# A Graph Propagation Architecture for Massively-Parallel Constraint Processing of Natural Language

**Hideto Tomabechi**
Carnegie Mellon University
Smith Hall 109, Pittsburgh PA 15213
tomabech@cs.cmu.edu

## Abstract

We describe a model of natural language understanding based on the notion of propagating constraints in a semantic memory. This model contains a massively-parallel memory-network in which constraint graphs that represent syntactic and other constraints that are associated with the nodes that triggered activations are propagated. The propagated constraint graphs of complement nodes that collide with the constraint graphs postulated by the head nodes are unified to perform constraint applications. This mechanism handles linguistic phenomena such as case, agreement, binding and control in a principled manner in effect equivalent to the manner that they are handled in modern linguistic theories.

## Motivation

The so called 'direct memory access' (or memory-based) paradigm of natural language recognition seems to have established itself as an important approach to natural language. Originated by Quillian ([Quillian, 1969]), this model of natural language processing views natural language understanding as the activity of identifying the input utterance with existing memory structures. While these memory structures have taken various forms depending upon underlying theoretical approaches (such as MOPs of [Riesbeck and Martin, 1985] and Cases in [Martin, 1989]), the underlying control structure has remained shared, namely, spreading activation and marker passing. The advantage of this control structure is that since spreading activation takes place directly in the memory network, direct memory-based inferential processing can be employed at any point of recognition. A large number of systems have taken advantage of this control structure in order to perform recognition and inference directly on a semantic memory. One well known weakness of this approach is its inability to handle complex syntactic constraints. Since the memory network represents the hierarchical organization of conceptual relations and packaging of thematic and experiential memory units, adequate representation of syntactic constraints has been very difficult, if not impossible. It is because syntactic constraints are often configurational and are assigned based upon particular structural patterns of the constituent buildup at the time of utterance. For example, a syntactic *case* of a particular noun phrase is dynamically determined at the time of the utterance based upon the particular syntactic configuration and is essentially impossible to capture *a priori* in the semantic network (otherwise we will have to specify infinitely many sentential patterns as well as many redundant subcategories of concepts based upon *case* variety simply to capture the *case* agreement phenomenon). Similarly a recognition of *Sue said that Mary ran* under the memory-based paradigm would require instantiation of two sequences of concepts in memory [*PERSON *MTRANS-word that *ACTION] and [*PERSON *RUN] (i.e., a typical sentential recognition methodology currently employed under this paradigm using patterns of concepts). The first sequence is probably attached to a root node such as *MTRANS-EVENT and the second sequence is attached to *RUN-ACTION which is a subclass of *ACTION. The last element of the first sequence *ACTION will be activated by the recognition of any subclass of *ACTION including *RUN-ACTION, *SLEEP-ACTION, etc.. However, there is no way to ensure that if the *MTRANS-word is *said*, the set of entities that caused the activation of *ACTION contain (be headed by) a finite verb (unless we create a possibly infinite number of concepts for different combinations of syntactic features such as *FINITE-FORM-RUNNING-ACTION-TAKING-NOMINATIVE-SUBJECT). Thus, it is essentially impossible to capture the grammaticality of *John said Mary runs* and the ungrammaticality of *John said Mary run* (or *John said Mary to run*) within the constraints capturable inside the semantic memory network. Another well known syntactic phenomenon that is essentially uncapturable within the semantic network methodology is the so called *obligatory control* phenomenon in which a subject (or object) of the external clause controls the unexpressed subject of the internal clause. For example, in *John persuaded Mary to* *give Sandy sushi*, the subject of *give* is unexpressed but is 'controlled' to be the object of the external clause, i.e., *Mary*. This is called *object control* and in order to handle this, a configurational formulation of the syntactic constraints will be needed, which is non-trivial in a straightforward semantic network.[1]

Other problematic constraints include *long-distance dependency* in which the dependency relation between the two noun phrase positions is postulated for an unbounded length and depth of the syntactic tree configuration. Some binding of anaphors, such as the binding of reflexive pronouns in English, also requires configurational constraint processing. Thus, many syntactic phenomena require constraint processing based upon syntactic constituents and configurations that are dynamically created during the recognition and *a priori* formulation of such constraints within the semantic memory network is very difficult, if not impossible.

