Interleaved semantic interpretation in environment-based parsing[*]

William Schuler
Computer and Information Science Dept.
University of Pennsylvania
Philadelphia, PA 19103
schuler@linc.cis.upenn.edu

Abstract

This paper extends a polynomial-time parsing algorithm that resolves structural ambiguity in input sentences by calculating and comparing the denotations of rival constituents, given some model of the application environment (Schuler, 2001). The algorithm is extended to incorporate a full set of logical operators, including quantifiers and conjunctions, into this calculation without increasing the complexity of the overall algorithm beyond polynomial time, both in terms of the length of the input and the number of entities in the environment model.

1 Introduction

The development of speaker-independent mixed-initiative speech interfaces, in which users not only answer questions but also ask questions and give instructions, is currently limited by the inadequacy of existing corpus-based disambiguation techniques. This paper explores the use of semantic and pragmatic information, in the form of the entities and relations in the interfaced application's run-time environment, as an additional source of information to guide disambiguation.

In particular, this paper extends an existing parsing algorithm that calculates and compares the denotations of rival parse tree constituents in order to resolve structural ambiguity in input sentences (Schuler, 2001). The algorithm is extended to incorporate a full set of logical operators into this calculation so as to improve the accuracy of the resulting denotations – and thereby improve the accuracy of parsing – without increasing the complexity of the overall algorithm beyond polynomial time (both in terms of the length of the input and the number of entities in the environment model). This parsimony is achieved by localizing certain kinds of semantic relations during parsing, particularly those between quantifiers and their restrictor and body arguments

(similar to the way dependencies between predicate and argument head words are localized in lexicalized formalisms such as tree adjoining grammars), in order to avoid calculating exponential higher-order denotations for expressions like generalized quantifiers.

2 Basic algorithm

This section describes the basic environment-based parser (Schuler, 2001) which will be extended in Section 3. Because it will crucially rely on the denotations (or interpretations) of proposed constituents in order to guide disambiguation, the parser will be defined on categorial grammars (Ajdukiewicz, 1935; Bar-Hillel, 1953), whose categories all have well defined types and worst-case denotations. These categories are drawn from a minimal set of symbols $\mathcal{C}$ such that:

$$\text{NP} \in \mathcal{C} \text{ and S} \in \mathcal{C},$$
$$\text{if } \gamma, \delta \in \mathcal{C} \text{ then } \gamma/\delta \in \mathcal{C} \text{ and } \gamma\backslash\delta \in \mathcal{C}.$$

Intuitively, the category NP describes a noun phrase and the category S describes a sentence, and the complex categories γ/δ and $\gamma\backslash\delta$ describe 'a γ lacking a δ to the right' and 'a γ lacking a δ to the left' respectively; so for example S\NP would describe a declarative verb phrase lacking an NP subject to its left in the input.

The type T and worst-case (most general) denotation W of each possible category are defined below, given a set of entities $\mathcal{E}$ as an environment:

$$
\begin{array}{ll}
T(\text{S}) = t : \text{truth value} & W(\text{S}) = \{\text{TRUE}, \text{FALSE}\} \\
T(\text{NP}) = e : \text{entity} & W(\text{NP}) = \mathcal{E} \\
T(\gamma/\delta) = \langle T(\delta), T(\gamma)\rangle & W(\gamma/\delta) = W(\delta) \times W(\gamma) \\
T(\gamma\backslash\delta) = \langle T(\delta), T(\gamma)\rangle & W(\gamma\backslash\delta) = W(\delta) \times W(\gamma)
\end{array}
$$

The denotation D of any proposed constituent is constrained to be a subset of the worst-case denotation W of the constituent's category; so a constituent of category NP would denote a set of entities, $\{e_1, e_2, \dots\}$, and a constituent of category S\NP would denote a set of entity $\times$ truth value pairs, $\{\langle e_1, \text{TRUE}\rangle, \langle e_2, \text{FALSE}\rangle, \dots\}$. Note that no denotation of a constituent can contain more than

---

$\mathcal{O}(|\mathcal{E}|^v)$ different elements, where v is a valency measure of the number of NP symbols occurring within the constituent's category.

This paper will use the following definition of a categorial grammar (CG):

Definition A *categorial grammar* G is a formal grammar (N, Σ, P) such that:

- Σ is a finite set of words w;
- P is a finite set of productions containing:
  $\gamma \to w$ for all $w \in \Sigma$, with $\gamma \in \mathcal{C}$,
  $\gamma \to \gamma/\delta \;\; \delta$ for every rule $\gamma/\delta \to \ldots$ in P,
  $\gamma \to \delta \;\; \gamma\backslash\delta$ for every rule $\gamma\backslash\delta \to \ldots$ in P,
  and nothing else;
- N is the nonterminal set $\{\gamma \mid \gamma \to \ldots \in P\}$.

and the following deductive parser,[1] which will be extended later to handle a richer set of semantic operations. The parser is defined with:

- constituent chart items $[i, j, \gamma]$ drawn from $\mathbb{I}_0^n \times \mathbb{I}_0^n \times N$, indicating that positions i through j in the input can be characterized by category γ;
- a lexical item $[i, j, \gamma]$ for every rule $\gamma \to w \in P$ if w occurs between positions i and j in the input;
- a set of rules of the form:
  $\frac{[i,k,\gamma/\delta] \; [k,j,\delta]}{[i,j,\gamma]}$ for all $\gamma \to \gamma/\delta \;\; \delta \in P$, $i, j, k \in \mathbb{I}_0^n$,
  $\frac{[k,j,\gamma\backslash\delta] \; [i,k,\delta]}{[i,j,\gamma]}$ for all $\gamma \to \delta \;\; \gamma\backslash\delta \in P$, $i, j, k \in \mathbb{I}_0^n$.

and can recognize an n-length input as a constituent of category γ (for example, as an S) if it can deduce the chart item $[0, n, \gamma]$.

