

TR-I-0197

MONA-LISA: Multimodal Ontological Neural
Architecture for Linguistic Interactions and
Scalable Adaptations

Hideto Tomabechi

苦米地 英人

1991,3.4

概要

自然言語の認識過程において、入力言語の認識系と言語解釈による知識構成系とが相補的に作用しあうモデルが考えられる。このモデルを構成する手法について報告する。記号化されない経験的な過程を想定することにより認識系を構成し、グラフに基づく制約伝播ネットワークにより知識構成系を構成する。前者は再起型ニューラル・ネットワークで実現でき、後者は超並列ネットワークで実現できる。両者の情報受渡しを行うエンコーダとデコーダを用意し、これら非記号系と記号系を統合する。この統合により、知識構成系からの予測を使った認識系の実現と、認識系からの入力に従った知識構成系の動的な言語理解状態の実現とが可能になる。

MONA-LISA: Multimodal Ontological Neural Architecture for Linguistic Interactions and Scalable Adaptations

Technical Report

ATR Interpreting Telephony Research Laboratories

Hideto Tomabechi*

*Visiting Research Scientist from Carnegie Mellon University. tomabech@cs.cmu.edu

Abstract

This paper describes an architecture for symbolic and subsymbolic interactions during massively-parallel processing of natural language recognition. The model is centered around a graph-based constraint propagation network which is connected to a recurrent neural network which provides contextually sensitive predictions. The integration of symbolic massive parallelism and subsymbolic neural net PDP processing provides a smooth *a posteriori* learning to the symbolic system and a focused guided learning as well as strong constraints during recognition to the neural network. As the result, the architecture provides the ability to handle strict and structured symbolic constraints during recognition while attaining a smooth contextual prediction applied with a least rigidity and learning sentential regularities from actual dialog samples.

1. Introduction

MONA-LISA stands for Multimodal Ontological Neural Architecture for Linguistic Interactions and Scalable Adaptations. It is a joint project of the Center for Machine Translation of Carnegie Mellon University and ATR Interpreting Research Laboratories. The MONA-LISA architecture has the following characteristics:

- Integration of neural-net based signal processing and constraint-based symbolic processing.
- Massively-Parallel Constraint Propagation Architecture.
- Multi-Modal Input and Output Channels.

Historically MONA-LISA joins two traditions of Massively-parallel cognitive processing, namely, symbolic massive-parallelism and subsymbolic parallel distributed processing. As a model of symbolic massively-parallel processing, MONA-LISA follows the tradition of memory-based processing that was originated by [Quillian, 1968] followed by the Direct Memory Access models developed by [Riesbeck and Martin, 1985], [Tomabechi, 1987], etc.¹. Independently a number of researchers in symbolic massive parallelism demonstrated the strength of spreading-activation and marker-passing approaches especially in terms of contextual inferencing and memory-based natural language recognition. This included the research done by [Hirst and Charniak, 1982], [Fahlman, 1983], [Small and Reiger, 1982], [Charniak, 1983], [Hahn and Reimer, 1983], [Hirst, 1984], [Charniak, 1986], [Hendler, 1986], [Charniak and Santos, 1987], [Norvig, 1987], [Hendler, 1989], and [Norvig, 1989]. The other tradition of massively-parallel cognitive processing has been the tradition of subsymbolic parallel distributed processing pursued by connectionists including

¹Including [Tomabechi and Tomita, 1988], [Tomabechi and Tomita, 1988b], [Tomabechi, *et al.*, 1988], [Kitano, 1989], and [Tomabechi and Levin, 1989].

the work of [Granger and Eiselt, 1984], [Waltz and Pollack, 1984], [Waltz and Pollack, 1985], [Berg, 1987], [Bookman, 1987], and [Elman, 1988].

The symbolic massive parallelism models' contribution has been their ability to take advantage of memory-based processing by directly spreading activation through an *a priori* prepared conceptual network. Thus, these models have shown strength in handling extra-sentential processing of natural language and memory-based inferential tasks that have been architecturally difficult to achieve in the traditional symbolic models (as demonstrated in [Kitano, 1989], [Tomabechi and Levin, 1989], and [Norvig, 1989]). However, there are some issues in symbolic cognitive processing that the massively parallel models have not addressed effectively so far, namely, 1) learning contextual knowledge from the actual data is difficult; 2) robust application of stored knowledge is difficult; 3) preparing knowledge for different contexts is subject to computational explosion. Learning is difficult because the knowledge prepared in the form of conceptual networks and a grammar of a language is too limited in contrast to the actual variety of linguistic inputs to perform any meaningful inferencing for learning². Application of stored knowledge to the input is rigid in the symbolic models because symbolic matching of stored knowledge has to be exact (as exemplified in unification-based processing) and fixed. In symbolic contextual processing, knowledge for each context has to be prepared separately to accomodate all distinct and acceptable contexts. Thus the rapid growth in the size of contextual knowledge to be prepared is inevitable and actually, very few systems have incorporated any significant amount of contextual knowledge for handling a realistic domain.

Connectionist models have shown the strength of learning from *a posteriori* provided actual training sets without being provided declarative knowledge *a priori*. This has contributed to desirable performance in a number of areas including speech and visual

²For example, if a sentential input to a parsing system is found to be illformed syntactically or semantically, there is no way to determine whether 1) the system should conclude that it needs to modify its grammar under the particular context; 2) its grammar is incomplete regardless of the context; 3) the input is simply illformed.

recognition. Since neural net learning is performed in a smooth activation space effectively generalizing the patterns in the input data, the application of input vectors does not need to be exact and rigid. Also, the recent work in recurrent neural network has shown the effectiveness of learning time-differentiated contextual sensitivity of input activations ([Elman, 1988], [Jordan, 1986]). Thus, the characteristics of connectionist models seem desirable for handling contextually sensitive inputs. However, a major obstacle in using these models for abstract cognitive processing (instead of low-level signal processing) has been that the neural net representations are fully distributed and also that learning is stored in the hidden layers as dynamic patterns of activations. Thus, neural networks have been effectively applied as robust vector pattern discrimination modules (such as for discriminating consonants from fourier transformed vector patterns) that are recognizable through output layer activation patterns, but the context (time) sensitive learning captured in the hidden layer has not been used for symbolic contextual processing.

Another problem in using subsymbolic connectionist learning for symbolic contextual processing is due to the difference in the granularity of the massive-parallelism. Symbolic massive-parallelism (spreading activation, marker-passing and constraint-propagation models) require medium to sometimes large grain size in their parallelism to handle the somewhat complex constraints required in the symbolic inferential tasks. On the other hand, parallelism in PDP neural net models requires much finer granularity. It is because each unit in the network is a mere vector location and activities never require complex functional applications. With these difficulties, we still find the appeal of the cooperative processing at symbolic and subsymbolic levels to be significant. If such an integration is attained, then symbolic cognitive processing will have the capacity to handle contextual sensitive inputs with smooth *a posteriori* learning and robust knowledge applications. It will also become possible for the neural net subsymbolic learning and recognition to take advantage of declarative symbolic *a priori* constraints as *focus of attention* to conduct its learning and as enhancement during learning and recognition activities.

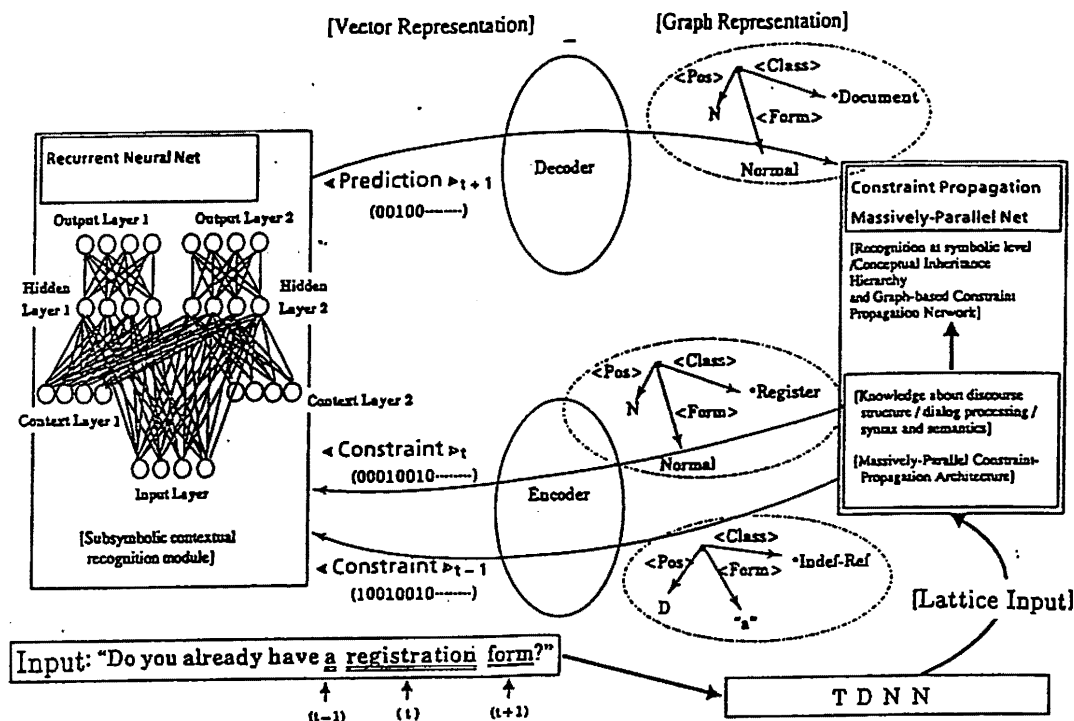


Figure 1: MONA-LISA Architecture

2. Multimodal Ontological Neural Architecture

MONA-LISA is the result of our efforts to connect symbolic and subsymbolic cognitive processing during natural language recognition.