## Constraint Propagation in Memory

The central notion of the proposed model is the **propagation of constraints** in the semantic memory network. This is in contrast to past models of semantic memory-based natural language recognition in which activations and pointers to sources of activations (i.e., *markers*) are propagated in the memory network. Instead of propagating markings on memory nodes, we explicitly propagate constraints. As a representational scheme for capturing constraints to be propagated, we find directed graphs to be appropriate. It is because their expressivity has been well supported by linguists ([Bresnan and Kaplan, 1982], [Pollard and Sag, 1987], etc.) and the operations on them (i.e., graph unification) have been extensively researched[2]. In fact the problematic linguistic phenomena for the memory-based paradigm such as *agreement*, *obligatory control*, and *long-distance dependency* have all been captured in the unification-based grammar theories in which graph representation and graph unification are assumed to be the tool for capturing the constraints required for handling these phenomena. Thus, we propose a model of natural language recognition based upon propagation of constraints in a semantic memory network in which constraints are captured using directed graphs. We will call the proposed semantic network with capacity to propagate graphs *Graph-based Constraint Propagation Network or GCPN* in this paper. The graphs[3] propagated in the GCPN may point to any location in the network and may contain complex paths including convergent arcs and cycles. The existential nature of the graphs is different from that of the constraint propagation network itself[4] in the sense that the semantic memory network constitutes (following the tradition since [Quillian, 1966]) the hierarchical organization of concepts that represents a semantic memory and the constraint graphs represent mostly syntactic constraints such as head features and configurational constraints which do not belong to the semantic memory.

The expressivity of the graph-based constraint propagation scheme is significantly greater than that of traditional memory-based schemes and their extensions. As we mentioned above, syntactic expressivity of this scheme is essentially equivalent to the expressivity of unification-based theories (such as HPSG, [Pollard and Sag, 1987] and LFG [Bresnan and Kaplan, 1982]) while the semantic and pragmatic expressivity of the memory-based paradigm is maintained.

Below is a sample lexicon from our experimental system called MONA-LISA ([Tomabechi, ms]) which uses the GCPN as the central representational network. We have adopted HPSG ([Pollard and Sag, 1987]) as the basis of providing the linguistic constraints to the system as graphs. The first sample entry is the specification for the concept *JOHN with the constraints specified here using path equations similar to the PATR-II notation ([Shieber, *et al*, 1983]).

```
(def-frame *JOHN
  (inherits-from *MALE-PERSON)
  (type :lex-comp)
  (spelling   John)
  (synsem
   (def-path
    (<0 loc cat head maj> == n)
    (<0 loc cat marking> == unmarked)
    (<0 loc cont para index> == [[per 3rd]
                                 [num sng]
                                 [gend masc]])
    (<0 loc cont restr reln> == *JOHN)
    (<0 loc context backgr> == [[reln naming]
                                [name JOHN]])
    (<0 loc context backgr bearer>
                         == <0 loc cont para index>)
    (<0 mem> == <0 loc cont para index iden>))))
```

When this lexical definition is read into the system, the path equations are converted into graphs as shown below.

```
(*JOHN
  (INHERITS-FROM *MALE-PERSON)
  (TYPE
```

[1]In fact, the DMA systems that handle these phenomena rely on either prestored functional constraint applications specifically prepared for particular sentential constructions (such as [Tomabechi and Levin, 1989]) or on prestored (hard-wired) relations between conceptual entities for particular sentential constructions (such as [Kitano, 1990]). In either case, formulation of prestored relational constraints are at best *ad hoc* and no generality is guaranteed.