This parser can be implemented in a dynamic programming algorithm, using the recursive function:

$$F(x) = \bigvee_{a_1 \ldots a_k \; s.t. \; \frac{a_1 \ldots a_k}{x}} \bigwedge_{i=1}^{k} F(a_i)$$

(where $x, a_1 \ldots a_k$ are proposed constituents drawn from $\mathbb{I}_0^n \times \mathbb{I}_0^n \times N$, $\bigvee_\emptyset = \text{FALSE}$, and $\bigwedge_\emptyset = \text{TRUE}$), by recording the result of every recursive sub-call to $F(x)$ in a chart, then consulting this chart on subsequent calls to $F(x)$ for the same x constituent.[2] Since the indices in every rule's antecedent constituents $a_1 \ldots a_k$ each cover smaller spans than those in the consequent x, the algorithm will not enter into an infinite recursion; and since there are only $n^2|N|$ different values of x, and only $2n$ different rules that could prove any consequent x (two rule forms for $/$ and $\backslash$, each with n different values of k), the algorithm runs in polynomial time: $\mathcal{O}(n^3|N|)$. The resulting chart can then be annotated with back pointers to produce a polynomial-sized shared forest

[1] Following Shieber et al. (1995).
[2] Following Goodman (1999).

representation of all possible grammatical trees (Billot and Lang, 1989).

Traditional corpus-based parsers select preferred trees from such forests by calculating Viterbi scores for each proposed constituent, according to the recursive function:

$$S_V(x) = \max_{a_1 \ldots a_k \; s.t. \; \frac{a_1 \ldots a_k}{x}} \left(\prod_{i=1}^{k} S_V(a_i) \right) \cdot \mathsf{P}(a_1 \ldots a_k \mid x)$$

These scores can be calculated in polynomial time, using the same dynamic programming algorithm as that described for parsing. A tree can then be selected, from the top down, by expanding the highest-scoring rule application for each constituent.

The environment-based parser described here uses a similar mechanism to select preferred trees, but the scores are based on the presence or absence of entities in the denotation (interpretation) of each proposed constituent:[3]

$$S_D(x) = \max_{a_1 \ldots a_k \; s.t. \; \frac{a_1 \ldots a_k}{x}} \left(\sum_{i=1}^{k} S_D(a_i) \right) + \begin{cases} 1 & \text{if } D(x) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

where the denotation $D(x)$ of a proposed constituent x is calculated using another recursive function:

$$D(x) = \bigcup_{a_1 \ldots a_k \; s.t. \; \frac{a_1 \ldots a_k}{x}} \left(\pi \bowtie_{i=1}^{k} D(a_i) \right) \bowtie \begin{cases} R(x) & \text{if } k = 0 \\ \{\langle\rangle\} & \text{otherwise} \end{cases}$$

in which $R(x)$ is a lexical relation defined for each axiom x of category γ equal to some subset of γ's worst-case denotation $W(\gamma)$, as defined above.[4] The operator $\bowtie$ is natural (relational) join on the fields of its operands:

$$A \bowtie B = \{ \langle e_1 \ldots e_{max(a,b)} \rangle \mid \langle e_1 \ldots e_a \rangle \in A, \langle e_1 \ldots e_b \rangle \in B \}$$

where $a, b \geq 0$; and π is a projection that removes the first element of the result (corresponding the most recently discharged argument of the head or functor category):

$$\pi A = \{ \langle e_2 \ldots e_a \rangle \mid \langle e_1 \ldots e_a \rangle \in A \}$$

This interleaving of semantic evaluation and parsing for the purpose of disambiguation has much in common with that of Dowding et al. (1994), except

[3] Here, the score is simply equal to the number of non-empty constituents in an analysis, but other metrics are possible.
[4] So a lexical relation for the constituent 'lemon' of category NP would contain all and only the lemons in the environment, and a lexical relation for the constituent 'falling' of category S\NP would contain a mapping from every entity in the environment to some truth value (TRUE if that entity is falling, FALSE otherwise): e.g. $\{\langle lemon_1, TRUE \rangle, \langle lemon_2, FALSE \rangle, \ldots \}$.