Figure 1 shows the conceptual diagram of the MONA-LISA architecture. The left side module is the contextual recognition subsymbolic neural network and the right hand side module is the constraint propagation symbolic conceptual network. The two networks are connected by the vector encoder/decoder modules. The constraint-propagation network is also connected to the Time-Delay Neural Network (TDNN, [Miyatake, *et al.*, 1990], [Sawai, *et al.*, 1989]) speech recognition module. We are also planning to connect the visual recognition neural network under study at our Vision Laboratory. The external signal to the TDNN (and visual NN in the near future) activates the nodes in the constraint propagation massively-parallel symbolic network and the stored constraints that are represented by graphs are propagated in the network. The constraint graphs are directed graphs which can point to any location in the constraint propagation network and represent both linguistic and non-linguistic constraints which are originally provided in

the form of path equations (and are converted into directed graphs). The input recognition in the constraint propagation network proceeds by massively propagating constraints that are activated by input from the signal processing neural network (ie. TDNN) and invokes nodes that received constraints to perform constraint satisfaction activities. When constraints are satisfied, further activations are in order. The constraint propagation network is also organized as an abstraction hierarchy and activations of nodes with low abstraction levels are encoded into vectors through the encoder module and are fed into the contextual recognition neural network. The contextual recognition neural network is a recurrent neural network based on [Elman, 1988] with some modifications to predict further into the future (and to receive priming further into the past). One difference from Elman's original work is that the vectors do not represent specific surface realizations of a particular input (such as a word), instead, its features (syntactic, semantic, etc) are encoded into vectors. Hence, in our model the output predictions can be decoded and are fed back to the constraint propagations as further (reverse) constraints.

The strength of this architecture is that: 1) Any symbolic constraints can be represented in the constraint propagation network as long as the constraints are representable using directed graphs (i.e., unification-based syntax/semantics, semantic networks, logical formulas, etc.). Therefore, both syntactic and semantic lexicon and memory-based contextual knowledge can be represented in a uniform framework; 2) Contextual regularities of input can be *a posteriori* acquired in the contextual recognition network. In the case of dialog processing, we can provide a set of sample dialogs and the contextual recognition network learns from the actual input. This enhances the contextual knowledge provided in the constraint propagation network since the recognition in the recurrent network is fully context sensitive whereas most of the constraints captured in the constraint propagation network are context-free.

3. Graph-based Constraint Propagation Network

The main part of the MONA-LISA architecture is the Graph-based Constraint Propagation Network (GCPN). What is propagated in the GCPN are the graphs³ and they may point to any location in the network and may contain complex paths including convergent arcs and loops. The expressivity of the graph-based constraint propagation scheme is significantly greater than that of so called marker-passing schemes. For example, if we want to represent the *object control* constraint of English in which the object of the external VP is coindexed with the subject of the embedded VP (as in *John persuaded Mary to eat sushi*), using the marker-passing scheme (such as in [Tomabechi and Levin, 1989], and [Kitano, 1989]), we will have to lexically store functional applications of *object control* which are triggered by activations of object control verbs. It is because in marker-passing schemes, markers can simply store bundles of feature and value pairs which are simple (i.e., cannot be complex structures to satisfy path equations) and local (i.e., they do not point anywhere in the global network). We can also view marker-passing models as strongly restricted versions of GCPN where graphs are only allowed to be one level deep without convergence or loops, and are not allowed to point to locations in the network.

If we allow arbitrary directed graphs to be passed around in the network, constraints such as *control* can be handled straightforwardly. For example, in case of *object control* verbs, we will only need to store graphs with arcs converging on the same variable that corresponds to the object of the external VP and the subject of the matrix VP (as is done in the lexical specification of this constraint in the framework of unification-based linguistic processing). Actually in practice, in handling linguistic constraints such as *control* we will only need to store graphs that are converted from path equations that are used in the unification-based grammar. We will not need any special functions to be stored for each different type of linguistic phenomenon.

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

Below is the sample lexicon from our current MONA-LISA implementation which processes English input. We have adopted HPSG ([Pollard and Sag, 1987]) as the basis of providing the linguistic constraints to the system as graphs. The first example is the specification for the concept *JOHN which represents the set of nodes that satisfies the constraints specified here using path equations:

```
(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 in Appendix I. In the 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. 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 actual graph expansion in the GCPN looks as shown in the Appendix I. This way, instead of simply storing simple case-frame type lexical specifications in the lexical nodes, we would like to 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>) ;;; obj control
(<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) ;;; must not unify
(<3 loc cat subcat 3 loc cat head> == saturated) ;;; must not unify
(<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:

$$\begin{aligned}
 (\langle 0 \text{ loc cont persuadee} \rangle &== \langle 2 \text{ loc cont para index} \rangle) \\
 (\langle 0 \text{ loc cont persuadee} \rangle &== \langle 0 \text{ loc cont circumstance agent} \rangle)
 \end{aligned}$$

can easily specify the control constraints lexically in the network. The graph representation for the path equation looks as given in the Appendix I once read into the system. With an addition of a specification for the intermediate subject control VP head *to*, as we will see in Appendix II, the constraints are adequate for handling the *object control* phenomenon. One thing to be noted is that because we use HPSG-based constraints to be specified as graphs in the lexical nodes in the GCPN network, naturally, the lexical nodes look much like HPSG lexical entries. However, the GCPN processing scheme does not assume unification as the only type of graph constraint checking mechanism⁴. More importantly, as we will see in the next section, lexical-nodes are parts of the GCPN network and the network includes constraints at different levels of abstraction and compositionality as well as sentential HPSG-based unification-based grammar syntax/semantics. Also, in

⁴We use graph-unification as currently desirable scheme of checking graph-based constraints, but other method may replace unification in the future implementations.

GCPN whatever is bound to the variables in the constraints graphs are actual instances in memory for the current utterance and are not strings (or symbols) independent of contexts.

4. The GCPN Natural Language Recognition Algorithm

The GCPN 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 TDNN output layer. We will focus activities on lexical nodes and instance nodes in this paper and we will not discuss activities of conceptual-class nodes and phonological nodes. (Please refer to [Tomabeche, *et al.*, 1988] and [Kitano, 1989] for activities of those nodes.)

Below is the central part of our sentential recognition algorithm for word lattice activation from the TDNN.

```
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-lattice in input-stream do
    for word-hypothesis in word-lattice 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-head
    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 (unlike many marker-passing models). Conceptual relations (other than inheritance) between lexical nodes are specified through constraint graphs (as seen in the sample entry in the previous section). For example, Figure 2 is the part of the GCPN that participates in the sentential recognition of the input *John persuaded Mary to give Sandy sushi* which encompasses two control relations (i.e., *persuaded* object controls *Mary* and *to* subject controls *Mary*).

Appendix II provides a sample output from our current implementation on a shared memory parallel hardware⁵. In the sample output, the input to the system is a correctly hypothesized word instead of a word lattice. When each word is input, an instance of

⁵We use 16 CPU Sequent Symmetry running Dynix parallel Unix. Concurrent processes are created as light-weight-processes *lwps* on a parallel CommonLisp running on Sequent.

Graph-based Constraint Propagation Network:
Final state after recognition of "John Persuaded Mary to give Sandy Susi"

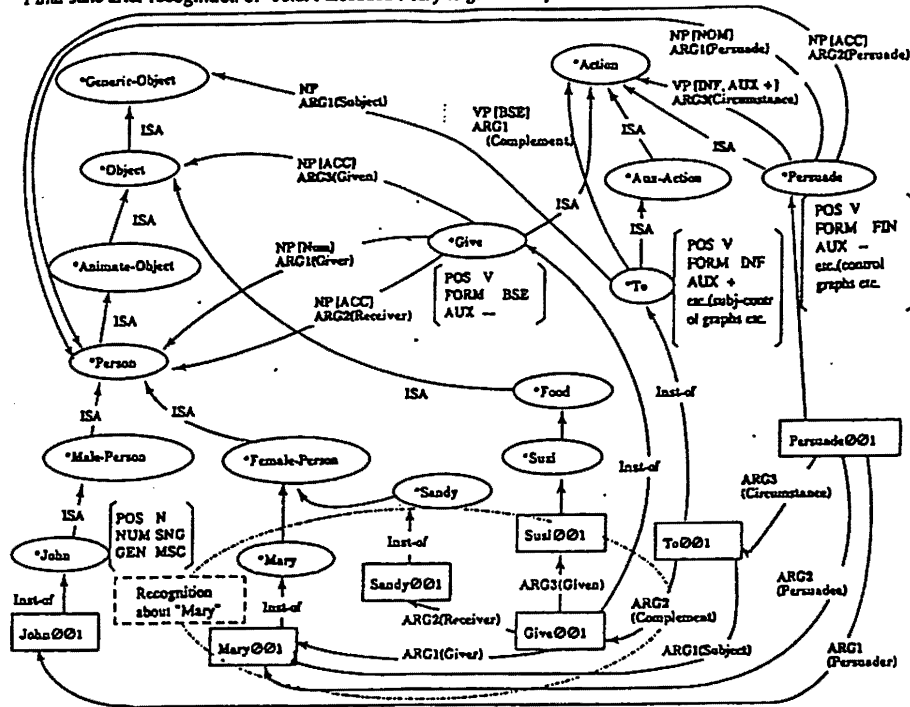


Figure 2: A Portion of GCPN representing *control* relation

the corresponding lexical node is created. Also the stored constraint graph is copied and modified to point to the created instance. If the lexical node is a complement then the graph is simply propagated upward and the global massively parallel activity is invoked when the activation reaches the top of the hierarchy. If the corresponding lexical node is a head-complement then the global incidents are invoked immediately. During the global invocation, instances of activated lexical-head nodes try to fill the complements (subcategorization elements) by unifying the stored subcat graphs with the constraint graphs propagated from the activated lexical complements. If the subcat list saturates in the activated-head instances through successful unifications, then the node states decay (put to DecayingLayer). All nodes originally belong to StaticLayer and when they are lexically invoked their state changes into activated (moves to ActivationLayer). Only the saturated lexical-head nodes move to the DecayingLayer. So at the end of the recognition, the constraint 0 graphs (i.e. the constraint to itself) of the nodes in the DecayLayer contain information that can be used for further processing (such as generation). Actually, as we can see from the sample output, the printout of the constraint 0 graph of the Decaying node looks exactly like that of the output of a unification-based parser. This is expected because our graph constraints propagated in the network are initially prepared accord-

ing to the unification-based grammar constraints (HPSG). One thing to be reiterated here is that there still is an underlying difference between this model and the unification-based parsing schemes in 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"). Therefore, the MONA-LISA scheme allows for different kinds of memory-based and contextual inferences at any point of recognition activity. We have already stated our efforts to connect the vision neural network to the system so that some of the memory instances actually receive the activations from visual inputs as well. Another thing to note is that during these recognition activities many nodes in the network receive priming activations from the conceptual recognition neural network, which is the topic of discussion in the next section.

