Incremental Interpretation and Lexicalized Grammar

Vincenzo Lombardo (vincenzo@di.unito.it)
Leonardo Lesmo (lesmo@di.unito.it)
Luca Ferraris
Crispino Seidenari
Dipartimento di Informatica - Universita' di Torino
Centro di Scienza Cognitiva - Universita' di Torino
c.so Svizzera 185, 10149 Torino, Italy

Abstract

The increasing lexicalization of syntactic theories poses new difficulties for incremental models of language processing. In this paper, we describe an incremental interpreter that makes use of knowledge on categories to keep the syntactic structure always connected. This, in turn, guarantees a fine-grained syntax-semantics interaction. The paper introduces the general problem of formalizing the notion of incremental interpretation, and analyzes the current approaches in the cognitive literature.

Introduction

Incremental interpretation has been widely assumed in the psycholinguistic models of human sentence processing since the work of (Marslen-Wilson 1973). Incremental interpretation constrains the language processor to analyze the input from left to right, and to produce a semantic representation for the partial syntactic structures associated with initial sentence fragments. The incremental strategy provides a major memory advantage to the processor, by keeping as low as possible the number of unstructured items in the working memory.

The vast body of experimental evidence in favor of incremental interpretation ranges from the high speed that humans exhibit in shadowing and interpreting speech (Marslen-Wilson 1973), to the on-line data on the processing of head-final languages like Dutch (Frazier 1987), Japanese (Yamashita 1994), German (Bader, Lasser 1994). A recent series of multimedial experiments, which integrate visual recognition from spoken instructions, has revealed a fine grained interaction of the syntactic, semantic and discourse knowledge (Tanenhaus, Spivey-Knowlton, Eberhard, Sedivy 1995). In a typical experiment of this series (fig. 1), the subject initially looks at the cross. Then s/he hears the utterance "Touch the starred yellow square" through a headphone. A head-mounted eye-tracker records the movements of her/his eyes during sentence comprehension. The results are that s/he directs her/his eyes toward the right object, as soon as s/he hears some distinctive attribute ("starred" in fig. 1), before hearing the head "square". This suggests that the commitment to the syntactic analysis that contributes to the semantic interpretation occurs very early (in particular, this result excludes head-licensing models (Abney 1989; Pritchett 1992)).

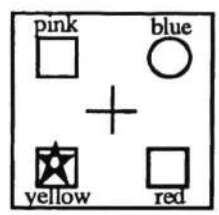


Figure 1. A typical experimental setting for object recognition on a screen.

The design of a computational model that implements some form of incremental interpretation is not immediate. Beyond the core intuition, the formalization of the notion of incrementality has to deal with a number of details which are related to both the linguistic theory and the language processor architecture. There are two major approaches to the formalization (and implementation) of incrementality. One is to use a syntactic formalism which allows structural aggregations that go beyond standard constituency, in order to assign a semantic interpretation to most of the sentence fragments; the other is to work on the synchronization between the parser and the interpreter, in order to assign a semantic interpretation to incomplete trees.

Most of the work in the first approach has been developed in the Categorial Grammar framework. The combination operators introduced in Combinatory Categorial Grammar (Steedman 1997) express the syntactic constraints and provide the basic tools of the processing architecture. In fact, on one hand, they permit the treatment of most linguistic phenomena (including unbounded dependencies and coordination) via an increase of the expressive power; on the other, they are able to assign a semantic type to a vast number of initial sentence fragments (those licensed by the grammar). The increase of the expressive power may cause an unwanted overassignment of syntactic types to word strings together with the problem of spurious ambiguity (many derivations for one interpretation). Milward (1995) solves some of these problems by using a simpler categorial framework (Applicative Categorial Grammar, with only functional application), which is expressed in a HPSG notation. However, the computational model does not make any abstraction (from words to syntactic categories) over the lexicalized character of the formalism. This can cause tractability problems in processing, because of the

impossibility to predict the words of the sentence. The abstraction over syntactic categories could avoid some of these problems, by using a sort of underspecified representation (with respect to a lexicalized grammar).