[2]Such as [Pereira, 1985], [Karttunen, 1986], [Kasper, 1987], [Wroblewski, 1987], and [Tomabechi, 1991].

[3]Implementationally, they are pointers to graphs instead of graphs themselves.

[4]Of course, the GCPN itself is a graph; however, here, 'graphs' means the constraint graphs and not the memory-network.

```
    (VALUE
      (COMMON :LEX-COMP)))
  (SPELLING
    (VALUE
      (COMMON JOHN)))
  (SYNSEM
    (VALUE
      (COMMON
X01[[0 X02[[MEM X03[]]
          [LOC X04[[CONTEXT
                    X05[[BACKGR
                         X06[[BEARER
                              X07[[IDEN  X03]
                                  [GEND  X08 MASC]
                                  [NUM   X09 SNG]
                                  [PER   X10 3RD]]
                                  [NAME  X11 JOHN]
                                  [RELN  X12 NAMING]]]
                    [CONT X13[[RESTR
                               X14[[RELN X15 *JOHN]]
                              [PARA  X16[[INDEX  X07]]]
                    [CAT  X17[[MARKING  X18 UNMARKED]
                              [HEAD
                               X19[[MAJ X20 N]]]]]]])))))
```

In the current formulation of GCPN, the constraint graphs are stored in synsem values of the nodes and the top level number 0 arc represents constraints to the node itself. If a node has its complement nodes the constraints are specified by numbers higher than 0 (in the order of obliqueness). For example, the lexical specification for the node *GIVE looks as follows:

```
(def-frame *GIVE
 (inherits-from *GIVE-ACTION)
 (type :lex-head)
 (spelling give)
 (synsem
  (def-path
   (<0 loc cat head> == [[maj v]
                         [vform bse]
                         [aux -]
                         [inv -]
                         [prd -]])
   (<0 loc cat marking> == unmarked)
   (<0 loc cat subcat 1> == <1>)
   (<0 loc cat subcat 2> == <2>)
   (<0 loc cat subcat 3> == <3>)
   (<0 loc cont reln> == *give-action)
   (<1 loc cat head maj> == n)
   (<1 loc cat head case> == nom)
   (<0 loc cont agent> == <1 loc cont para index>)
   (<1 loc cont restr reln> == *person)
   (<2 loc cat head maj> == n)
   (<2 loc cat head case> == acc)
   (<0 loc cont goal> == <2 loc cont para index>)
   (<2 loc cont restr reln> == *person)
   (<3 loc cat head maj> == n)
   (<3 loc cat head case> == acc)
   (<0 loc cont theme> == <3 loc cont para index>)
   (<3 loc cont restr reln> == *matter))))
```

The equations are converted into directed graphs when read into the system. This way, instead of simply storing simple case-frame type lexical specifications in the lexical nodes, we provide full graph-based lexical constraints in the lexical level nodes in the constraint propagation network. Let us provide a sample lexical node definition for the object control verb ***persuaded***:

```
(def-frame *PERSUADED
 (inherits-from *PERSUADE-ACTION)
 (type :lex-head)
 (spelling persuaded)
 (synsem
  (def-path
   (<0 loc cat head> == [[maj v]
                         [vform inf]
                         [aux +]
                         [inv -]
                         [prd -]])
   (<0 loc cat marking> == unmarked)
   (<0 loc cat subcat 1> == <1>)
   (<0 loc cat subcat 2> == <2>)
   (<0 loc cat subcat 3> == <3>)
   (<1 loc cat head maj> == n)
   (<1 loc cat head case> == nom)
   (<1 loc cont restr reln> == *person)
   (<0 loc cont agent> == <1 loc cont para index>)
   (<0 loc cont persuadee> == <2 loc cont para index>)
   (<0 loc cont persuadee>
                == <0 loc cont circumstance agent>)
   (<2 loc cat head maj> == n)
   (<2 loc cat head case> == acc)
   (<2 loc cont restr reln> == *person)
   (<3 loc cat head maj> == v)
   (<3 loc cat head vform> == inf)
   (<3 loc cat head aux> == +)
   (<3 loc cat subcat 1 loc cat head> == [[maj n]
                                          [case nom]])
   (<3 loc cat subcat 2 loc cat head> == saturated)
   (<3 loc cat subcat 3 loc cat head> == saturated)
   (<0 loc cont circumstance> == <3 loc cont>)
   (<0 loc cont reln> == *PERSUADE-ACTION)
   (<3 loc cont restr reln> == <3 loc cont reln>)
   (<3 loc cont restr reln> == *action))))
```