NP[lemon]
$\{l_1\} \cup \emptyset$

PP:NP\NP[in]
$\{\langle l_1, l_1 \rangle\}$

NP[lemon]
$\{l_1\}$

NP[bin]
$\{b_1, b_2\}$

PP:NP\NP[in]
$\{\langle l_1, l_1 \rangle\}$

PP:NP\NP[by]
$\{\langle b_1, b_1 \rangle, \langle b_2, b_2 \rangle\}$

NP[lemon]
$\{l_1, l_2, l_3, l_4\}$

P:NP\NP/NP[in]
$\{\langle b_1, \langle l_1, l_1 \rangle \rangle, \langle m_1, \langle l_2, l_2 \rangle \rangle\}$

NP[bin]
$\{b_1, b_2\}$

P:NP\NP/NP[by]
$\{\langle m_1, \langle b_1, b_1 \rangle \rangle, \langle m_2, \langle b_2, b_2 \rangle \rangle\}$
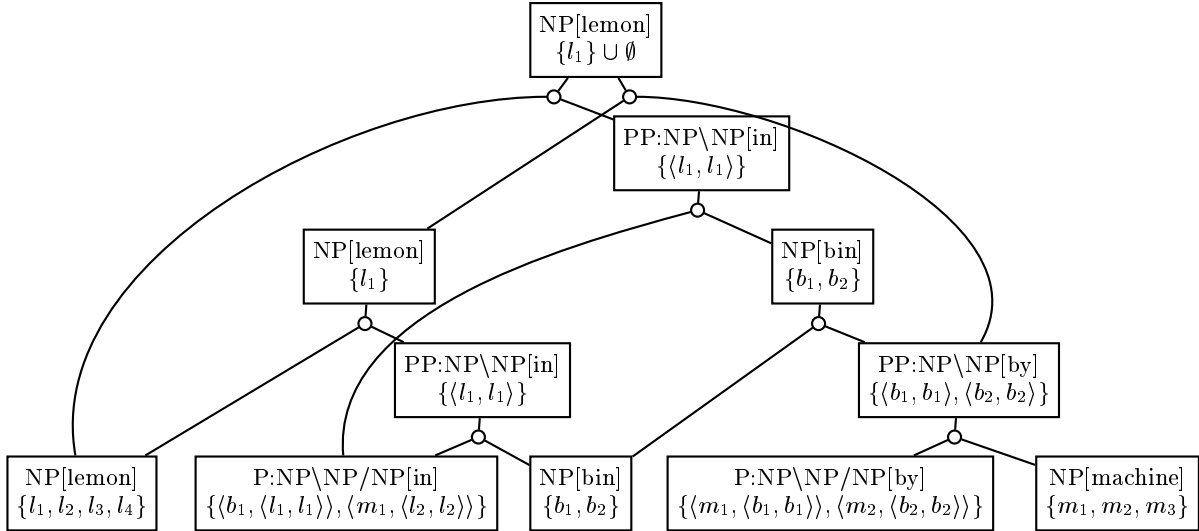
NP[machine]
$\{m_1, m_2, m_3\}$

Figure 1: Denotation-annotated forest for 'lemon in bin by machine.'

that in this case, constituents are not only semantically type-checked, but are also fully interpreted each time they are proposed.

Figure 1 shows a sample denotation-annotated forest for the phrase 'the lemon in the bin by the machine', using the lexicalized grammar:

$$\text{lemon, bin, machine : NP}$$
$$\text{the : NP/NP}$$
$$\text{in, by : NP\backslash NP/NP}$$

in which the denotation of each constituent (the set in each rectangle) is calculated using a join on the denotations of each pair of constituents that combine to produce it. In this example, the right-branching tree would be preferred because the denotation resulting from the composition at the root of the other tree would be empty.

Since this use of the join operation is linear on the sum of the cardinalities of its operands, and since the denotations of the categories in a grammar G are bounded in cardinality by $\mathcal{O}(|\mathcal{E}|^v)$ where v is the maximum valency of the categories in G, the total complexity of the above algorithm can be shown to be $\mathcal{O}(n^3 |\mathcal{E}|^v)$: polynomial not only on the length of the input n, but also on the size of the environment $\mathcal{E}$ (Schuler, 2001).

3 Extended algorithm

The above algorithm works well for attaching ordinary complements and modifiers, but as a semantic theory it is not sufficiently expressive to produce correct denotations in all cases. For example, the lexical relations defined above are insufficient to represent quantifiers like 'no' (using category NP/NP) in the phrase 'the boy with no backpack.'[5] A similar problem occurs with conjunctions; for example, the word 'and' (using category NP\NP/NP) in the phrase 'the child wearing glasses and blue pants', also cannot be properly represented as a lexical relation.[6] This raises the question: how much expressivity can be allowed in a shared semantic interpretation without exceeding the tractable parsing complexity necessary for practical environment-based parsing?

In traditional categorial semantics (Montague, 1973; Barwise and Cooper, 1981; Keenan and Stavi, 1986) quantifiers and noun phrase conjunctions denote higher-order relations: that is, relations between whole sets of entities instead of just between individuals. Under this interpretation, a quantifier like 'no' would denote a set of pairs $\{\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle, \dots\}$ where each A_i and B_i are disjoint subsets of $\mathcal{E}$, corresponding to an acceptable pair of restrictor and body sets satisfying the quantifier 'no'. Unfortunately, since the cardinalities of these higher-order denotations can be exponential on the size of the environment $\mathcal{E}$ (there are $2^{|\mathcal{E}|}$ possible subsets of $\mathcal{E}$ and $2^{2|\mathcal{E}|}$ possible combinations of two such subsets), such an approach would destroy the polynomial complexity of the environment-based parsing algorithm.