The second approach, which works on the processing architecture, usually assumes a traditional phrase structure syntax. The idea is the following: given that a well-formed (i.e. complete) tree T maps to a semantic type S_T (which can be a state, an event, a truth-value, ...), when T misses some subconstituent C which maps to the semantic type S_C, it can be interpreted as a function from S_C to S_T. A basic parsing algorithm provides the partial (i.e. incomplete) constituents to the interpreter at a rate which depends from the general architecture. Most approaches assume a word basis interaction and focus on the actual parsing strategy: Pulman (1986) adopts a left-corner parser, Stabler (1991) a top-down parser. Other authors define more complex interactions, which also affect the parser design. The authors of the system COMPERE (Eiselt, Mahesh, Holbrook 1993; Mahesh 1994) adopt a particular variant of the left-corner strategy, called Head-Signaled Left Corner (HSLC), which is claimed to be more effective than other interaction forms. HSLC defines a set of special daughters for each non terminal symbol, and requires that attachments are executed only when all the special daughters of a node have been parsed. A weakness of this approach is that the definition of the special daughters has an empirical origin, instead of descending from some theory. Also, according to the guidelines provided by the authors, adjuncts are unlikely to be special daughters; but, to explain the experimental data on the visual recognition mentioned above (Tanenhaus et al. 1995), it is essential to assume an eager attachment of modifiers (adjectives). Furthermore, a general limit of these approaches is the difficulty of dealing with left-embedding structures, because of the termination problems of the top-down parser and of the non incrementality of the input buffering of the left-corner parser, respectively. However, these models have been assumed in most research in psycholinguistics, and the existence of experimental data, not widely available for the categorial approaches, make them an interesting framework for further research.

In this paper we propose an architecture for incremental processing, that works with a lexicalized grammar. Lexicalization is a common trend of recent linguistic theories (GPSG, lexicalized CFG, lexicalized TAG, LFG, HPSG, Minimalist theory). Syntactic constraints tend to refer to individual words rather than to general syntactic categories. In parsing, the notion of rule application is no longer distinct from inserting the word in the structure. The traditional parsing schemata which support an incremental processing (top-down, left-corner, and variants) need to account for the processing peculiarities of lexicalized grammars. In particular, they need to acount for the non lexical structure building, which is necessary to keep the syntactic structure fully connected. The architecture proposed in this paper allows a fine grained syntax-semantics interaction by pairing the parsing primitives with semantic builders, while keeping the syntactic structure fully connected.

The paper is organized as follows. In the next section we describe a lexicalized dependency grammar based on subcategorization frames; then we illustrate the incremental interpreter and trace an example; finally, we provide some conclusions.

A lexicalized grammar

In order to illustrate the details of the processing mechanism, we introduce a lexicalized formalism: a word-based dependency grammar. Dependency syntax is not new to psycholinguistic modeling (see, e.g., (Milward 1994; Pickering 1994)), especially in the form of a variant of categorial grammar. In general, the categorial approach defines the combination operations to implement incrementality (e.g., functional application and composition), and the dependency theory restricts the syntactic relations which are relevant in language processing. In this work, we use a pure dependency grammar, expressed in a lexicalized form. The syntactic combination occurs through general parsing primitives coupled with operations that extend the semantic representation monotonically making use of the notion of underspecification.

Dependency syntax describes the structure of a sentence in terms of binary *head-dependent* (also called *dependency*) *relations* on the words of the sentence. The set of the dependency relations of a sentence forms the *dependency tree*. One special word, the *root* of the dependency tree, does not play the role of dependent in any relation. The dependency tree of the sentence "George touched the starred yellow square" is in fig. 2.