Thus the two equations:

$(\langle$ 0 loc cont persuadee $\rangle == \langle$ 2 loc cont para index $\rangle)$

$(\langle$ 0 loc cont persuadee $\rangle == \langle$ 0 loc cont circumstance agent $\rangle)$

can easily specify the control constraints lexically in the network.

## The Graph Propagation Recognition

Currently, our graph-based constraint propagation inheritance network has 5 types of nodes: conceptual-class nodes, lexical-head nodes, lexical-complement nodes, memory-instance nodes, and phonological-activity nodes. The conceptual-class nodes are nodes in the high levels of abstraction and are used for discourse and episodic recognition. Lexical-head nodes are nodes that are phonologically invoked with lexical activations and they package the complement nodes.

Lexical-complement nodes are the nodes that are lexically invoked and do not have their own complements. Memory-instance nodes are actual instances of lexical-heads and lexical-complements that are specific to the current utterance. Phonological-activity nodes are the nodes that represent phonemic units and are connected to the phonemic recognition modules. The activity of phonological-activity nodes is not discussed in this paper and the input is assumed to be already hypothesized as words (Please refer to [Tomabechi, ms] for activities of those nodes).

Below is the central part of our sentential recognition algorithm for word level input.

```
function sentential-recognize (input-stream)
  create-process (recognize-lexical (input-stream));
  invoke-global-incidents;
  for all NODE in DecayingLayer do
     print-node(NODE);

function recognize-lexical (input-stream)
 reset activities in Activation Layer
                 and Decaying Layer
 for word-hypothesis in input-stream do
    create-process (activate-lex-node
                            (word-hypothesis));
 invoke-global-incidents;

function activate-lex-node (word-hypothesis)
 create instance of word-hypothesis
 and make a copy of constraint graph
 with addition of an 'mem' arc pointing
 to the created instance;
 if the node type is lexical-complement
    then propagate copied (and modified)
         constraint graph upward;

function invoke-global-incidents ()
 for head-instance in ActivationLayer do
    create-process (grab-subcats (head-instance)) ;

function grab-subcats (head-instance)
 for arcs specified in subcat graph
                 (i.e, <0 loc cat subcat>) do
     if conceptual restriction node exists
                 (i.e, <loc cont rest reln> has value)
     and if that node has received
             the constraint graph propagation
          then unify the subcat graph
               with the propagated graph;
               if unify succeeds
                  and obliqueness order is met
                  then store result destructively
                       in head-instance;
                       propagate synsem graph upward;
```