---

[5] Assigning the identity relation $\{\langle e_1, e_1 \rangle, \langle e_2, e_2 \rangle, \dots\}$ to the quantifier would incorrectly yield the set of boys *with* a backpack as a denotation for the full noun phrase; and assigning the converse relation (from each entity in the environment to every other entity $\{\langle e_1, e_2 \rangle, \langle e_1, e_3 \rangle, \dots\}$) would incorrectly yield the set of boys with anything that is not a backpack.

[6] The identity relation $\{\langle e_1, e_1, e_1 \rangle, \langle e_2, e_2, e_2 \rangle, \dots\}$, which yields a correct interpretation in verb phrase conjunction, would yield an incorrect denotation for the noun phrase 'glasses and blue pants,' containing only entities which are at once both glasses and pants.

However, if the number of possible higher-order functions is restricted to a finite set (say, to some subset of words in a lexicon), it becomes tractable to store them by name rather than by denotation (i.e. as sets). Such function can then discharge all their first-order arguments in a single derivational step to produce a first-order result, in order to avoid generating or evaluating any higher-order partial results. Syntactically, this would be analogous to composing a quantifier with both a noun phrase restrictor and a body predicate (e.g. a verb or verb phrase) at the same time, to produce another first-order predicate (e.g. a verb phrase or sentence). Since a generalized quantifier function merely counts and compares the cardinalities of its arguments in a linear time operation, this analysis provides a tractable shortcut to the exponential calculations required in the conventional analysis.

Note that this analysis by itself does not admit productive modification of quantifiers (because their functions are drawn from some finite set) or of quantified noun phrases (because they are no longer derived as a partial result). This causes no disruption to the attachment of non-conjunctive modifiers, because ordinary syntactic modifiers of quantifier constituents are seldom productive (in the sense that their composition does not yield functions outside some finite set), and syntactic modifiers of NP constituents usually only modify the restrictor set of the quantifier rather than the entire quantified function, and can therefore safely be taken to attach below the quantifier, to the unquantified NP.

But this is not true in cases involving conjunction. Conjoined quantifiers, like 'some but not all,' cannot always be defined using a single standard lexical function; and conjunctions of quantified noun phrases, like 'one orange and one lemon', cannot be applied to unquantified subconstituents (syntactically, because this would fail to subsume the second quantifier, and semantically, because it is not the restrictor sets which are conjoined). Keenan and Stavi (1986) model conjunctions of quantifiers and quantified noun phrases using lattice operations on higher-order sets, but as previously stated, these higher-order sets preclude tractable interleaving of semantic interpretation with parsing.

The solution proposed here is to treat each quantifier or quantified noun phrase conjunction as an elliptical conjunction of two complete first-order predicates (e.g. verb phrases or sentences), each subsuming a different quantifier and noun phrase restrictor (in the case of NP conjunction), but sharing or *duplicating* a common body predicate. This analysis requires multiple components to keep track of the duplicated material above the conjunction, but as long as the number of components is bounded, the polynomial complexity of the parsing algorithm is
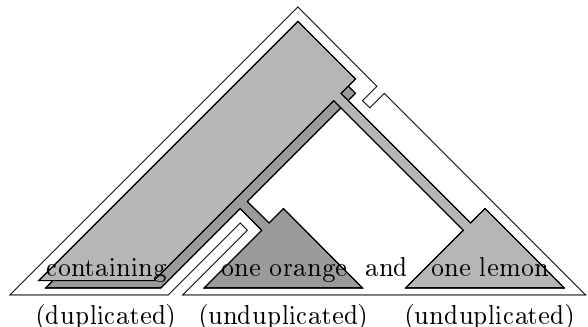

(duplicated)   (unduplicated)   (unduplicated)

Figure 2: Duplicated verb in NP conjunction.

retained.[7]

Figure 2 shows a duplicated verb predicate in the derivation of an NP conjunction. The conjoined constituents (the shaded regions in the figure) are each composed of two components: one for the NP itself, containing the quantifier and the restrictor predicate, and one for the verb which supplies the body predicate of the quantifier. Since the conjoined constituents both correspond to complete quantifier expressions with no unsatisfied first-order arguments, their categories are that of simple first-order predicates (they are each complete verb phrases in essence: 'containing one orange' and 'containing one lemon'). The conjunction then forms a larger constituent of the same form (the unshaded outline in the figure), with a lower component containing the conjoined constituents' NP components concatenated in the usual way, and an upper component in which the conjoined constituents' non-NP components are identified or overlapped. If the duplicated components do not cover the same string yield, the conjunction does not apply.

Note that, since they are only applied to ordinary first-order predicates (e.g. sentences or verb phrases) in this analysis, conjunctions can now safely be assigned the familiar truth-functional denotations in every case.[8] Also, since the resulting constituent has the same number of components as the conjoined constituents, there is nothing to prevent its use as an argument in subsequent conjunction operations.