A dependency grammar consists of a set of syntactic categories (including a specification of the root categories), a set of words, a set of *lexical signs* (see below). Each word is associated with a number of lexical signs (because of lexicalambiguity). A lexical sign defines the syntactic and the semantic features of the word, in terms of the representational entities involved and of how these entities combine to yield the syntactic and the semantic representations of the whole sentence, respectively. A lexical sign is a feature structure which consists of three main parts: the syntactic features (SYN), expressed in the form of *dependency rules*; the semantic features, including a *semantic type* and the corresponding *selectional restrictions*; the syntax-semantics interface, the so-called *linking rules*.

A dependency rule describes the subcategorization constraints of an individual word. Given the lexical sign of

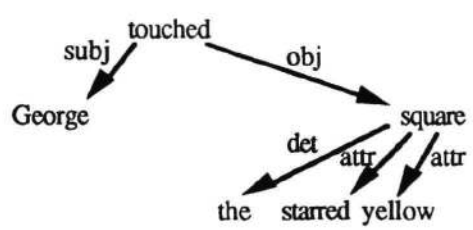


Figure 2. The dependency tree of the sentence "George touched the starred yellow square". The orientation of the arcs represents the linear order of the words in the sentence.

the word x, a generic dependency rule is of the form

$X(<r_1Y_1><r_2Y_2>...<r_{i-1}Y_{i-1}>\#<r_{i+1}Y_{i+1}>...<r_mY_m>)$

where X and Y_j ($1 \le j \le m$) are syntactic categories, r_j ($1 \le j \le m$) is a dependency relation (SUBJ, OBJ, ATTR, ...), # is a special symbol which marks the position of the head in the linear order of dependents. A dependency rule $X(r_1Y_1\ r_2Y_2\ ...\ r_{i-1}Y_{i-1}\ \#\ r_{i+1}Y_{i+1}\ ...\ r_mY_m)$ constrains the form of a configuration head-direct dependents in a dependency tree, when the head is a word of category X. The dependent words y_j refer to lexical signs with dependency rules of the form $Y_j(...)$. The following are dependency rules:

V (<SUBJ,N> # <OBJ,N>) (1)

N (<DET,D> <ATTR,A>* #) (2)

(1) is a dependency rule for transitive verbs, which require a subject (that precedes the verb in the linear order) and an object (which follows the verb); (2) is the rule for common nouns (preceded by a determiner and an indefinite number of adjectives). N, V, A and D are syntactic categories. Some categories can encode specific feature values. For example, N[+Proper] and V[+PAS] are the categories for proper names and passive verbs, respectively. These categories have different dependency rules.

The semantic part of a lexical sign (SEM) contains the semantic type of a word, together with the selectional restrictions on the fillers of the thematic roles required by the semantic type. Semantic types are arranged in taxonomies of concepts, and are represented in a terminological knowledge base. Each base syntactic category C, which stands at the top of some categorial taxonomy, maps onto a top semantic type S_C; individual words of category C maps onto a specialization S' of the corresponding semantic type S_C. Verbs are mapped onto a taxonomy whose root is the concept of "state or event", while nouns refer to a taxonomy rooted in "individual or state or event"; adjectives implicitly refer both to properties and to property values (although the property can be specified explicitly; cf. "red ball" vs. "ball of red color"). For the sake of readability, we associated concepts of different kinds with different prefixes: @ for states and events, § for individuals and properties, # for property values. So, we have @state-or-event, @touch, §physical-object, §person, #red, etc. Note, however, that the taxonomies are interconnected, so that the syntactic difference need not imply semantic difference (consider deverbalized nouns as *destruction* from *destroy*, both of which refer to an event).[1]

The value of the feature SEM consists of three attributes: QUANT, INDIV and MATRIX. We assume that each node in the dependency tree refers to a given individual (a variable which is the value of INDIV), quantified according to QUANT, and with properties specified by the formula MATRIX, where the variable in INDIV occurs free.

