parsing. Technical Report TR-11-94, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University.