Originally, the GCPN is configured hierarchically in terms of conceptual inheritance. Graph propagation occurs only upward in the inheritance hierarchy and never horizontally. Conceptual relations (other than inheritance) between lexical nodes are specified through constraint graphs (as seen in the sample entry in the previous section). All nodes originally belong to StaticLayer and when they are lexically invoked they move into an activated state (the nodes move to ActivationLayer). Only the saturated lexical-head nodes move to the DecayingLayer. So at the end of the recognition, the constraint 0 graph (i.e. the constraint to itself) of the nodes in the DecayingLayer contains information that can be used for further processing (such as generation). Actually, the printout of the constraint 0 graph of the Decaying node should look exactly like that of the output of a unification-based parser[5]. Let us briefly look at the constraint propagation activity with the recognition of *John persuaded Mary to give Sandy sushi.* First when the word *John* activates the lexical-complement node *JOHN, an instance *JOHN001 is created.[6] When the instance is created, the synsem graph of *JOHN is copied while putting the created instance as the destination of the ⟨ 0 mem ⟩ path of the newly created synsem graph. This synsem graph is propagated upward in the abstraction hierarchy (through *MALE-PERSON, *PERSON, ...). When the activation reaches the top of the abstraction hierarchy, then global-incidents is triggered. All currently activated head-instances concurrently check their subcat graph paths and if their destination nodes have currently received a graph propagation, then the head-instance's constraint graph and the propagated graphs are unified. Since at this moment no head-instance is activated nothing happens. When the next word *persuaded* is hypothesized, a head-instance *PERSUADE001 is created and a constraint-graph containing *PERSUADE001 in the ⟨ 0 mem ⟩ graph path is propagated upward in the abstraction hierarchy. When the activation reaches the top, global-incidents is invoked and PERSUADE001 which is a head-instance tries to grab its subcat fillers by checking the content of ⟨ loc cont rest reln ⟩ path for each of the subcategorization elements. Since *PERSON has already received a graph propagation at the recognition of *John* the propagated constraint graph of *JOHN001 is unified against the ⟨ 0 loc cat subcat 1 ⟩ graph of the constraint graph contained in *PERSUADE001. This succeeds and the result graph is stored in *PERSUADE001. When the rest of the sentence is input, *GIVE gets partially saturated with *SANDY and *SUSHI and while the subject is still unfilled is grabbed by the subject control verb head *to* which is in turn grabbed by object control verb head *persuade* (*PERSUADE001). With the graph con-

[5]Except for one significant difference which is that the constraint 0 graph actually contains pointers to the real instances in memory (such as Mary001) instead of a simple string (such as "Mary") found in unification-based parsers.

[6]If an instance of *JOHN already exists then both are considered as candidates with currently existing instance with less cost assigned. Cost (and reverse cost) information is used for phonemic confusions, lexical ambiguity, and multiple instance candidates and recurrent net subsymbolic conceptual predictions in the MONA-LISA system which is not discussed in this paper.

vergence specified in the constraint graphs representing control relations both *TO001 and *PERSUADE001 get their subject and object positions filled respectively with the appropriate instance MARY001. At the end of the recognition only *PERSUADE001 moves to the DecayingLayer (only fully saturated head-instance) and the synsem graph of *PERSUADE001 contains the fully built result of the recognition.

## Discussion:

We have seen in the previous sections that our model is capable of taking advantage of constraints that are postulated in modern linguistic theories by fully supporting arbitrary graphs to be used as constraints in the network. In fact our current implementation uses HPSG which is represented by path equations (which are automatically converted to directed subsumption-ordering graphs[7]) that are no different from the grammar that we are using for the unification-based parsers. This is in contrast to the past attempts in augmenting the memory-based paradigm with syntax (such as [Tomabechi and Levin, 1989]) where due to the architectural limitation of the memory-based system, full configurational constraint processing could not be supported. Given that our model is basically memory-based (i.e., spreading activation on memory) advantages of the memory-based recognition (such as direct memory access inferences on episodic memory discussed extensively in the existing literature) are maintained on top of the capacity to handle syntactic constraints. Moreover, clearly memory-based (DMA) network would significantly overgenerate without strong syntactic constraints as demonstrated in our example of *object control* phenomena. Without the *control* constraint, the subject of an embedded clause could be any instance that meets semantic requirements (for example *JOHN001 and *MARY001). Seemingly simpler phenomena such as agreement of *case* are equally problematic for the memory-based systems since the *case* of a noun phrase (entity) is not determined until the actual time of utterance and often *case* requirements are postulated from external clauses. In other words, without strict syntactic processing, there will be a massive overgeneration of acceptable candidates with semantics and phrasal lexicon alone with a realistic grammar. This means that naive acceptance of the DMA style marker-passing scheme as an extremely fast way of natural language recognition needs to be reexamined. Application of fragments of knowledge could be very fast in a simple marker intersection search, but with a realistic linguistic coverage that in return means massive overgeneration which cannot be solved by increase in number of processing units and parallelism.