A sample multi-component analysis for quantifiers is shown below, allowing material to be duplicated both to the left and to the right of a conjoined NP:

$$\text{some,all,no,etc.} : X\backslash\text{NP}_q \cdot \text{NP}_q\backslash\text{NP}_q \cdot \text{NP}_q/\text{NP}_\epsilon$$
$$X/\text{NP}_q \cdot \text{NP}_q\backslash\text{NP}_q \cdot \text{NP}_q/\text{NP}_\epsilon$$

The lexical entry for a quantifier can be split in this

---

[7] Dahl and McCord (1983) propose a similar duplication mechanism to produce appropriate semantic representations for NP and other conjunctions, but for different reasons.

[8] e.g. for the word 'and': {⟨...TRUE, ...TRUE, ...TRUE⟩, ⟨..TRUE, ..FALSE, ..FALSE⟩, ⟨..FALSE, ..TRUE, ..FALSE⟩, ⟨..FALSE, ..FALSE, ..FALSE⟩}

way into a number of components, the last (or lowest) of which is not duplicated in conjunction while others may or may not be. These include a component for the quantifier NP_q/NP_ϵ (which will ultimately also contain a noun phrase restrictor of category NP_ϵ), a component for restrictor PPs and relative clauses of category $NP_q \backslash NP_q$ that are attached above the quantifier and duplicated in the conjunction, and a component for the body (a verb or verb phrase or other predicate) of category $X \backslash NP_q$ or X/NP_q. The subscript q specifies one of a finite set of quantifiers, and the subscript ϵ indicates an unquantified NP.

The deductive parser presented in Section 2 can now be extended by incorporating sequences of recognized and unrecognized components into the constituent chart items. As constituents are composed, components are shifted from the unrecognized sequence $\gamma_1 \cdots \gamma_c$ to the recognized sequence $\langle i_1, j_1, \gamma_1 \rangle \cdots \langle i_c, j_c, \gamma_c \rangle$, until the unrecognized sequence is empty.

The extended parser is defined with:

- chart items of the form $[i, j, \Delta, \Sigma]$, where Δ is a sequence of unrecognized components γ, Σ is a sequence of recognized components $\langle a, b, \gamma \rangle$, and i, j, k, a, b, c are indices in the input. Each item $[i, j, \Delta \cdot \gamma, \langle i_1, j_1, \gamma_1 \rangle \cdots \langle i_c, j_c, \gamma_c \rangle]$ indicates that the span from i to j in the input can be characterized by the categories γ_1 through γ_c at positions i_1 to j_1 through i_c to j_c respectively, so that if these spans are concatenated in whatever order they occur in the input string, they form a grammatical constituent of category γ with unrecognized components Δ.

- a lexical item $[i, j, \gamma, \langle i, j, \gamma \rangle]$ for every rule $\gamma \rightarrow w \in P$ if w occurs between positions i and j in the input;

- a set of rules for all $i, j, k, a, b, c \in \mathbb{I}_0^n$ as below. Two rules to invoke left and right function application to an existing component:

$$\frac{[i,k,\gamma/\delta,\langle i,k,\gamma/\delta\rangle]\ [k,j,\Delta\cdot\delta,\langle k,b,\delta/\varepsilon\rangle\cdot\Sigma]}{[i,j,\Delta\cdot\gamma,\langle i,b,\gamma/\varepsilon\rangle\cdot\Sigma]}\gamma\rightarrow\gamma/\delta\ \ \delta\in P,$$

$$\frac{[k,j,\gamma\backslash\delta,\langle k,j,\gamma\backslash\delta\rangle]\ [i,k,\Delta\cdot\delta,\langle a,k,\delta\backslash\varepsilon\rangle\cdot\Sigma]}{[i,j,\Delta\cdot\gamma,\langle a,j,\gamma\backslash\varepsilon\rangle\cdot\Sigma]}\gamma\rightarrow\delta\ \ \gamma\backslash\delta\in P,$$

Two rules to invoke left and right function application to a fresh component:

$$\frac{[i,k,\gamma/\delta,\langle i,k,\gamma/\delta\rangle]\ [k,j,\Delta\cdot\gamma/\delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\gamma,\langle i,k,\gamma/\delta\rangle\cdot\Sigma]}\gamma\rightarrow\gamma/\delta\ \ \delta\in P,$$

$$\frac{[k,j,\gamma\backslash\delta,\langle k,j,\gamma\backslash\delta\rangle]\ [i,k,\Delta\cdot\gamma\backslash\delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\gamma,\langle k,j,\gamma\backslash\delta\rangle\cdot\Sigma]}\gamma\rightarrow\delta\ \ \gamma\backslash\delta\in P,$$

Two rules to discharge empty components:

$$\frac{[i,j,\Delta\cdot\gamma/\delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\gamma,\Sigma]}\qquad\frac{[i,j,\Delta\cdot\gamma\backslash\delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\gamma,\Sigma]}$$

Three rules to skip conjunctions, by adding a gap between the components in a constituent

(the first rule consumes the conjunction to create a partial result of category Conj'_δ, and the latter two use this to skip the opposing NP):

$$\frac{[k,j,\Delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\text{Conj}'_\delta,\Sigma]}[i,k,\text{Conj},\langle i,k,\text{Conj}\rangle]$$

$$\frac{[k,j,\Delta\cdot\text{Conj}'_\delta,\Sigma]}{[i,j,\Delta\cdot\delta,\Sigma]}[i,k,\Delta\cdot\delta,\_]\qquad\frac{[i,k,\Delta\cdot\delta,\Sigma]}{[i,j,\Delta\cdot\delta,\Sigma]}[k,j,\Delta\cdot\text{Conj}'_\delta,\_]$$