5. Integrating Subsymbolism with Symbolism

The advantage of integrating symbolic massive parallelism with the subsymbolic massive parallelism seems clear. First, such a system can provide a *posteriori* acquired contextually sensitive recognition learned in a smooth activation space, whereas *a priori* given or derivable symbolic knowledge may be rigid and sometimes *ad hoc*. Second, the neural net learning is meaningless unless we can provide what to learn. In other words, it is the traditional *focus of attention* problem that the external world contains too much vectors to learn and without a strongly constrained focus of attention, learning by neural network may never converge. Even if a fast neural network learns extremely large amounts of long vector to vector matching, learned subsymbolic knowledge lies in the hidden layer as vector patterns which are inaccessible externally. We would like to integrate the neural network as a part of the constraint propagation network instead of having it as a separate module that performs simple signal processing and pattern discrimination. We will be reviewing our scheme for this purpose in the first subsection. Another obstacle in integrating symbolism and subsymbolism is that of parallelism. Symbolic constraints require

medium to large grain processing and are not postulatable using a fully distributed fine grain parallel architecture (i.e., standard PDP architecture). On the other hand, neural-net learning requires a fully distributed PDP architecture and granularity of parallelism is very fine.

5.1. Our Scheme for Neural-Net/Symbolic Net Integration: Participation of Recurrent Net

We would like to have the subsymbolic learning module assume the role of providing contextually sensitive recognition and priming based on the actual input data (i.e., dialogs in terms of natural language systems). Since this requires the introduction of time differentiation, we have adopted Elman's recurrent neural network ([Elman, 1988]) as the base of our neural network. We have been experimenting with different modifications of the recurrent network modifying them to predict different time spans into the future ($t+n$) and to receive hidden layer activity from different time spans from the past ($t-n$)⁶. For example, the model that predicts $t+1$ and $t+2$, which is currently adopted for MONA-LISA, looks as provided in the Figure 1.

In order to attain interaction with the symbolic GCPN, we introduced explicit encoding (and decoding) of constraint graphs into vectors. By encoding the graphs into vectors and decoding them, learning by the recurrent network can be accessible directly from the output layer, whereas if we simply provide token vectors for unencoded surface strings, the result of learning has to be extracted from the hidden layer (by methods such as hidden cluster analysis).

⁶A report on the experiments on different configurations of the recurrent network is forthcoming as our technical report.

5.1.1. Encoding CPN Constraints: Syntax

Each lexical entry in our system is encoded as a 45 position vector, where each position is filled by either a 1 or a 0. The first 20 positions contain syntactic (head-feature) information while the rest represent semantics. The head-features of a word determine many of the syntactic properties of the phrase which is headed by that word ([Jackendoff, 1977]). The specific constraint features encoded in our lexical entries are based on those postulated in the HPSG ([Pollard and Sag, 1987]) and are specified in the propagated graphs in the GCPN. The first six vector positions represent its the activated lexical node's major category⁷ (MAJ for major) which may be one of the following: Noun (N), Verb (V), Adjective (A), Preposition (P), Determiner (D), and Adverb (ADV). The next seven vector positions of each lexical entry represent its form (FORM). The possible values of FORM vary depending on the word's major (MAJ) category. Thus, a verb may have one of the following seven forms: Finite or tensed (FIN), Base (BSE), Past Participle (PSP), Present Participle (PRP), Passive (PAS), Infinitival (INF), and Gerundive (GER). For nouns, on the other hand, we distinguish five different forms: the expletive pronoun *there*⁸ (THERE), the expletive pronoun *it* of extraposition⁹ and pseudocleft¹⁰ constructions (IT), non-reflexive pronouns (PRO), reflexive pronouns (ANA), and all other nouns (NORM). There are as many preposition forms as there are prepositions. The same holds for determiners. In our current implementation (due to economy reasons), we distinguish the six most frequently used prepositions, while all other prepositions are encoded as 'other'. The same holds for determiners. The head-features encoded in the remaining 7 syntactic positions are different for each grammatical category. Thus for example, a 1 occupying the 14th vector position indicates in the case of a noun that its case is Nominative (NOM)¹¹,

⁷The major categories correspond to the notion of part-of-speech.

⁸For example, as in existential constructions such as *There is a moon out tonight*. (Examples here are from [Pollard and Sag, 1987].)

⁹For example, as in *it bothers me that he resigned*.

¹⁰For example, as in *it's bagels that I want*.

¹¹In English, only pronouns exhibit overt case marking differences; e.g., *he* (Nominative) versus *him* (Accusative).

in the case of a verb that it is an Auxiliary verb (AUX+)¹², and in the case of an adjective or a preposition that it is Predicative (PRD+)¹³.

The 20-unit representation of the head-feature information associated with verbs is: #(N V A P D ADV FIN BSE BSP PRP PAS INF GER AUX+ INV+ 1ST 2ND 3RD SNG PLU) and that of nouns is: #(N V A P D ADV THERE IT NORM PRO ANA _ _ NOM ACC 1ST 2ND 3RD SNG PLU).

For example, with our encoding scheme, the syntactic part of the lexical vector for the word *attends* is as shown below¹⁴:

#(0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0).

The first 6 units indicate that its Major category is V, the next 7 units indicate that its V FORM is BSE, and the rest indicating that it is AUX- and INV-¹⁵, and that its agreement values are: 3rd person, singular¹⁶.

5.1.2. Encoding CPN Constraints: Inheritance Hierarchy

The latter half of the lexical vectors represent the semantics of the lexical-entry in terms of its location in the conceptual inheritance hierarchy. The inheritance hierarchy is represented by groups of units (vectors) representing the branching from the previous layer

¹²Such as the Auxiliary verb *will* in *John will come*.

¹³For example, the adjective *ajar* in *The door was ajar* and the prepositional phrase with the preposition *on* in *Felix is on the oak library table*.

¹⁴We will be using #(...) instead of [...] to represent vector to be consistent with our sample outputs.

¹⁵This feature is necessary to distinguish auxiliary verbs that invert (most of them do) from those that do not, such as the verb *better* as in *You better stay here* (see [Pollard and Sag, 1987]).

¹⁶Gender information is not encoded as a syntactic head-feature because it would be redundant in the case of English since English is a natural gender language and gender information is encoded in the semantics portion of the vector.

to the next layer. Currently we have 5 groups of 5 units each representing one level of abstraction in the inheritance hierarchy. The location of a unit whose value is 1 represents the branching for the next layer. For example, a simplified part of our inheritance hierarchy looks as shown below (descending levels of abstractions from left to right):

1st level	2nd level	3rd level	4th level	5th level
Object	Physical-Object	Animate-Object	Person	Male-Person
		Inanimate-Obj	Natural-Subst	Stones
		:	Artificial-Subst	Document
			:	:
	Mental-Object	Theory&Rule ...		
		Abstract-Tool	Language	English
		:	:	Japanese
			:	
	Social-Object ...			
Phenomenon	Physical-phenomenon			
Attribute				
Force				

From the highest level to the 5th level, each level has 5 categories and therefore, about 3,000 concepts (5^5) are representable with 25 units. For example, the semantic part of the lexical-vector for *Japanese* would be:

#(1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0); the value 1 indicating the branching for the next level down (left to right).

5.1.3. Decoding the Encoded Vector

We apply the following rules to decoded the output (output layer vector) of the recurrent net forward propagation:

Lexical-Vector Decoding Rule

- Apply predetermined threshold (currently 0.8 in our system) to the output levels of each vector position.
- Decode syntactic part from left to right. If Major category is ambiguous (i.e., more than one unit is over the threshold in the first six positions, or none are above the threshold), then syntax is ambiguous. If Major category is unambiguous, decode other head-features by checking the vector position for each feature.
- Decode the semantic part from left to right level by level. If the output is ambiguous (more than one or none above the threshold) at any level, stop decoding immediately.

This left to right decoding of the syntax and semantics vector guarantees that the decoding always returns its most unambiguous hypothesis for each output configuration. For example, if the output after applying the specific threshold is: #(1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0) we decode the syntactic part to be ambiguous, because the first six units indicate that it can be either a Noun or a Preposition (because of previous words, that next word can be hypothesized to be either N or P). The semantic part is decoded as *Artificial-Substance*, because we can decode from left to right as *Object*, *Physical-Object*, *Inanimate-Object*, *Artificial Substance* climbing down the abstraction hierarchy; however, one more level down, it is now ambiguous between *Document* and something else. Therefore, our decoding stops at *Artificial-Substance*.

5.1.4. Recurrent Net Priming

In the MONA-LISA system, some nodes that are one to two levels higher than the lexical-nodes in the abstraction hierarchy are designated as 'contextual interaction nodes' which

send and receive activations to/from the recurrent network. We have trained the recurrent network by supplying actual dialog sample sentences into the system. We have 12 dialogs of telephone conversations in the domain of international conference registration prepared based on actual telephone conversations. Each sentence is 10 to 15 words long and each dialog contains 10 to 20 sentences. We treated one dialog as one epoch and a training set consisted of 3 sets of 12 dialogs ordered pseudo randomly. During the sentential activations, when a particular lexical node is activated and the activation is passed upward by propagating constraint graphs, if the 'contextual interaction nodes' are activated, then head features¹⁷ and inheritance information of the constraint graphs are encoded into vectors and are supplied to the recurrent network. A forward propagation is performed (followed by backpropagation if it is a part of a training session) and the output vector is decoded and fed back to the constraint propagation network. This activation from the recurrent network is used as reverse costs in the GCPN to be used in subsequent disambiguations at different levels of abstractions. The sample output in Appendix III shows the actual output from the recurrent network during the recognition of one dialog. The current sentence is *I would like*¹⁸ *to register for the conference*. As we can see when *to* is input the next inheritance class concept **register* is actually predicted along with the correct syntactic feature prediction for ((MAJ V) (FORM BSE)). This way the prediction from the recurrent network may be strongly specialized when the activation pattern in the training data is specialized. When we retrained the same network with sentences that included *I would like to* but with other verbs after *to* the network either predicted the concepts that are higher in the abstraction or did not predict anything at all. Since our inheritance decoding is from left to right (higher in abstraction to lower), in effect, some inductive generalization is attained by training with a different variety of sentences. Also, when the sentential activation patterns are specialized, the recurrent network fine tunes to the provided patterns and the predictions may be strongly specialized. The impact

¹⁷We do not vectorize all graph structures simply of economy reasons because recurrent net training takes large amount of time.