The mapping of the dependency (grammatical) relations between the verb and its dependents (e.g. SUBJ, OBJ) onto the semantic labels expressing the (thematic) roles of the participants (e.g. AGT, PATIENT), i.e. the "linking" problem (see, for instance, (Levin 1993)), is solved here by explicitly specifying the mapping in the verbal entry under the LINK attribute.[2]

The pieces of information associated with lexical items are put together in an incremental way (see the next section) in order to build up the final representations of the sentence. The syntactic representation is a dependency tree, where nodes are instantiations of lexical signs, that include a word and a dotted dependency rule (see below) as SYN, and copies of QUANT, INDIV and MATRIX a SEM. The semantic representation is a first-order formula, where events and situation are reified. The predicates express the connections between an argument individual (which can be an event, a state, an entity, etc.) and an argument concept (e.g. §person, @touch), or among individuals. For instance, *is-a(e, @touch)* specifies that *e* is an individual event of type *@touch*, and *agt(x,e)* says that the individual *x* is the agent of the action *e*. In a dependency framework, the value of SEM of a node n includes only the properties derived from the single word associated with n, while the complete set of properties of the individual can be obtained from the nodes linked to n. The interpretation algorithm collects all the pieces of information associated with the various nodes (as they are built) in order to build the complete formula.

Incremental interpretation

To yield an incremental interpretation, we have to be able to integrate the new pieces of semantic representation as soon as possible: the more connected is the syntactic structure during the parsing process, the more connected is the partial semantic representation.

Top-down parsing, as opposed to bottom-up and left-corner, keeps the syntactic structure always connected. The problems manifested by top-down parsers with the left-embedding structures (an example of infinite local ambiguity) can be faced with the Minimal Recursive

---

[1] The taxonomies do not include only specialization (class-subclass) links, but they are the repository for much other knowledge required for interpreting linguistic expressions and for making predictions on their structure and meaning. For instance, the constraints on the relationships between properties and individuals are represented as explicit links restricting the range of applicability of the property to a given concept (e.g §color is linked to §physical-object). Analogously, the possible participants in an event (as the "toucher" and the "touchee" in a touching event) are specified in the taxonomy as "roles" of the event together with the associated selectional restrictions (the "toucher" and the "touchee" must be physical objects). Different senses can be associated with the same verb (cf. "the discourse touched different aspects of the problem"). Actually, since the taxonomy enforces an inheritance mechanism, some role labels are inherited from more general concepts; so, in the examples below, the role labels "toucher" and "touchee" are replaced by the (more general) labels AGT and PATIENT (see (Goy, Lesmo 1997) for a more detailed description of the structure of the taxonomy, and (Di Tomaso, Lombardo, Lesmo 1998) for more information on the analysis of locative expressions).

[2] In the complete model, we have defined a taxonomy of linking classes separate from, but closely similar to, the taxonomy of states and events.