Two rules to reassemble discontinuous constituents (again, using a partial result Conj'_δ to reduce the number of ranging variables):

$$\frac{[a,c,\text{Conj},\langle a,c,\text{Conj}\rangle]\ [i,j,\gamma,\Sigma\cdot\langle c,b,\delta\rangle]}{[i,j,\gamma,\Sigma\cdot\langle a,b,\text{Conj}'_\delta\rangle]}$$

$$\frac{[i,j,\gamma,\Sigma\cdot\langle c,b,\text{Conj}'_\delta\rangle]\ [i,j,\gamma,\Sigma\cdot\langle a,c,\delta\rangle]}{[i,j,\gamma,\Sigma\cdot\langle a,b,\delta\rangle]}$$

Two rules to combine adjacent components:

$$\frac{[i,j,\gamma,\Sigma\cdot\langle a,c,\delta/\varepsilon\rangle\cdot\langle c,b,\varepsilon\rangle]}{[i,j,\gamma,\Sigma\cdot\langle a,b,\delta\rangle]}\qquad\frac{[i,j,\gamma,\Sigma\cdot\langle c,b,\delta\backslash\varepsilon\rangle\cdot\langle a,c,\varepsilon\rangle]}{[i,j,\gamma,\Sigma\cdot\langle a,b,\delta\rangle]}$$

And one rule to apply quantifier functions:

$$\frac{[i,j,\gamma,\Sigma\cdot\langle a,b,\delta_q\rangle]}{[i,j,\gamma,\Sigma\cdot\langle a,b,\delta_\epsilon\rangle]}$$

The parsing and scoring functions remain identical to those in Section 2, but an additional $k = 1$ case containing a modified projection function π is now added to the interpretation function, in order to make the denotations of quantified constituents depend on their associated quantifiers:

$$D(x) = \bigcup_{a_1\ldots a_k\ s.t.\ \frac{a_1\ldots a_k}{x}} \begin{cases} R(x) & \text{if } k = 0 \\ \pi_q D(a_1) & \text{if } k = 1 \text{ and} \\ & \frac{a_1}{x} = \frac{[\ldots\langle\ldots\delta_q\rangle]}{[\ldots\langle\ldots\delta_\epsilon\rangle]} \\ \overset{k}{\underset{i=1}{\bowtie}} D(a_i) & \text{otherwise} \end{cases}$$

The modified projection function evaluates a quantifier function q on some argument denotation A, comparing the cardinality of the image of the restrictor set in A with the the cardinality of image of the intersected restrictor and body sets in A:[9]

$$\pi_q A = \{\langle e_2\ldots e_a, t\rangle \mid \langle\_, e_2\ldots e_a,\_\rangle \in A,\ t = q(|R|,|S|)$$
$$R = A \bowtie \{\langle\_, e_2\ldots e_a,\_\rangle\},$$
$$S = A \bowtie \{\langle\_, e_2\ldots e_a,\text{TRUE}\rangle\}\ \}$$

This algorithm parses a categorial grammar in the usual way – constituents are initially added to the chart as single components covering a certain yield in the input string (the indices of the component are the same as the indices of the constituent itself), and they are combined by concatenating the yields of smaller constituents to make larger ones – until a conjunction is encountered. When a conjunction is

---

[9] Following Keenan and Stavi (1986).

$$0 \quad \underline{\textit{containing}} \quad 1 \quad \underline{\textit{one orange}} \quad 2 \quad \underline{\textit{and}} \quad 3 \quad \underline{\textit{one lemon}} \quad 4$$

$[0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_{q'},$ $\quad\quad [1, 2, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists,$ $\quad\quad [2, 3, \mathrm{Conj},$ $\quad\quad [3, 4, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists,$

$\quad\quad \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_{q'}\rangle]$ $\quad\quad \langle 1, 2, \mathrm{NP}_\exists\rangle]\ \{o_1, o_2, o_3, o_4\}$ $\quad\quad \langle 2, 3, \mathrm{Conj}\rangle]$ $\quad\quad \langle 3, 4, \mathrm{NP}_\exists\rangle]\ \{l_1, l_2, l_3\}$

$\{\langle o_1, x_1\rangle, \langle l_2, x_1\rangle, \langle l_3, x_3\rangle\}$

$$[1, 4, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists, \langle 1, 2, \mathit{NP}_\exists\rangle]\ \{o_1, o_2, o_3, o_4\} \tag{1}$$

$$[1, 4, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists, \langle 3, 4, \mathit{NP}_\exists\rangle]\ \{l_1, l_2, l_3\} \tag{2}$$

$$[1, 4, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists, \langle 1, 2, \mathit{NP}_\exists\rangle]\ \{o_1, o_2, o_3, o_4\} \tag{3}$$