¹⁸*Would like* is treated as one unit when recurrent network is activated.

of this is that the recurrent network priming mechanism can be utilized in fine-tuning the system's performance based on the actual usage patterns when the system is finally installed for some specific applications. By running the backpropagation while actually using the system, the system is capable of tuning to the actual sentential patterns used at the specific site. Also, as already demonstrated by [Elman, 1988] the recurrent network successfully learns syntax. As shown in the sample output after the recognition of *I would like to* the network actually predicts the next lexical input to have the syntactic major category Verb with the Base form. The implication of this should be significant especially considering the fact that past natural language systems always *a priori* provided context-free grammars (perhaps, partially motivated by the fact that syntactic knowledge is believed to be innate).

We found some interesting predictions made by the recurrent network. Often the prediction for the next word may be inaccurate but the prediction for the word after next can be highly accurate. Also, as seen in the *register example, the predictions can be too strong due to the size of the training data. Also, we have not succeeded so far in performing credit assignment for the particular vector patterns from time $t-n$ to appear in time $t+m$. We are currently experimenting with different weight connections from different times ($t-n$) with a multiple number of context layers to explore this question further.

6. Conclusion

The MONA-LISA architecture assumes multimodal input/output activity and aims at receiving input activations from the neural network into the symbolic massively-parallel network. The interaction within GCPN and with the contextual recognition recurrent neural network is performed in terms of propagating constraints that are provided as

graphs or converted from vectors to graphs.¹⁹ As we can see, 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 countered by an increase in the number of processing units.

We have seen that the neural net can actually provide valuable generalizations learned over real dialog samples to the symbolic network. This was attained because the symbolic network provided the neural network with the encoded constraints to be learned during the actual symbolic constraint propagation activity. MONA-LISA seems to demonstrate that cooperative symbolic/subsymbolic activities are possible in the way that such activities attain the performance that either systems were capable of attaining without the other.

Finally, there is one underlying implication of connecting subsymbolic processing with a symbolic one. It is the implication that we are connecting the physical signal recognition activity effectively with more conceptual symbolic inferencing activity. In other words, attaining the integrated subsymbolic/symbolic processing by connecting symbolism with subsymbolism should mean that we have one model that proposes a scheme to connect the activities in physical world with the activity in the abstract world.

¹⁹Also, in terms of the implementational problem due to the different grain sizes of parallelism, although we did not discuss it in this paper due to space limitations, we have countered this question by inserting the intermediate light-weight processes to handle different levels of parallelism. Namely, we have divided the parallel processing into three levels:

- **Node level:** this is the level where nodes receive and fire activations, i.e., the representational level of memory nodes. In the past models of massively-parallel artificial intelligence, this was assumed to be the level of processing as well.
- **Light weight process (lwp) level:** this is the level at which massively parallel processing is performed. Any number of *lwps* may be created during processing, independently of the number of nodes and the number of processing units.
- **Processing unit level:** this is the level of actual processing hardware. Any number of processors may be configured depending on the hardware architecture. One (or more) processing unit may be dedicated to the scheduling of *lwps*.

References

- [Berg, 1987] Berg, G. "A Parallel Natural Language Processing Architecture with Distributed Control". In *Proceedings of the CogSci-87*, 1987.
- [Bookman, 1987] Bookman, L.A. "A Microfeature Based Scheme for Modeling Semantics". In *Proceedings of the IJCAI-87*, 1987,
- [Charniak, 1983] Charniak, E. "Passing Markers: A theory of Contextual Influence in Language Comprehension". *Cognitive Science* 7, 1983.
- [Charniak and Santos, 1987] Charniak, E and Santos, E. "A Connectionist Context-Free Parser Which is not Context-Free, But Then It is Not Really Connectionist Either". In *Proceedings of the CogSci-87*. 1987.
- [Charniak, 1986] Charniak, E. "A neat theory of marker passing". In *Proceedings of the AAAI-86*. 1986.
- [Chomsky, 1975] Chomsky, N. "Questions of the form and interpretation". *Linguistic Analysis* 1, 1975.
- [Dowty, et al., 1988] Dowty, D.R., Wall, R.E., and Peters (1981) *Introduction to Montague Semantics*, D.Reidel Pub., 1981.
- [Elman, 1988] Elman, J. *Finding structure in time*. CRL TR-8801. Center for Research in Language, University of California, San Diego, 1988.
- [Fahlman, 1983] Fahlman, S.E. *NETL: A system for representing and using real-world knowledge*. The MIT Press, 1983.
- [Fahlman, 1988] Fahlman, S. E. 'Faster-Learning Variations on Back-Propagation: An Empirical Study' In *Proceedings of the 1988 Connectionist Models Summer School*. 1988
- [Granger and Eiselt, 1984] Granger, R.H., Eiselt, K.P. "The parallel organization of lexical, syntactic, and pragmatic inference processes" In *Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing*, 1984.
- [Hahn and Reimer, 1983] Hahn, U. and Reimer U. *World expert parsing: An approach to text parsing with a distributed lexical grammar*. Technical Report, Universitat Konstanz, West Germany, 1983.
- [Hendler, 1986] Hendler, J. *Integrating Marker-Passing and Problem Solving: A Spreading Activation Approach to Improved Choice in Planning*. Department of Computer Science, University of Maryland, 1986.
- [Hendler, 1989] Hendler, J. "Marker-Passing over Microfeatures: Towards a Hybrid Symbolic / Connectionist Model". In *Cognitive Science*, Vol. 13. Num 1, 1989.
- [Hirst and Charniak, 1982] Hirst, G. and Charniak, E. "Word Sense and Slot Disambiguation". In *Proceedings of AAAI-82*, 1982.
- [Hirst, 1984] Hirst, G. "A Semantic Process for Syntactic Disambiguation". In *Proceedings of AAAI-84*, 1984.
- [Jackendoff, 1977] Jackendoff, R. *X-Bar Syntax: A Study of Phrase Structure*, MIT Press, Cambridge, 1977.

- [Jordan, 1986] Jordan, M. *Serial Order: A parallel distributed processing approach*. Technical Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.
- [Kitano, 1989] Kitano, H. 'Massively Parallel Model of Simultaneous Interpretation: The *PhiDMDIALOG* System,' Technical Report CMU-CMT-89-116, 1989.
- [Kitano, Tomabechi, and Levin, 1988] Kitano, H., Tomabechi, H., and Levin, L. "Ambiguity Resolution in the *DmTrans Plus*". In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, 1988,
- [Martin, 1989] Martin, C. "Case-based Parsing" in Riesbeck, C. and Schank, R., eds., *Inside Case-based Reasoning*, Lawrence Erlbaum Associates.
- [Miyatake, et al., 1990] Miyatake, M., Sawai, H., Minami, Y., and Shikano, K. "Integrated Training for Spotting Japanese Phonemes Using Large Phonemic Time-Delay Neural Networks". In *IEEE Proceedings of International Conference on Acoustics, Speech and Signal Processing, S8.10, Vol 1 (ICASSP'90)*, 1990.
- [Montague, 1970] Montague, R. "English as a formal language", *Linguaggi nella e nella Tecnica*, 1970. Reprinted in *Formal Philosophy: Selected Papers of Richard Montague*, ed. R.H. Thomason, Yale University Press, 1974.
- [Norvig, 1987] Norvig, P. "Inference in Text Understanding". In *Proceedings of the AAAI-87*, 1987.
- [Norvig, 1989] Norvig, P. "Marker Passing as a Weak Method for Text Inferencing". In *Cognitive Science*, Vol. 13, Num. 4, 1989.
- [Nyberg, 1989] Nyberg, E. *The HyperFrame User's Guide Version 1.0*. Technical Memo. Cognitive Research Laboratories. 1989.
- [Pinker and Prince, 1988] Pinker, S., and Prince, A. "On language and connectionism: Analysis of a parallel distributed processing model of language acquisition". In Pinker, S., and Mehler, J. ed., *Connections and Symbols*, MIT Press, 1988.
- [Pollard and Sag, 1987] Pollard, C. and Sag, A. *Information-based Syntax and Semantics*. Vol 1, CSLI, 1987.
- [Quillian, 1968] Quillian, M.R. "Semantic Memory". In *Semantic Information Processing*, ed. Minsky, M. MIT Press, 1968.
- [Quine, 1971] Quine, W. V. "Two dogmas of empiricism". In *Readings in the Philosophy of Language*, ed. Rosenberg, J. F., and Tavis, C.. Prentice-Hall, 1971.
- [Riesbeck and Martin, 1985] Riesbeck, C. and Martin, C. *Direct Memory Access Parsing*. Yale University Report 35, 1985.
- [Sawai, et al., 1989] Sawai, H., Waibel, A., Haffner, P., Miyatake, M., and Shikano, K. "Parallelism, Hierarchy, Scaling in Time-Delay Neural Networks for Spotting Japanese Phonemes / CV-Syllables". In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 1989.
- [Servan-Schreiber, et al., 1988] Servan-Schreiber, D., Cleeremans, A., and McClelland, J. 'Encoding sequential structure in simple recurrent networks'. CMU-CS-88-183, Carnegie Mellon University, 1988.