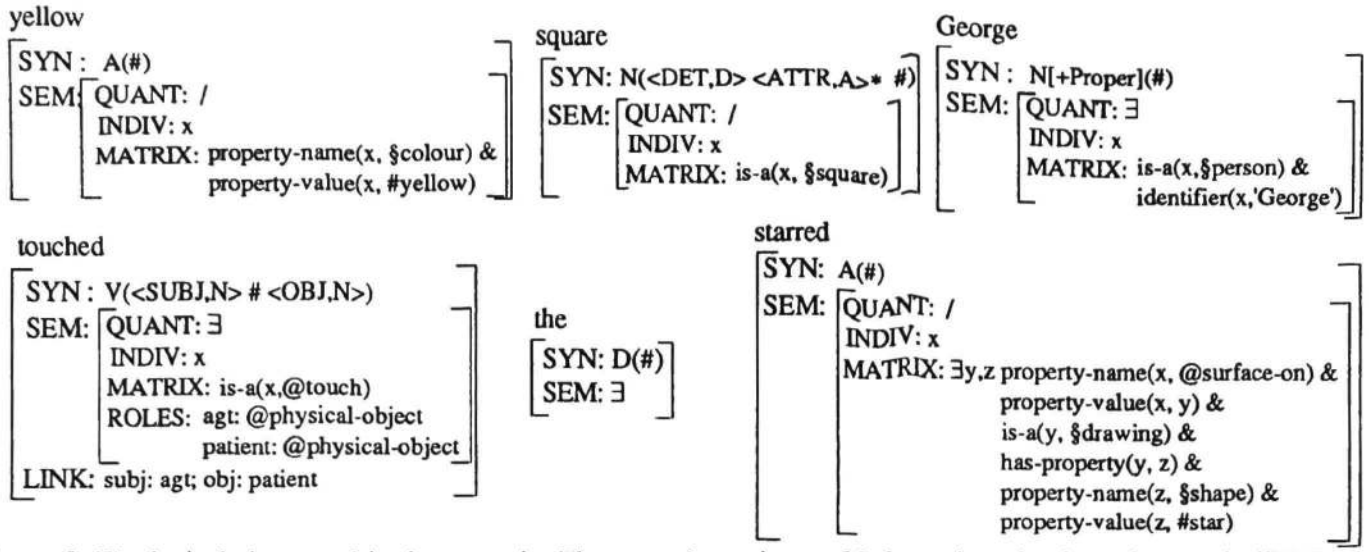


Figure 3. The lexical signs used in the examples.There are three pieces of information: the dependency rule (SYN), the meaning (SEM) and, possibly, some "linking" between the dependency relation and the corresponding thematic role (LINK).

Structure mechanism described in (Lombardo, Sturt 1997). This augmented top-down parser exhibit a cognitively plausible behavior, because it correctly assigns a processing difficulty to center-embedding structures with respect to left- and right-embedding structures.

The (augmented) top-down strategy builds the syntactic structure from the root: at each step, it guesses a portion of structure (the *connection path*) and inserts the input word. In dependency grammar, node guessing corresponds to the prediction of a word of category C in the input: the parser introduces a template node labelled C_x in the structure via the primitive *crlink* (x is a numeric index), and then C_x will be filled with an input word (primitive *fill*).

To apply the augmented top down parser to lexicalized grammar, we need to augment the grammar with some knowledge on the left corner of categories: given a syntactic category C, we define as LC(C) the set of categories that can appear as the left corner of a dependency tree rooted by a word of category C. This permits to build the connection path for each word, and to keep the syntactic structure connected. For our example grammar (fig. 3), we only have

LC(V)={N, D}, LC(N)={N, D}

The other categories (A, D, N[+Proper]) only have themselves as left corner. The left-corner knowledge is easily extracted from the dependency rules.

The use of connection path knowledge, which in our case corresponds to left-corner, is not new in psycholinguistic modeling, where it refers in general to the problem of non-lexical structure building in lexicalized grammar parsing (see, e.g., the discussion in (Sturt 1997)). This knowledge also avoids the explosion of useless predictions in traditional (non lexicalized) top-down parsers.

Parsing primitives

The parsing primitives *crlink* and *fill* consist of two parts: the syntactic part, that builds the dependency tree, and the semantic part, that accumulates the semantic representation. The synatctic part of *Crlink(n, r, Cat)* creates a node of category Cat and links it to (i.e. makes it depend on) the node n through the relation r; *fill(n,w)* associates the input word w (of category Cat) with the node n (again of category Cat). A node which does not contain an input word is called *empty*; a node which does contain an input word is called *full*. The *crlink* action creates empty nodes; the *fill* action transforms an empty node into a full node. A full node that governs all the modifiers licensed by a dependency rule is called *complete*. Note that the *crlink* action top-down creates and links a new node: since each new node is immediately linked to an existing node, the syntactic structure is always connected.