Another important advantage of our model is that the singularity of the control structure and the uniformity of the processing are maintained. Past attempts to enhance spreading activation marker passing recognition to handle syntactic constraints either inevitably involved introduction of arbitrary functional applications to handle syntax (such as [Tomabechi and Levin, 1989]) or required a separate control structure (i.e., syntactic parsers) to handle syntax (such as [Norvig, 1989] and [Hendler, 1989]). In the graph-based constraint propagation network, the control structure is the propagation of constraint graphs which is uniform. The constituent build-up for subsumption-ordering graphs and the constraint applications are also uniformly performed by graph-unification requiring no prestored and arbitrary functional constraint checking mechanism. One rather practical advantage due to the support of graph unification as the constraint application mechanism is that the result of configurational buildup of the subsumption-ordering graph (as postulated in unification-based linguistic theories) is incrementally created during the recognition. These results of constituent buildup are stored in the path 0 graph in our implementation. (Path 0 represents the constraint to the node itself, whereas paths 1, 2, and 3 represent the constraints for its complements). Thus, at the end of the sentential recognition, the constraint 0 graph of the head-instance which survived through constraint propagation and moved into the DecayingLayer represents the feature-structure representation of the result of the recognition. This result graph can be extracted out and be passed to other natural language and inferential modules if necessary.

## Conclusion

The paradigm of natural language recognition based upon the direct recognition in a semantic memory has been appealing from different view points. These included the appeal from the cognitive view-points as well as the phenomenological ones (especially the assumed massive-parallelism). The paradigm's practical appeal as a parsing architecture has been strong as well due to its strength in handling contextually sensitive inputs. However, one significant weakness of this paradigm has been its lack of capacity to handle syntactic constraints in a manner that is consistent with the rest of the memory structure. Linguistic and psychological communities have long accepted the significance of syntactic constraints as playing an important role in many types of linguistic phenomena. On the other hand, with few exceptions, syntactic constraints have been either ignored or handled in an *ad hoc* manner. Our model attempts to model interactions among various syntactic constraints (as represented by graphs) as well as between syntax, semantics and pragmatics in a principled manner through a uni-

[7]Subsumption ordering is an assumed condition for the graph-based representation in unification-based grammar formalisms. Simply put, the information content represented by the directed graph node which is closer to the root of the graph subsumes the information content of the nodes further from the root in the same graph.

form representation and a singular control structure of constraint propagation. It should be emphasized that the syntactic processing is fully integrated into semantic processing since both semantic and syntactic constraints are represented through directed graphs. The processing itself is semantic-driven since the graph propagation in the semantic network is the underlying control structure.[8] In our model, phenomena such as *case*, *agreement*, *control*, *binding*, and *long-distance dependency* can be handled in a generalized manner most straightforwardly as formulated by the modern linguistic theories based upon feature structures and unification. The difference between this model and past 'syntactic daemon' models is substantial both in terms of the expressivity and generality of constraint applications. Generalized linguistic phenomena such as those mentioned above are represented and processed in a principled manner compared to an *ad hoc* and specialized manner associated with the prestored *daemon* models[9]. With the expressivity of our model in capturing the syntactic constraints as postulated in the modern linguistic theories and uniformity of processing them that is integrated to a spreading activation memory-based natural language recognition in a principled manner, our model seems viable as one paradigm of natural language recognition.

## Acknowledgments

The author would like to thank Masaru Tomita, Jaime Carbonell, David Evans, Alex Waibel, Hiroaki Kitano, and Margalit Zabludowski for their comments. Special thanks are due to Akira Kurematsu, Hitoshi Iida and the members of the ATR Interpreting Telephony Research Laboratories for their support while I was visiting ATR implementing the central part of the architecture.