$$[1, 4, X\backslash\mathrm{NP}_\exists \cdot \mathrm{NP}_\exists, \langle 3, 4, \mathit{NP}_\exists\rangle]\ \{l_1, l_2, l_3\} \tag{4}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_\exists\rangle \cdot \langle 1, 2, \mathrm{NP}_\exists\rangle]\ \{\langle o_1, x_1\rangle\} \tag{5}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_\exists\rangle \cdot \langle 3, 4, \mathrm{NP}_\exists\rangle]\ \{\langle l_2, x_1\rangle, \langle l_3, x_3\rangle\} \tag{6}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_\epsilon\rangle \cdot \langle 1, 2, \mathrm{NP}_\epsilon\rangle]\ \{x_1\} \tag{7}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_\epsilon\rangle \cdot \langle 3, 4, \mathrm{NP}_\epsilon\rangle]\ \{x_1, x_3\} \tag{8}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 1, \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_\epsilon\rangle \cdot \langle 1, 4, \mathrm{NP}_\epsilon\rangle]\ \{x_1\} \tag{9}$$

$$[0, 4, \mathrm{S}\backslash\mathrm{NP}_q, \langle 0, 4, \mathrm{S}\backslash\mathrm{NP}_q\rangle]\ \{x_1\} \tag{10}$$
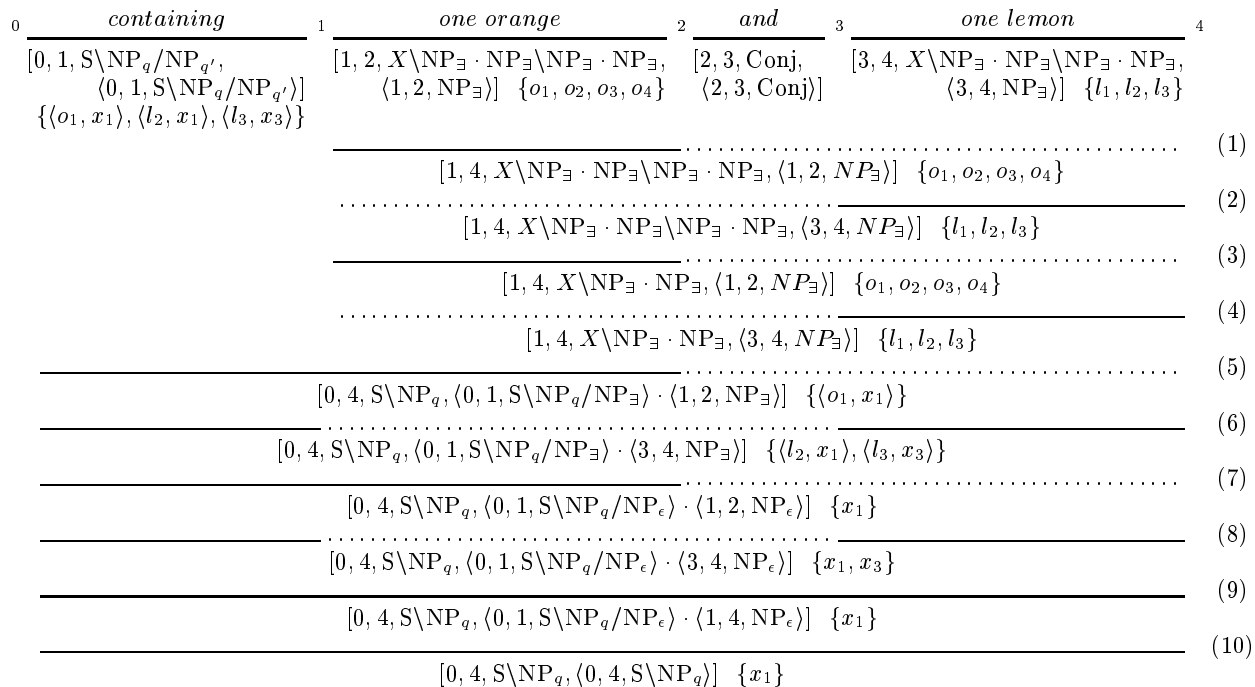
Figure 3: Sample derivation of conjoined NP.

encountered immediately to the left or right of a recognized constituent constituent x, and another constituent of the same category is found immediately beyond that conjunction, the parser creates a new constituent that has the combined yield of both constituents, but copies x's component yield (the string indices of x's original components) with no change. This has the effect of creating two new constituents every time two existing constituents are conjoined: each with a different component yield, but both with the same (combined) constituent yield. These new discontinuous constituents (with component yields that do not exhaust their constituent yields) are still treated as ordinary constituents by the parser, which combines them with arguments and modifiers until all of their argument positions have been successfully discharged, at which point pairs of discontinuous constituents with the same constituent yield can be reassembled into whole – or at least less discontinuous – constituents again.

A sample derivation for the verb phrase 'containing one orange and one lemon,' involving conjunction of existentially quantified noun phrases, is shown in Figure 3, using the above parse rules and the lexicalized grammar:

$$\begin{aligned}
\text{containing} &: \mathrm{S}\backslash\mathrm{NP}_q/\mathrm{NP}_{q'} \\
\text{one} &: X\backslash\mathrm{NP}_q \cdot \mathrm{NP}_q\backslash\mathrm{NP}_q \cdot \mathrm{NP}_q/\mathrm{NP}_\epsilon \\
&\ X/\mathrm{NP}_q \cdot \mathrm{NP}_q\backslash\mathrm{NP}_q \cdot \mathrm{NP}_q/\mathrm{NP}_\epsilon \\
\text{orange, lemon} &: \mathrm{NP}_\epsilon \\
\text{and} &: \mathrm{Conj}
\end{aligned}$$

First the parser applies the *skip conjunction* rules to obtain the discontinuous constituents shown after steps (1) and (2), and a component is discharged from each of the resulting constituents using the *empty component* rule in steps (3) and (4). The constituents resulting from (3) and (4) are then composed with the verb constituent for 'containing' in steps (5) and (6), using the *left attachment rule for fresh components*. The quantifiers are then applied in steps (7) and (8), and the resulting constituents are reassembled using the *conjunction rules* in step (9). The adjacent components in the constituent resulting from step (9) are then merged using the *combination rule* in step (10), producing a complete gapless constituent for the entire input.

Since the parser rules are fixed, and the number of components in any chart constituent is bounded by the maximum number of components in a category (inasmuch as the rules can only add a component to the recognized list by subtracting one from the unrecognized list), the algorithm must run in polynomial space and time on the length of the input sentence. Since the cardinality of each constituent's denotation is bounded by $|\mathcal{E}|^v$ (where $\mathcal{E}$ is the set of entities in the environment and v is the maximum valency of any category), the algorithm runs in worst-case polynomial space on $|\mathcal{E}|$; and since there is no more than one set composition operation performed when a rule is applied, and each composition operation runs in worst-case quadratic time on the

size of its composed sets (due to the quantifier operation), the algorithm runs in worst-case polynomial time on $|\mathcal{E}|$ as well.

4   Evaluation

The extended parser described above has been implemented and evaluated on a corpus of 340 spoken instructions to simulated human-like agents in a controlled 3-D environment (that of children running a lemonade stand, which was deemed suitably familiar to undergraduate student subjects). The parser was run on the word lattice output of an off-the-shelf speech recognizer (CMU Sphinx II) and the parser chart was seeded with every hypothesized word. The parser was also compared with the recognizer by itself, in order to determine the degree to which an environment-based approach could complement corpus-based disambiguation. The systems were evaluated as word recognizers (i.e. ignoring the brackets in the parser output) on the first 100 sentences of the corpus (corresponding to the first seven of 33 subjects); the latter 240 sentences were reserved for training the recognizer and for developing the grammar and semantic lexicon.

The average utterance length was approximately three seconds (subsuming about 300 frames or positions in the parser chart), containing an average of nine words. Parsing time averaged under 40 seconds per sentence on a P4-1500MHz, most of which was spent in forest construction rather than denotation calculation.

Accuracy results show that the parser was able to correctly identify a significant number of words that the recognizer missed (and vice versa), such that a perfect synthesis of the two (choosing the correct word if it is recognized by either system) would produce an average of 8 percentage points more recall than the recognizer by itself on successful parses, and as much as 19 percentage points more for some subjects:[10]

	recognizer		parser			joint
subject	prec	recall	fail	prec	recall	recall
0	76	79	18	72	74	92
1	77	75	28	63	55	83
2	70	71	33	49	54	69
3	71	67	43	49	45	69
4	66	54	37	44	39	67
5	53	52	54	36	31	72
6	84	84	50	56	63	83
all	68	67	37	53	50	75

which indicates that the environment may offer a useful additional source of information for disambiguation. Though it may not be possible to implement a perfect synthesis of the environment-based

and corpus-based approaches, if even half of the above gains can be realized, it would mark a significant advance.

5   Conclusion

This paper has described an extension to an environment-based parsing algorithm, increasing its semantic coverage to include quantifier and conjunction operations without destroying its polynomial worst-case complexity. Experiments using an implementation of this algorithm on a corpus of spoken instructions indicate that 1) the observed complexity of the algorithm is suitable for practical user interface applications, and 2) the ability to draw on this kind of environment information in an interfaced application has the potential to greatly improve recognition accuracy in speaker-independent mixed-initiative interfaces.

References

Kazimierz Ajdukiewicz. 1935. Die syntaktische konnexitat. In S. McCall, editor, *Polish Logic 1920-1939*, pages 207–231. Oxford University Press. Translated from Studia Philosophica 1: 1–27.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL '89)*, pages 143–151.

Verónica Dahl and Michael C. McCord. 1983. Treating coordination in logic grammars. *American Journal of Computational Linguistics*, 9(2):69–91.

John Dowding, Robert Moore, François Andery, and Douglas Moran. 1994. Interleaving syntax and semantics in an efficient bottom-up parser. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94)*.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

E. Keenan and J. Stavi. 1986. A semantic characterization of natural language determiners. *Linguistics and Philosophy*, 9:253–326.

Richard Montague. 1973. The proper treatment of quantification in ordinary English. In J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Langauge*, pages 221–242. D. Riedel, Dordrecht. Reprinted in R. H. Thomason ed., *Formal Philosophy*, Yale University Press, 1994.

William Schuler. 2001. Computational properties of environment-based disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL '01)*, Toulouse, France.

Stuart M. Shieber, Yves Schabes, and Fernando C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

---

[10] Successful parses are those that result in one or more complete analyses of the input, even if the correct tree is not among them.