A semantic action is associated with *crlink* and *fill*. In case of *crlink*, a new individual is introduced. Its type is still very general, but it is not completely unknown, since it depends on the category of the created node. For instance, in case of a new verbal node, its SEM.INDIV is a fresh variable e_{new}, its SEM.MATRIX gets the value *is-a* (e_{new}, *@state-or-event)*, and its SEM.QUANT is a special symbol standing for a still unknown quantifier[3]. Moreover, the semantic connection between the existing representation and the newly entered individual is determined on the basis of the dependency relation and of the meanings of the two linked nodes (via the function *link*). In case of *fill*, the semantic effect is to extend the value to the feature SEM.MATRIX of the filled node, according to the lexical semantics of the word. This is obtained by and-ing the current value with the new information and substituting the INDIV variable for the main variable of the lexical SEM. So, if the original value of SEM.MATRIX is *is-a(e,@state-or-event)*, the new value will be *is-a(e,@state-or-event) & is-a(e,@touch)* . Of course, since *@touch* is more specific than *@state-or-event*, the new MATRIX is equivalent to *is-a(e,@touch)* .

[3] In the example of the next section, we assume existential quantification, since the treatment of determiners and of the subfeature QUANT is rather complex and outside the scope of this paper.

The algorithm

The parsing algorithm presents two phases: prediction (*crlink*) and scanning (*fill*). Here follows the algorithm. The algorithm is in non deterministic form: we do not include any preference for local ambiguity, but we assume that it always makes the correct move. In other works, we show how this approach to incremental interpretation can face ambiguity resolution and reanalysis (Lombardo 1995; 1998; Lombardo, Sturt 1997).

Input: A sentence x = w_0 w_1 ... w_{n-1}
Output: A dependency tree and a logical form
begin
n := newnode (V); {where V is the root category}
n.SEM:= ∃e. is-a(e, @state-or-event)
for each input word w do
  select an interpretation for w (syntactic category Cat_w)
  select a continuation node n on the tree edge
  if Cat_w=n.cat & the next symbol in n.rule is #
  then fill(n, w)
  elseif the next symbol of n.rule is <d-rel, Cat>
      & Cat_w belongs to LC(Cat) then
    crlink(d-rel, Cat, n)
  else fail
    if all the nodes in the dependency tree are complete
    then accept else reject
end

The algorithm begins by initializing the dependency tree with the root, a node of category Cat_r created by the primitive *newnode*, and the logical form, providing a quant(∃), an indiv (e), and a matrix (is-a(e, @state-or-event)). The main loop cycles on the input words. For each word, the algorithm selects an interpretation for the input word (including a category Cat_w) and a node for continuation in the dependency tree. These selections depend on the strategies used for (lexical and attachment) ambiguity resolution. Then, if the category of the input word is equal to the category of the node and the node is empty, the word is stored in this node, and the logical form is appropriately updated (*fill*). Otherwise, if the category of the word is one of the left corner categories of the next dependent of the continuation node, it initializes the subtree of that dependent by creating the root node (*crlink*). If none of these continuations are possible, the process fails. At the end of the sentence, all the nodes in the tree must be complete, in the sense they satisfy the constraints of subcategorization (dependency rule) provided by the respective words.

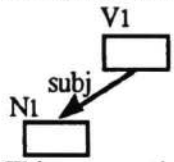
An example

In this section we describe the application of the incremental algorithm to the example sentence "George touched the starred yellow square", with a structure similar to the utterances used in the visual recognition experiment described in the introduction. We also assume the (limited) referential context provided in that experiment.

At the beginning, the dependency tree is an empty node of category V (the root). This creation triggers the semantic processor, which identifies V_1 as a node of category V, and produces the interpretation

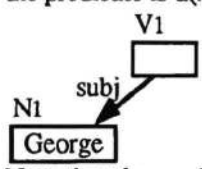
∃e.is-a (e, @state-or-event)

The first word (George) produces a sequence of *crlink* operations, the first of which creates a subject (left) dependent of V_1 of category N. The interpreter expands the semantic representation as follows (in the figures, we used *@st-or-ev* and *@ind-or-st-or-ev* in place of *@state-or-event* and *@individual-or-space-or-event* because of space reasons):



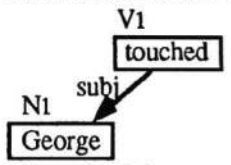
∃e, x. is-a (e, @st-or-ev) &
is-a (x, §ind-or-st-or-ev) &
link (@st-or-ev, §ind-or-st-or-ev, subj) (x, e)

This means that there is some entity playing some role (corresponding to the relation subj) in the state or event described. This entity has type *§ind-or-st-or-ev* i.e. it is the most general nominal concept. Its exact role has not yet been identified, because of the lack of information about the event. Then, N_1 is filled with *George*, and this specializes the predicate is-a(x, §ind-or-st-or-ev).