- [Small and Reiger, 1982] Small, S. and Reiger, C. "Parsing and comprehending with word experts (a theory and its realization)". In *Strategies for natural language processing*. Eds. Lenhart G. and Ringle M. Lawrence Erlbaum, 1982.
- [Tomabechi, 1987] Tomabechi, H. "Direct Memory Access Translation". In *Proceedings of the IJCAI-87*, 1987.
- [Tomabechi and Levin, 1989] Tomabechi, H. and Levin, L. "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.
- [Tomabechi and Tomita, 1988] Tomabechi, H. and Tomita, M. "The Integration of Unification-based Syntax/Semantics and Memory-based Pragmatics for Real-Time Understanding of Noisy Continuous Speech Input". In *Proceedings of the AAAI-88*, 1988.
- [Tomabechi and Tomita, 1988b] Tomabechi, H. and Tomita, M. "Application of the Direct Memory Access paradigm to natural language interfaces to knowledge-based systems". In *Proceedings of the COLING-88*, 1988.
- [Tomabechi and Tomita, 1989] Tomabechi, H. and Tomita, M. "The Direct Memory Access Paradigm and its Application to Natural Language Processing". In *Computers and Artificial Intelligence*, Vol 8, 1989.
- [Tomabechi, et al., 1988] Tomabechi, H. Mitamura, T. and Tomita, M. "Direct Memory Access Translation for Speech Input: A Massively Parallel Network of Episodic/Thematic and Phonological Memory". In *Proceedings of the International Conference on Fifth Generation Computer Systems 1988 (FGCS'88)*, 1988.
- [Tomabechi, et al., 1989] Tomabechi, H., Kitano, H., Mitamura, T., Levin, L., Tomita, M. *Direct Memory Access Speech-to-Speech Translation: A Theory of Simultaneous Interpretation*. CMU-CMT-89-111, Carnegie Mellon University, 1989.
- [Tomabechi, ms] Tomabechi, H.
'Feature-based Dual-Recurrent Neural Network for Symbolic/Subsymbolic Constraint Interactions', Manuscript. Carnegie Mellon University 1990.
- [Wittgenstein, 1933] Wittgenstein, L. "The Proposition, and Its Sense". In *Philosophical Grammar* Ed. Rhees, R.. Trans., Kenny, A., U of California Press., 1974.
- [Waibel, et al., 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. and Lang, K. "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE, Transactions on Acoustics, Speech and Signal Processing*, March 1989.
- [Waltz and Pollack, 1984] Waltz, D. and Pollack, J. 'Phenomenologically plausible parsing'. In *Proceedings of the AAAI-84*, 1984.
- [Waltz and Pollack, 1985] Waltz, D. L. and Pollack, J. B., (1985) 'Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation.' *Cognitive Science* 9(1).

Appendix I:

The output below shows the path equations converted into graph internally in MONA-LISA lexical nodes. Numbers such as X03 are for printing purposes and if more than one number appears in different locations it means that it is a shared-structure (i.e., convergent arcs).

```
(*JOHN
  (INHERITS-FROM *MALE-PERSON)
  (TYPE
    (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]
                                                [GENE 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]]]]]]]]))

(*GIVE
  (INHERITS-FROM *GIVE-ACTION)
  (TYPE
    (VALUE
      (COMMON :LEX-HEAD)))
  (SPELLING
    (VALUE
      (COMMON GIVE)))
  (SYNSEM
    (VALUE
      (COMMON
X01[[3 X02[[LOC X03[[CONT X04[[RESTR X05[[RELN X06 *MATTER]]
      [PARA X07[[INDEX X08 []]]]
      [CAT X09[[HEAD X10[[CASE X11 ACC]
      [MAJ X12 N]]]]]
      [2 X13[[LOC X14[[CONT X15[[RESTR X16[[RELN X17 *PERSON]]
      [PARA X18[[INDEX X19 []]]]
      [CAT X20[[HEAD X21[[CASE X22 ACC]
      [MAJ X23 N]]]]]
      [1 X24[[LOC X25[[CONT X26[[RESTR X27[[RELN X28 *PERSON]]
      [PARA X29[[INDEX X30 []]]]
      [CAT X31[[HEAD X32[[CASE X33 NOM]
      [MAJ X34 N]]]]]

```

```

[0 X35[[LOC X36[[CONT X37[[THEME X08]
    [GOAL X19]
    [AGENT X30]
    [RELN X38 *GIVE-ACTION]]
[CAT X39[[SUBCAT X40[[3 X02]
    [2 X13]
    [1 X24]]
    [MARKING X41 UNMARKED]
    [HEAD X42[[PRD X43 -]
        [INV X44 -]
        [AUX X45 -]
        [VFORM X46 BSE]
        [MAJ X47 V]]]]]]]]))

(*PERSUADED
(INHERITS-FROM *PERSUADE-ACTION)
(TYPE
(VALUE
(COMMON :LEX-HEAD)))
(SPELLING
(VALUE
(COMMON PERSUADED)))
(SYNSEM
(VALUE
(COMMON
X01[[3 X02[[LOC X03[[CONT X04[[RELN X05 *ACTION]
    [RESTR X06[[RELN X05]]
    [AGENT X07]]]]
[CAT X08[[SUBCAT X09[[3 X10[[LOC
    X11[[CAT X12[[HEAD X13 SATURATED]]]]
    [2 X14[[LOC
    X15[[CAT X16[[HEAD X17 SATURATED]]]]
    [1 X18[[LOC
    X19[[CAT X20[[HEAD X21[[CASE X22 NOM]
        [MAJ X23 N]]]]]]
    [HEAD X24[[AUX X25 +]
        [VFORM X26 INF]
        [MAJ X27 V]]]]]
[2 X28[[LOC X29[[CAT X30[[HEAD X31[[CASE X32 ACC]
    [MAJ X33 N]]]
    [CONT X34[[RESTR X35[[RELN X36 *PERSON]]
        [PARA X37[[INDEX X07]]]]]
[1 X38[[LOC X39[[CONT X40[[PARA X41[[INDEX X42]]]
    [RESTR X43[[RELN X44 *PERSON]]]
    [CAT X45[[HEAD X46[[CASE X47 NOM]
        [MAJ X48 N]]]]]
[0 X49[[LOC X50[[CONT X51[[RELN X52 *PERSUADE-ACTION]
    [CIRCUMSTANCE X04]
    [PERSUADEE X07]
    [AGENT X42]]
[CAT X53[[SUBCAT X54[[3 X02]
    [2 X28]
    [1 X38]]
    [MARKING X55 UNMARKED]
    [HEAD X56[[PRD X57 -]
        [INV X58 -]
        [AUX X59 +]
        [VFORM X60 INF]
        [MAJ X61 V]]]]]]]]))

```

Appendix II: Sample output of MONA-LISA system

Allegro CLIP 3.0.3 [sequent] (9/11/90 16:22)
Copyright (C) 1985-1990, Franz Inc., Berkeley, CA, USA
<Initial lwp> (set-numprocs 1) ::: We use only 1 processor for sample output.
1 ::: otherwise we cannot read the formatted output
1 ::: because multiple cpus formats simultaneously.

<Initial lwp> (parse '(John persuaded Mary to give Sandy susi))
Using 1 processor ::: this is the word level input demo (not phonological)

ACTIVATING THE LEXICAL NODE: *JOHN...
A complement instance created: *JOHN2265
Propagating through: *JOHN
Propagating through: *MALE-PERSON
Propagating through: *PERSON
Propagating through: *HUMAN
Propagating through: *ORGANIC-MATTER
Propagating through: *NON-ABSTRACT-MATTER
Propagating through: *MATTER
Propagating through: *GENERIC-OBJECT*
Propagating through: *GENERIC-CONCEPT*
Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *PERSUADED...
A head instance created: *PERSUADED2356
Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *MARY...
A complement instance created: *MARY2384
Propagating through: *MARY
Propagating through: *FEMALE-PERSON
Propagating through: *PERSON
Propagating through: *HUMAN
Propagating through: *ORGANIC-MATTER
Propagating through: *NON-ABSTRACT-MATTER
Propagating through: *MATTER
Propagating through: *GENERIC-OBJECT*
Propagating through: *GENERIC-CONCEPT*
Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *TO...
A head instance created: *TO2468
Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *GIVE...
A head instance created: *GIVE2496
Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *SANDY...
 A complement instance created: *SANDY2524
 Propagating through: *SANDY
 Propagating through: *FEMALE-PERSON
 Propagating through: *PERSON
 Propagating through: *HUMAN
 Propagating through: *ORGANIC-MATTER
 Propagating through: *NON-ABSTRACT-MATTER
 Propagating through: *MATTER
 Propagating through: *GENERIC-OBJECT*
 Propagating through: *GENERIC-CONCEPT*
 Global massive parallelism invoked...

ACTIVATING THE LEXICAL NODE: *SUSI...
 A complement instance created: *SUSI2607
 Propagating through: *SUSI
 Propagating through: *FOOD
 Propagating through: *RELATION-ATTRIBUTE
 Propagating through: *ARTIFICIAL-MATTER
 Propagating through: *NON-ABSTRACT-MATTER
 Propagating through: *MATTER
 Propagating through: *GENERIC-OBJECT*
 Propagating through: *GENERIC-CONCEPT*
 Global massive parallelism invoked...

case information ambiguous.

RecurrentNet: Receiving features: [(MAJ N) (FORM NORM) ((CLASS _*FIRST-NAME))]

RecurrentNet: Forward propagating...

RecurrentNet: Prediction:

(0.014699033 0.1123531 0.005615511 0.086756654 0.32903638 0.0013546887 0.68235165 0.0021127297 0.5569747 0.37053
 04 0.02657212 0.05936804 0.066709384 0.045453567 0.1352333 0.102601826 0.005092019 0.0041175676 0.12681322 0.06572117
 0.39188597 0.4495346 0.7061425 0.037395228 0.0014276046 0.6112053 3.5331127e-4 0.058075085 0.06910719 0.07064937 0.169
 70897 0.39242262 0.0044099395 0.01582387 0.14374946 0.15425228 0.11064617 0.020329852 0.006546378 0.34938157 0.6864514
 4 0.07599342 0.037360895 0.21914004 0.3014929)

RecurrentNET: Unambiguously priming:

Syntax: Specific syntactic features not primed enough. NIL

Conceptual priming: NIL

RecurrentNet: Receiving features: [(MAJ V) (FORM FIN) ((CLASS _*OTHER-ACTION))]

RecurrentNet: Forward propagating...

RecurrentNet: Prediction:

(0.6409094 6.671678e-5 0.008282855 0.27178025 0.00601796 2.6298094e-6 0.07189822 4.755308e-4 0.8969017 0.3088039
 4.3985072e-5 0.0010595805 8.573201e-4 5.263948e-4 4.8080444e-5 0.006857598 5.403738e-4 0.00270635 4.054648e-4 0.01763
 7191 0.012824329 0.087318175 0.6491578 0.26978797 0.0011637454 0.65390503 0.002120616 0.017216874 0.9183577 0.00154854
 58 0.3039199 0.015787806 0.002412431 8.228919e-5 0.041439652 0.33712184 0.021771248 6.48168e-5 9.1088104e-6 2.803855e-
 4 0.9190739 1.032367e-4 0.301443 6.201519e-4 0.04934663)

RecurrentNET: Unambiguously priming:

Syntax: Specific syntactic features not primed enough. NIL

Conceptual priming: NIL

Trying to grab subcat 1 filler for *PERSUADED2356 satisfying: *PERSON

*PERSON already received constraint propagation from *JOHN2265.