## Implementation

We have implemented our demo system on a Sequent Symmetry shared-memory parallel machine with 16 CPUs by simulating massive-parallelism by taking advantage of micro-tasking parallelism using light-weight processes in parallel CommonLisp. The activation in the GCPN is only propagated upward in the inheritance hierarchy and never horizontally. Since the increase in the grammar size takes place horizontally, the complexity increase can be essentially countered by an increase in the number of processing units.

## References

[Bresnan and Kaplan, 1982] Bresnan, J. and R. Kaplan "Lexical-Functional Grammar: A Formal System for Grammatical Representation". In J. Bresnan (ed) *The Mental Representation of Grammatical Relations*, MIT Press, 1982.

[Hendler, 1989] Hendler, J. "Marker-Passing over Microfeatures: Towards a Hybrid Symbolic / Connectionist Model". In *Cognitive Science*, Vol. 13. Num 1, 1989.

[Karttunen, 1986] Karttunen, L. "D-PATR: A Development Environment for Unification-Based Grammars". In *Proceedings of COLING-86*. 1986.

[Kasper, 1987] Kasper, R. "A Unification Method for Disjunctive Feature Descriptions". In *Proceedings of ACL-87*. 1987.

[Kitano, 1990] Kitano, H. *PhiDMDIALOG A Speech-to-Speech Dialogue Translation System* Machine Translation 5 (1990), 301-338. Kluwer Academic Pub., 1990.

[Martin, 1989] Martin, C. "Case-based Parsing" in Riesbeck, C. and R. Schank eds., *Inside Case-based Reasoning*, Lawrence Erlbaum Associates.

[Norvig, 1989] Norvig, P. "Marker Passing as a Weak Method for Text Inferencing". In *Cognitive Science*, Vol. 13, Num. 4, 1989.

[Pollard and Sag, 1987] Pollard, C. and I. Sag *Information-based Syntax and Semantics*. Vol 1, CSLI, 1987.

[Pereira, 1985] Pereira, F. "A Structure-Sharing Representation for Unification-Based Grammar Formalisms". In *Proceedings of ACL-85*. 1985.

[Quillian, 1966] Quillian, M.R. *Semantic Memory*, Doctoral Dissertation, Carnegie Institute of Technology (Carnegie Mellon University), 1966.

[Quillian, 1969] Quillian, M.R. *The teachable language comprehender*. BBN Scientific Report 10, 1969.

[Riesbeck and Martin, 1985] Riesbeck, C. and C. Martin *Direct Memory Access Parsing*. Yale University Report 35, 1985.

[Shieber, *et al*, 1983] Shieber, S., H. Uszkoreit, J. Robinson, and M. Tyson *The Formalism and Implementation of PATR-II*. In Research on Interactive Acquisition and Use of Knowledge. Artificial Intelligence Center, SRI International, 1983

[Tomabechi, 1991] Tomabechi, H. "Quasi-Destructive Graph Unification". In *Proceedings of ACL-91*, 1991.

[Tomabechi, ms] Tomabechi, H. *MONA-LISA: Multimodal Ontological Neural Architecture for Linguistic Interactions and Scalable Adaptations*, forthcoming as Technical Report, ATR Interpreting Telephony Research Laboratories, 1991.

[Tomabechi and Levin, 1989] Tomabechi, H. and L. Levin "Head-driven Massively-parallel Constraint Propagation: Head-features and subcategorization as interacting constraints in associative memory", In *Proceedings of The Eleventh Annual Conference of the Cognitive Science Society*, 1989.

[Wroblewski, 1987] Wroblewski, D."Nondestructive Graph Unification" In *Proceedings of AAAI87*. 1987.

[8]In other words, syntax is not simply acting as a filter for marker intersection; instead, both syntactic and semantic constraints are applied through graph propagation and unification constraint satisfaction activities.

[9]which normally works for one or two typical sample sentences alone.