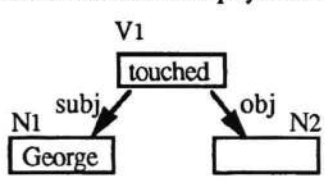
∃e, x. is-a (e, @st-or-ev) &
is-a (x, §person) &
link (@st-or-ev, §person, subj) (x, e) &
identifier (x, 'George')

Note that the predicate is-a (x, §ind-or-st-or-ev) is still part of the representation, but we have excluded it from the figure, because it is specialized by *is-a(x,§person)*. Then, the interpreter fills the node V_1 with *touched*. This produces again a specialization (of is-a(e,@st-or-ev)), as well as the identification of the George's role (*agt*):



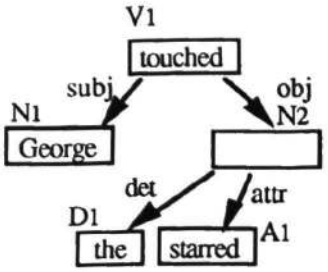
∃e, x. is-a (e, @touch) &
is-a (x, §person) &
agt (x, e) &
identifier (x, 'George')

When *the* (of category D) is analyzed, the check on the left corner of the category N is positive, and the parser builds the node N_2, and attaches it as right (*object*) dependent to V_1. The linking rules succeed because the verb is already present: they tell that the direct object is the *patient* of the touching event. On the basis of this result, it is possible to apply the selectional restriction to the *patient* argument of *touch*: It must be a physical object. So we get:



∃e, x, y. is-a (e, @touch) &
is-a (x, §person) &
agt (x, e) &
identifier (x, 'George') &
is-a (y, §physical-object) &
patient(y, e)

The subsequent creation and filling of a D_1 node with the word *the* contributes to the representation just with a specification of the quantifier associated with the variable y (that we gave for granted). When *starred* is found, we obtain, in two steps, the following representation:



∃e, x, y, z, w, t.
is-a (e, @touch) &
is-a (x, §person) &
agt (x, e) &
identifier (x, 'George') &
is-a (y, §physical-object) &
patient (y, e) &
has-property (y, z) &
property-name(z,@surface-on) &
property-value (z, w)
is-a (w, §drawing) &
has-property (w, t) &
property-name (t, §shape) &
property-val (t, #star)

This representation can be paraphrased as: there has been a touching event, whose agent has 'George' as identifier and whose patient is some physical object having on its surface a drawing whose shape is a star. This formula can be used in the context to find out any referent satisfying the description; in particular, any starred physical object can be found even if the associated head noun has not yet been analyzed. We must observe that the restriction of the *patient* to a *§physical-object* is conceptually important, but does not affect the global behavior: even a "starred" *§ind-or-st-or-ev* could be found successfully in a limited context. Note also that, in principle, linking rules could apply also to more general concepts (e.g. *@st-or-ev*), were they defined for them; since, in general, this is not the case, they must wait for a suitable specialization before returning the associated role label.

Conclusion

This paper has presented a computational model based on a lexicalized grammar that implements incrementality. Left-corner constraints on the syntactic category are applied in the prediction phase, when template nodes are instantiated and attached to the existing structure; word lexical information is applied in the scanning phase, when the word is processed and fills some template node previously created. An advantage of the dependency formalism is that the top-down strategy implies a minimal commitment on the syntactic structure, because it does not pose any superstructure over the word level. The abstraction on (left corner) knowledge on the categories guarantees the full connectivity of the syntactic structure, that is necessary for incrementality.

We do not take a position on the constraint-based/syntax-autonomy debate in ambiguity resolution. The model is compatible with both of them, and is intended to provide a basic mechanism for incremental processing.

References

Abney S. P., *A computational Model of Human Parsing*. **J. of Psycholinguistic Research 18/1**, 1989, pp. 129-144.

Bader M., Lasser I., *German Verb-Final Clauses and sentence Processing: Evidence for the Immediate Attachment* in Clifton C., Frazier L., and Rayner K., (eds) , **Perspectives on Sentence Processing**, Lawrence Erlbaum Associates, 1994, pp. 225-242.

Di Tomaso V., Lombardo V., Lesmo L., *A computational model for the interpretation of static locative expressions*, to appear in Olivier, Gapp (eds.), LEA, 1998.

Eiselt K.P., Mahesh K., Holbrook J., *Having your cake and eating it too: autonomy and interaction in a model of sentence processing*, Proc. of AAAI93, 1993, pp. 380-385.

Frazier L., *Syntactic Processing: Evidence from Dutch*,. **Natural Language and Linguistic Theory 5**, 1987, 519-559.

Goy A., Lesmo L., *Integrating Lexical Semantics and Pragmatics: The Case of Italian Communication Verbs*, Proc. Second Int.l Workshop on Computational Semantics, Tilburg, The Netherlands, 1997, pp.81-93.

Levin B., **English Verb Classes and Alternations**, The University of Chicago Press, Chicago, 1993.

Lombardo V., *Parsing and Recovery*, Proc. of the 17th Annual Meeting of the Cognitive Science Society, Pittsburgh, 1995.

Lombardo V., *A Computational Model of Recovery*, to appear in Fodor J.D., Ferreira F. (eds.), **Reanalysis in Sentence Processing**, Kluwer Academic Publishers, 1998.

Lombardo V., Sturt P., *Incremental Interpretation and Infinite Local Ambiguity*, Proc. of the 19th Annual Meeting of the Cognitive Science Society, Stanford, August 1997.

Mahesh K., *Reaping the benefits of interactive syntax and semantics*, Proc. of ACL94, 1994, pp. 310-312.

Marslen-Wilson W., *Linguistic Structure and Speech Shadowing at Very Short Latencies*, **Nature 244**, 1973, pp;. 522-523.

Milward D. R., *Dynamic dependency grammar*. **Linguistics and Philosophy**, December 1994.

Milward D. R., *Incremental interpretation of categorial grammar*. Proceedings of the 7th European ACL, Dublin, Ireland, 1995, pp. 119-126.

Pickering M.J., *Processing local and unbounded dependencies: a unified account*, **Journal of Psycholinguistic Research 23/4**, 1994, pp. 323-352.

Pritchett B. L., *Grammatical Competence and Parsing Performance*, University of Chicago Press 1992.

Pulman S.G., *Grammars, parsers, and memory limitations*, **Language and Cognitive Processes 1/3**, 1986,. 197-225.

Stabler E.P., *Avoid the pedestrian's paradox*, in Berwick R.C. et al. (eds.), **Principle-based parsing: Computation and Psycholinguistics**, Kluwer, The Netherlands, 1991, pp. 199-237.

Steedman M. J., **Surface structure and interpretation**, MIT Press, Cambridge, MA, 1997.

Sturt P., *Syntactic Reanalysis in Human Language Processing*, Ph.D. thesis, Centre for Cognitive Science, University of Edinburgh, Edinburgh, Scotland, 1997.

Sturt P., Crocker M. W., *Monotonic syntactic processing a cross-linguistic study of attachment and reanalysis*, **Language and Cognitive Processes 11/5**, 1996, pp. 449-494.

Tanenhaus M.K., Spivey-Knowlton M.J., Eberhard K.M., Sedivy J.C., *Integration of Visual and Linguistic Information in Spoken Language Comprehension*, **Science 268**, 1995, pp. 1632-1634.

Yamashita K., *Processing of Japanese and Korean*, Ph.D. thesis, Ohio State University, Columbus, Ohio, 1994.