Unifying the propagated constraint graph with the subcat 1

Propagated constraint graph:

X01[[LOC X02[[CAT X03[[HEAD X04[[MAJ X05 N]]
 [MARKING X06 UNMARKED]]
 [CONT X07[[PARA X08[[INDEX X09[[PER X10 3RD]
 [NUM X11 SNG]
 [GEND X12 MASC]
 [[DEN X13 *JOHN2265]]]
 [RESTR X14[[RELN X15 *JOHN]]]
 [CONTEXT X16[[BACKGR X17[[RELN X18 NAMING]
 [NAME X19 JOHN]
 [BEARER X09]]]]
 . [MEM X13]

The head-instance synsem to be unified into

X01[[3 X02[[LOC X03[[CAT X04[[HEAD X05[[MAJ X06 V]

```

[VFORM X07 INF]
[AUX X08 +]]
[SUBCAT X09[[1 X10[[LOC X11[[CAT X12[[HEAD X13[[MAJ X14 N]
[CASE X15 NOM]]]]]]
[2 X16[[LOC X17[[CAT X18[[HEAD X19 SATURATED]]]]
[3 X20[[LOC X21[[CAT X22[[HEAD X23 SATURATED]]]]]]]
[CONT X24[[AGENT X25]]]
[RESTR X26[[RELN X27 *ACTION]]]
[RELN X27]]]]
[2 X28[[LOC X29[[CONT X30[[PARA X31[[INDEX X25]]]
[RESTR X32[[RELN X33 *PERSON]]]
[CAT X34[[HEAD X35[[MAJ X36 N]
[CASE X37 ACC]]]]]]
[1 X38[[LOC X39[[CAT X40[[HEAD X41[[MAJ X42 N]
[CASE X43 NOM]]]]]
[CONT X44[[RESTR X45[[RELN X46 *PERSON]]]
[PARA X47[[INDEX X48 ]]]]]]
[0 X49[[LOC X50[[CAT X51[[HEAD X52[[MAJ X53 V]
[VFORM X54 INF]
[AUX X55 +]
[INV X56 -]
[PRD X57 -]]]
[MARKING X58 UNMARKED]
[SUBCAT X59[[1 X38]
[2 X28]
[3 X02]]]
[CONT X60[[AGENT X48]
[PERSUADEE X25]
[CIRCUMSTANCE X24]
[RELN X61 *PERSUADE-ACTION]]]
[MEM X62 *PERSUADED2356]]

```

Unifying propagated graph with subcat constraints respecting class inheritance subsumption...

Constraint unification successful

Resulting head instance:

(*PERSUADED2356

(INHERITS-FROM *PERSUADED)

(TYPE

(VALUE

(COMMON :INST-HEAD)))

(SYNSEM

(VALUE

(COMMON

X01[[3 X02[[LOC X03[[CONT X04[[RELN X05 *ACTION]

[RESTR X06[[RELN X05]]

[AGENT X07]]]]

[CAT X08[[SUBCAT X09[[3 X10[[LOC X11[[CAT X12[[HEAD X13 SATURATED]]]]

[2 X14[[LOC X15[[CAT X16[[HEAD X17 SATURATED]]]]

[1 X18[[LOC X19[[CAT X20[[HEAD X21[[CASE X22 NOM]
[MAJ X23 N]]]]]]]

[HEAD X24[[AUX X25 +]

[VFORM X26 INF]

[MAJ X27 V]]]]]

[2 X28[[LOC X29[[CAT X30[[HEAD X31[[CASE X32 ACC]

[MAJ X33 N]]]

[CONT X34[[RESTR X35[[RELN X36 *PERSON]]

[PARA X37[[INDEX X07]]]]]

[0 X38[[MEM X39 *PERSUADED2356]

[LOC X40[[CONT X41[[RELN X42 *PERSUADE-ACTION]

[CIRCUMSTANCE X04]

[PERSUADEE X07]

[AGENT X43[[IDEN X44 *JOHN2265]

[GEND X45 MASC]

[NUM X46 SNG]

[PER X47 3RD]]]

[CAT X48[[SUBCAT X49[[3 X02]

[2 X28]]

```

[MARKING X50 UNMARKED]
[HEAD X51[[PRD X52 -]
[INV X53 -]
[AUX X54 +]
[VFORM X55 INF]
[MAJ X56 V]]]]]

[1 X57[[MEM X44]
[LOC X58[[CONTEXT X59[[BACKGR X60[[BEARER X43]
[NAME X61 JOHN]
[RELN X62 NAMING]]]
[CONT X63[[RESTR X64[[RELN X65 *JOHN]
[PARA X66[[INDEX X43]]]
[CAT X67[[MARKING X68 UNMARKED]
[HEAD X69[[CASE X70 NOM]
[MAJ X71 N]]]]]]]]]

(S-TIME
(VALUE
(COMMON 1)))
(W-TIME
(VALUE
(COMMON 2)))

*PERSON already received constraint propagation from #MARY2384.
Not meeting complement order constraint based on obliqueness order.
*PERSON already received constraint propagation from #SANDY2524.
Not meeting complement order constraint based on obliqueness order.
Trying to grab subcat 2 filler for #PERSUADED2356 satisfying: *PERSON
*PERSON already received constraint propagation from #MARY2384.
Unifying the propagated constraint graph with the subcat 2
:
:
:
Trying to grab subcat 3 filler for #GIVE2496 satisfying: #MATTER
#MATTER already received constraint propagation from #SUS12607.
Unifying the propagated constraint graph with the subcat 3
Propagated constraint graph:
X01[[LOC X02[[CAT X03[[HEAD X04[[MAJ X05 N]
[MARKING X06 UNMARKED]]
[CONT X07[[PARA X08[[INDEX X09[[PER X10 3RD]
[GEND X11 NEUT]
[IDEN X12 *SUS12607]]]
[RESTR X13[[RELN X14 *SUS1]
[INST X09]]]]
[MEM X12]

The head-instance symsem to be unified into
X01[[3 X02[[LOC X03[[CONT X04[[RESTR X05[[RELN X06 #MATTER]
[PARA X07[[INDEX X08 []]]]
[CAT X09[[HEAD X10[[CASE X11 ACC]
[MAJ X12 N]]]]]
[1 X13[[LOC X14[[CONT X15[[RESTR X16[[RELN X17 *PERSON]
[PARA X18[[INDEX X19 []]]]
[CAT X20[[HEAD X21[[CASE X22 NOM]
[MAJ X23 N]]]]]
[0 X24[[MEM X25 #GIVE2496]
[LOC X26[[CONT X27[[THEME X08]
[GOAL X28[[IDEN X29 #SANDY2524]
[GEND X30 FEM]
[NUM X31 SNG]
[PER X32 3RD]]
[AGENT X19]
[RELN X33 #GIVE-ACTION]]
[CAT X34[[SUBCAT X35[[3 X02]
[1 X13]]
[MARKING X36 UNMARKED]
[HEAD X37[[PRD X38 -]
[INV X39 -]

```

```

[AUX X40 -]
[VFORM X41 BSE]
[MAJ X42 V]]]]]

[2 X43[[MEM X29]
  [LOC X44[[CONTEXT X45[[BACKGR X46[[BEARER X28]
    [NAME X47 SANDY]
    [RELN X48 NAMING]]]
  [CONT X49[[RESTR X50[[RELN X51 *SANDY]]
    [PARA X52[[INDEX X28]]]
  [CAT X53[[MARKING X54 UNMARKED]
    [HEAD X55[[CASE X56 ACC]
      [MAJ X57 N]]]]]]]

Unifying propagated graph with subcat constraints respecting class inheritance subsumption...
Constraint unification successful
Resulting head instance:
(*GIVE2496
  (INHERITS-FROM *GIVE)
  (TYPE
    (VALUE
      (COMMON :INST-HEAD)))
  (SYNSEM
    (VALUE
      (COMMON
X01[[1 X02[[LOC X03[[CAT X04[[HEAD X05[[MAJ X06 N]
  [CASE X07 NOM]]]
  [CONT X08[[PARA X09[[INDEX X10]]]
    [RESTR X11[[RELN X12 *PERSON]]]]]]
  [0 X13[[LOC X14[[CAT X15[[HEAD X16[[MAJ X17 V]
    [VFORM X18 BSE]
    [AUX X19 -]
    [INV X20 -]
    [PRD X21 -]]
    [MARKING X22 UNMARKED]
    [SUBCAT X23[[1 X02]]]
  [CONT X24[[RELN X25 *GIVE-ACTION]
    [AGENT X10]
    [GOAL X26[[PER X27 3RD]
      [NUM X28 SNG]
      [GENO X29 FEM]
      [IDEN X30 *SANDY2524]]
    [THEME X31[[IDEN X32 *SUSI2607]
      [GENO X33 NEUT]
      [PER X34 3RD]]]]]
    [MEM X35 *GIVE2496]]
  [2 X36[[LOC X37[[CAT X38[[HEAD X39[[MAJ X40 N]
    [CASE X41 ACC]]
    [MARKING X42 UNMARKED]]
  [CONT X43[[PARA X44[[INDEX X26]]
    [RESTR X45[[RELN X46 *SANDY]]]
  [CONTEXT X47[[BACKGR X48[[RELN X49 NAMING]
    [NAME X50 SANDY]
    [BEARER X26]]]]]
    [MEM X30]]
  [3 X51[[MEM X32]
    [LOC X52[[CONT X53[[RESTR X54[[INST X31]
      [RELN X55 *SUSI]]
      [PARA X56[[INDEX X31]]]
    [CAT X57[[MARKING X58 UNMARKED]
      [HEAD X59[[CASE X60 ACC]
        [MAJ X61 N]]]]]]]]]

(S-TIME
  (VALUE
    (COMMON 1)))
(W-TIME
  (VALUE
    (COMMON 5)))

```

case information ambiguous.

RecurrentNet: Receiving features: [(MAJ N) (FORM NORM) (CLASS _*FIRST-NAME)]

RecurrentNet: Forward propagating...

RecurrentNet: Prediction:

#(0.0154738 0.35002264 0.008896059 0.028663322 0.2347435 0.005945847 0.63691795 0.01255739 0.374973 0.47101313 0.048599258 0.04839388 0.17561457 0.24761909 0.33769128 0.17915344 0.022372289 0.016856076 0.23704979 0.08486211 0.37755772 0.34873196 0.7351534 0.041972607 0.0053865626 0.28311712 0.0035276331 0.07578456 0.0769191 0.09640902 0.14621502 0.3473997 0.011042247 0.03036274 0.18813397 0.40032905 0.2921809 0.039765105 0.02317913 0.6034165 0.30372223 0.20795058 0.024068216 0.20883459 0.6907715)

RecurrentNET: Unambiguously priming:

Syntax: Specific syntactic features not primed enough. NIL

Conceptual priming: NIL

Trying to grab subcat 3 filler for *PERSUADED2356 satisfying: *ACTION

*ACTION already received constraint propagation from *PERSUADED2356.

Not meeting complement order constraint based on obliqueness order.

*ACTION already received constraint propagation from *GIVE2496.

Not meeting complement order constraint based on obliqueness order.

Trying to grab subcat 1 filler for *TO2468 satisfying: *MATTER

Trying to grab subcat 2 filler for *TO2468 satisfying: *ACTION

*ACTION already received constraint propagation from *PERSUADED2356.

Not meeting complement order constraint based on obliqueness order.

*ACTION already received constraint propagation from *GIVE2496.

Unifying the propagated constraint graph with the subcat 2

Propagated constraint graph:

```
X01[[LOC X02[[CAT X03[[HEAD X04[[MAJ X05 V]
      [VFORM X06 BSE]
      [AUX X07 -]
      [INV X08 -]
      [PRD X09 -]]]
    [MARKING X10 UNMARKED]
    [SUBCAT X11[[[X12[[LOC X13[[CAT X14[[HEAD X15[[MAJ X16 N]
      [CASE X17 NOM]]]
      [CONT X18[[PARA X19[[INDEX X20]]]
      [RESTR X21[[RELN X22 *PERSON]]]]]]]]
    [CONT X23[[RELN X24 *GIVE-ACTION]
      [AGENT X20]
      [GOAL X25[[PER X26 3RD]
        [NUM X27 SNG]
        [GEND X28 FEM]
        [IDEN X29 *SANDY2524]]]
      [THEME X30[[IDEN X31 *SUS12607]
        [GEND X32 NEUT]
        [PER X33 3RD]]]]]
    [MEM X34 *GIVE2496]
```

The head-instance synsem to be unified into

```
X01[[2 X02[[LOC X03[[CAT X04[[SUBCAT X05[[1 X06[[LOC X07[[CONT X08[[AGENT X09]]]
      [CAT X10[[HEAD X11[[MAJ X12 N]
      [CASE X13 NOM]]]]]
      [2 X14[[LOC X15[[CAT X16[[HEAD X17 SATURATED]]]
      [3 X18[[LOC X19[[CAT X20[[HEAD X21 SATURATED]]]]]
      [HEAD X22[[MAJ X23 V]
      [VFORM X24 BSE]]]
      [CONT X25[[AGENT X09]
      [RESTR X26[[RELN X27 *ACTION]
      [RELN X27]]]]]
    [1 X28[[LOC X29[[CAT X30[[HEAD X31[[MAJ X32 N]]]
      [CONT X33[[PARA X34[[INDEX X09]
      [RESTR X35[[RELN X36 *MATTER]]]]]]]
    [0 X37[[LOC X38[[CAT X39[[HEAD X40[[MAJ X41 V]
      [VFORM X42 INF]
      [AUX X43 +]
      [INV X44 -]
      [PRD X45 -]]]
      [MARKING X46 UNMARKED]
      [SUBCAT X47[[1 X28]
```

```

[2 X02]]]
[CONT X25]]
[MEM X48 *T02468]]
Unifying propagated graph with subcat constraints respecting class inheritance subsumption...
Constraint unification successfull
Resulting head instance:
(*T02468
(INHERITS-FROM *TO)
(TYPE
(VALUE
(COMMON :INST-HEAD)))
(SYNSEM
(VALUE
(COMMON
X01[[1 X02[[LOC X03[[CONT X04[[RESTR X05[[RELN X06 *MATTER]]
[PARA X07[[INDEX X08]]]]
[CAT X09[[HEAD X10[[MAJ X11 N]]]]]
[0 X12[[MEM X13 *T02468]
[LOC X14[[CONT X15[[RESTR X16[[RELN X17 *GIVE-ACTION]]
[THEME X18[[PER X19 3RD]
[GENO X20 NEUT]
[IDEN X21 *SUS12607]]
[GOAL X22[[IDEN X23 *SANDY2524]
[GENO X24 FEM]
[NLM X25 SNG]
[PER X26 3RD]]
[AGENT X08]
[RELN X17]]
[CAT X27[[SUBCAT X28[[1 X02]]
[MARKING X29 UNMARKED]
[HEAD X30[[PRD X31 -]
[INV X32 -]
[AUX X33 +]
[VFORM X34 IHF]
[MAJ X35 V]]]]]]
[2 X36[[MEM X37 *GIVE2496]
[LOC X38[[CONT X15]
[CAT X39[[SUBCAT X40[[2 X41[[LOC X42[[CAT X43[[HEAD X44 SATURATED]]]]
[3 X45[[LOC X46[[CAT X47[[HEAD X48 SATURATED]]]]
[1 X49[[LOC X50[[CONT X51[[AGENT X08]
[RESTR X52[[RELN X53 *PERSON]]
[PARA X54[[INDEX X08]]]
[CAT X55[[HEAD X56[[CASE X57 NOM]
[MAJ X58 N]]]]]]]
[MARKING X59 UNMARKED]
[HEAD X60[[PRD X61 -]
[INV X62 -]
[AUX X63 -]
[VFORM X64 BSE]
[MAJ X65 V]]]]]]))
(S-TIME
(VALUE
(COMMON 1)))
(W-TIME
(VALUE
(COMMON 4)))
:
:
:
Unifying propagated graph with subcat constraints respecting class inheritance subsumption...
Constraint unification successfull
Resulting head instance:
(*PERSUADED2356
(INHERITS-FROM *PERSUADED)
(TYPE
(VALUE

```

```

(COMMON :INST-HEAD))
(SYNSEM
(VALUE
(COMMON
X01[[0 X02[[MEM X03 *PERSUADED2356]
      [LOC X04[[CONT X05[[RELN X06 *PERSUADE-ACTION]
            [CIRCUMSTANCE X07[[RELN X08 *GIVE-ACTION]
                  [AGENT X09[[PER X10 3RD]
                        [NUM X11 SNG]
                        [GEND X12 FEM]
                        [IDEN X13 *MARY2384]]
                  [GOAL X14[[PER X15 3RD]
                        [NUM X16 SNG]
                        [GEND X17 FEM]
                        [IDEN X18 *SANDY2524]]
                  [THEME X19[[IDEN X20 *SUS12607]
                        [GEND X21 NEUT]
                        [PER X22 3RD]]
                  [RESTR X23[[RELN X08]]]
            [PERSUADEE X09]
            [AGENT X24[[IDEN X25 *JOHN2265]
                  [GEND X26 MASC]
                  [NUM X27 SNG]
                  [PER X28 3RD]]]
      [CAT X29[[SUBCAT X30[NIL]]
            [MARKING X31 UNMARKED]
            [HEAD X32[[PRD X33 -]
                  [INV X34 -]
                  [AUX X35 +]
                  [VFORM X36 INF]
                  [MAJ X37 V]]]]]
      [1 X38[[MEM X25]
            [LOC X39[[CONTEXT X40[[BACKGR X41[[BEARER X24]
                  [NAME X42 JOHN]
                  [RELN X43 NAMING]]]
            [CONT X44[[RESTR X45[[RELN X46 *JOHN]
                  [PARA X47[[INDEX X24]]]
            [CAT X48[[MARKING X49 UNMARKED]
                  [HEAD X50[[CASE X51 NOM]
                        [MAJ X52 N]]]]]
      [2 X53[[LOC X54[[CAT X55[[HEAD X56[[MAJ X57 N]
            [CASE X58 ACC]
            [MARKING X59 UNMARKED]
            [CONT X60[[PARA X61[[INDEX X09]
                  [RESTR X62[[RELN X63 *MARY]]]
            [CONTEXT X64[[BACKGR X65[[RELN X66 NAMING]
                  [NAME X67 MARY]
                  [BEARER X09]]]
            [MEM X13]]
      [3 X68[[LOC X69[[CAT X70[[HEAD X71[[MAJ X72 V]
            [VFORM X73 INF]
            [AUX X74 +]
            [INV X75 -]
            [PRD X76 -]]
            [MARKING X77 UNMARKED]
            [SUBCAT X78[[2 X79[[LOC X80[[CAT X81[[HEAD X82 SATURATED]]]]
                  [3 X83[[LOC X84[[CAT X85[[HEAD X86 SATURATED]]]]
                  [1 X87[[LOC X88[[CAT X89[[HEAD X90[[CASE X91 NOM]
                        [MAJ X92 N]]]
                  [CONT X93[[PARA X94[[INDEX X09]
                        [RESTR X95[[RELN X96 *MATTER]]]]]]]]]
            [CONT X07]]
            [MEM X97 *T02468]])))]
(S-TIME
(VALUE
(COMMON 1)))

```

```

(W-TIME
(VALUE
(COMMON 2))))
Global massive parallelism invoked...Using 1 processor
Global massive parallelism invoked...Using 1 processor
Specific syntactic features not primed enough.
RecurrentNet: Receiving features: [NIL ((CLASS _*END-PUNCTUATION))]
RecurrentNet: Forward propagating...
RecurrentNet: Prediction:
  #[0.023348229 0.036268964 0.0069902344 0.08897273 0.07429021 0.0015437683 0.92383903 0.016427 0.3972626 0.2532076
8 0.0077487146 0.09314012 0.088530324 0.026151301 0.074738614 0.22995259 0.005168746 0.0028443348 0.16909087 0.0108955
73 0.94808227 0.723022 0.7163689 0.023458632 3.631415e-4 0.4608943 4.364973e-4 0.035874907 0.17080069 0.06231131 0.112
641536 0.64059454 0.0068304082 0.0028339091 0.106612 0.54372585 0.18167657 0.010155719 7.47683e-4 0.32288808 0.6085463
0.09237444 0.06486768 0.14129014 0.5319556]
RecurrentNET: Unambiguously priming:
  Syntax: Specific syntactic features not primed enough. NIL
  Conceptual priming: _*OBJECT

```

Resulting global memory state with 1 node(s) in the Decaying Layer:

```

(*PERSUADED2356
(INHERITS-FROM *PERSUADED)
(TYPE
(VALUE
(COMMON :INST-HEAD)))
(SYNSEM
(VALUE
(COMMON
X01[[0 X02[[MEM X03 *PERSUADED2356]
  [LOC X04[[CONT X05[[[RELN X06 *PERSUADE-ACTION]
    [CIRCUMSTANCE X07[[[RELN X08 *GIVE-ACTION]
      [AGENT X09[[[PER X10 3RD]
        [NUM X11 SNG]
        [GEND X12 FEM]
        [IDEN X13 *MARY2384]]
      [GOAL X14[[[PER X15 3RD]
        [NUM X16 SNG]
        [GEND X17 FEM]
        [IDEN X18 *SANDY2524]]
      [THEME X19[[[IDEN X20 *SUS12607]
        [GEND X21 NEUT]
        [PER X22 3RD]]
      [RESTR X23[[[RELN X08]]]
        [PERSUADEE X09]
        [AGENT X24[[[IDEN X25 *JOHN2265]
          [GEND X26 MASC]
          [NUM X27 SNG]
          [PER X28 3RD]]]
        [CAT X29[[[SUBCAT X30[[NIL]]
          [MARKING X31 UNMARKED]
          [HEAD X32[[[PRD X33 -]
            [INV X34 -]
            [AUX X35 +]
            [VFORM X36 INF]
            [MAJ X37 V]]]]]]
        [1 X38[[[MEM X25]
          [LOC X39[[[CONTEXT X40[[[BACKGR X41[[[BEARER X24]
            [NAME X42 JOHN]
            [RELN X43 NAMING]]]
          [CONT X44[[[RESTR X45[[[RELN X46 *JOHN]
            [PARA X47[[[INDEX X24]]]
          [CAT X48[[[MARKING X49 UNMARKED]
            [HEAD X50[[[CASE X51 NOM]
              [MAJ X52 N]]]]]]
        [2 X53[[[LOC X54[[[CAT X55[[[HEAD X56[[[MAJ X57 N]

```

```

[CASE X58 ACC]]
[MARKING X59 UNMARKED]]
[CONT X60[[PARA X61[[INDEX X09]]
[RESTR X62[[RELN X63 *MARY]]]
[CONTEXT X64[[BACKGR X65[[RELN X66 NAMING]
[NAME X67 MARY]
[BEARER X09]]]]
[MEM X13]]
[3 X68[[LOC X69[[CAT X70[[HEAD X71[[MAJ X72 V]
[VFORM X73 INF]
[AUX X74 +]
[INV X75 -]
[PRD X76 -]]
[MARKING X77 UNMARKED]
[SUBCAT X78[[2 X79[[LOC X80[[CAT X81[[HEAD X82 SATURATED]]]]
:: SUBCAT moved for print [3 X83[[LOC X84[[CAT X85[[HEAD X86 SATURATED]]]]
[1 X87[[LOC X88[[CAT X89[[HEAD X90[[CASE X91 NOM]
[MAJ X92 N]]]
[CONT X93[[PARA X94[[INDEX X09]]
[RESTR X95[[RELN X96 *MATTER]]]]]]]]
[CONT X07]]
[MEM X97 *T02468]])))]
(S-TIME
(VALUE
(COMMON 1)))
(W-TIME
(VALUE
(COMMON 2))))
NIL
<Initial lwp>

```

Appendix III:

Sample recurrent net run (without constraint propagation network):

Sentence: 5:

Input Word: I

#(1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0)

Decoding..

Syntax : ((MAJ N) (FORM NORM) (CASE NOM))

Semantics : *_IDENTITY

Forward propagating...

Prediction 1:

#(0.08065432 0.44271022 3.304273e-6 7.283204e-5 0.5740456 0.03322048 0.12289865 0.40194097 0.010169014 0.0 3.5834
056e-5 9.4155085e-6 1.4496867e-5 0.35719004 0.0 3.266633e-5 2.6477635e-5 2.8367454e-5 2.0770529e-5 1.3217542e-5 1.0105
801e-6 0.27352263 0.9208149 2.1918598e-5 0.02234797 1.9793946e-4 0.15002659 0.3563637 0.29727048 0.0 0.96435016 0.3886
9813 1.8852015e-5 4.3598768e-6 0.06382471 0.042883247 0.0031745322 0.06726367 4.6445906e-5 2.3818132e-5 0.0077976245 0
.7941675 0.0 1.8159598e-5 1.844641e-5)

Prediction 2:

#(0.001030652 0.0987835 5.319159e-6 0.0 0.0 0.0017127303 0.59903026 0.54681766 0.99999595 0.0 0.0 0.0 0.0 4.63497
88e-4 0.0 7.8377664e-6 5.7669246e-6 3.6822745e-6 1.9240405e-6 5.413526e-7 0.0 0.8725633 0.39791763 2.3441981e-6 0.0014
452058 0.0 5.2589756e-4 0.0 0.97264266 0.0 0.97116876 0.88693845 4.5521352e-6 0.0 0.0 0.9949084 0.021591425 0.0 3.7000
075e-5 6.0775995e-6 0.9779734 0.10828607 0.0 4.3352157e-6 2.0350656e-6)

Unambiguously priming:

Syntax: Specific syntactic features not primed enough. NIL

Semantics: NIL

Unambiguously priming:

Syntax: Specific syntactic features not primed enough. NIL

Semantics: NIL

Input Word: WOULD-LIKE

#(0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0)

Decoding..

Syntax : ((MAJ V) (FORM BSE))

Syntax: ((MAJ D) (FORM THE))
Semantics: *_INDEFINITE-REFERENCE
Input Word: FOR
#(0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0)
Decoding..
Syntax: ((MAJ P) (FORM FOR))
Semantics: *_IDENTITY
Forward propagating...
Prediction 1:
#(0.025430072 0.0800034 9.0245414e-7 1.2959078e-4 0.97995114 0.0010964621 0.66273415 0.026702615 0.0038762174 2.3
270592e-5 2.2359237e-4 0.0020364996 1.0972683e-5 0.061779097 0.0 6.644864e-5 4.938321e-5 4.6108612e-5 2.6802772e-5 8.0
954665e-5 6.7984146e-7 0.47155198 0.37317896 5.250912e-5 1.1134641e-5 0.29229137 0.010076122 0.9570185 0.0026000412 0.
0 0.090170965 0.051607817 5.170351e-5 3.12883e-4 0.0 1.4693663e-4 2.171143e-6 0.948931 0.0024116694 6.496662e-5 5.7048
875e-6 0.9987796 3.3912218e-5 4.6390715e-5 1.6625652e-5)
Prediction 2:
#(0.99587613 2.6843035e-4 1.8449661e-6 4.1015065e-4 0.001769016 3.5763386e-5 0.010490522 2.5873953e-4 0.7233231 0
.0 4.1577905e-7 0.044339884 0.0 0.0 1.4707312e-6 4.938835e-6 3.869858e-6 5.351776e-6 1.2117688e-5 4.0337974e-6 0.00654
8499 0.96414596 0.137382 5.570349e-6 0.0 0.10854193 0.019531576 1.1240703e-5 0.2590055 4.126145e-6 0.53806704 0.420100
2 3.4378007e-6 0.0 0.059565436 0.0020149 0.004508333 0.0010418281 0.34381053 4.7415084e-6 0.0041542156 0.092113815 0.0
6.557656e-6 5.428172e-7)
Unambiguously priming:
Syntax: Form information ambiguous. ((MAJ D))
Semantics: NIL
Unambiguously priming:
Syntax: Form unconstrained case information ambiguous. ((MAJ N))
Semantics: *_PHENOMENON
Input Word: THE
#(0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0)
Decoding..
Syntax: ((MAJ D) (FORM THE))
Semantics: *_INDEFINITE-REFERENCE
Forward propagating...
Prediction 1:
#(0.9994159 1.5150331e-5 7.1496844e-7 0.0 6.0090256e-4 2.6247492e-4 0.004174005 4.72376e-4 0.99733955 0.0 0.0 7.4
60583e-7 0.0 4.9344003e-7 0.0 1.08347155e-6 7.7414596e-7 1.6305378e-6 2.6482144e-6 2.8666532e-6 5.173069e-5 0.7268867
0.0012903553 1.5074804e-6 0.0 2.3997816e-4 0.043803982 0.004405924 0.7806072 0.0 0.9992938 0.0053379796 6.4787954e-7 0
.0 2.116903e-4 0.0039119534 0.10209093 4.0302816e-4 1.6973701e-5 1.2386159e-6 0.09763454 0.01080258 0.0 1.3040462e-6 3
.4048295e-7)
Prediction 2:
#(0.0018263549 4.8998085e-5 1.6378406e-6 0.0 8.499588e-5 0.117409 0.9763714 0.001119116 0.10569345 0.9337093 3.47
21392e-5 4.8418087e-6 6.35503e-7 9.253743e-4 1.9667076e-6 1.8688692e-5 8.768017e-6 3.6405584e-6 1.874708e-5 2.1647196e
-5 0.0 2.862651e-6 0.9998745 1.3526556e-5 0.0 3.3506823e-6 0.10532789 1.2680486e-5 0.64822525 0.0 3.9788792e-6 0.99874
83 1.5393664e-5 1.1082761e-4 0.014746789 0.8960314 1.1186282e-5 0.006877368 0.0 1.3740902e-5 0.025940238 0.15665819 0.
38879758 1.1645927e-5 3.979821e-7)
Unambiguously priming:
Syntax: case information ambiguous. ((MAJ N) (FORM NORM))
Semantics: NIL
Unambiguously priming:
Syntax: Specific syntactic features not primed enough. NIL
Semantics: NIL
Input Word: CONFERENCE
#(1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0)
Decoding.. case information ambiguous.
Syntax: ((MAJ N) (FORM NORM))
Semantics: *_CONFERENCE
Forward propagating...
:
:
Input Word: *PERIOD*
:
: