International Workshop on Fundamental Research for the Future Generation of Natural Language Processing (FGNLP)

Proceedings of the Workshop

23 & 24 July 1991
Kyoto International Community House
Kyoto, Japan

ATR Interpreting Telephony
Research Laboratories

In proceedings of International Workshop on Fundamental Research for the Future generation Natural Language Processing, 1991

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

A Massively-Parallel Architecture for Symbolic and Subsymbolic Interactions

Hideto Tomabechi
Carnegie Mellon University
109 EDSH, Pittsburgh, PA 15213-3890, USA,
tomabech@cs.cmu.edu.
and
ATR Interpreting Telephony Research Laboratories
Seika-cho, Soraku-gun, Kyoto, JAPAN
tomabech

Abstract

This paper describes an architecture for symbolic and subsymbolic interactions during machine processing of massively-parallel cognitive activities. 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 smooth *a posteriori* learning to the symbolic system and focused guided learning as well as strong constraints during recognition to the nerual network. As the result, the architecture provides the ability to handle strict and structured symbolic constraints during recognition while attaining a smooth contextual prediction ability applied with the least rigidity and learning of input regularities from actual samples. The domain discussed in this paper is natural language understanding for demonstration purposes; however the architecture is expected to show equal strength in other modal channels such as visual inputs.

1. Introduction

MONA-LISA stands for Multimodal Ontological Neural Architecture for Linguistic Interactions and Scalable Adaptations. 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 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 contextually sensitive tasks 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 is too limited in contrast to the actual variety of cognitive inputs to perform any meaningful inferencing for learning[2]. Application of stored knowledge to the input is rigid in the symbolic models because symbolic matching of stored knowledge has to be exact (as examplified 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.

---

[1] More recently, [Tomabechi, *et al.*, 1988], [Kitano, 1989], and [Tomabechi and Levin, 1989].

[2] For 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.

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

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. The architecture is designed to realize integration of subsymbolic PDP network with the symbolic massively-parallel network and therefore, the TDNN which is a speech recognition neural network is not the only kind of neural net input modal channel assumed in this architecture. For example, visual recognition neural network can equally be well integrated since the architecture is designed to integrate the neural net processing itself. The actual data processed in the networks may be different depending on whether they are sonic or visual but there is no fundamental difference between the actual neural networks used in processing them. Thus, although this paper focuses its discussion on integration of symbolic and subsymbolic information in terms of linguistic constraint processing, the choice of the focus should be regarded as

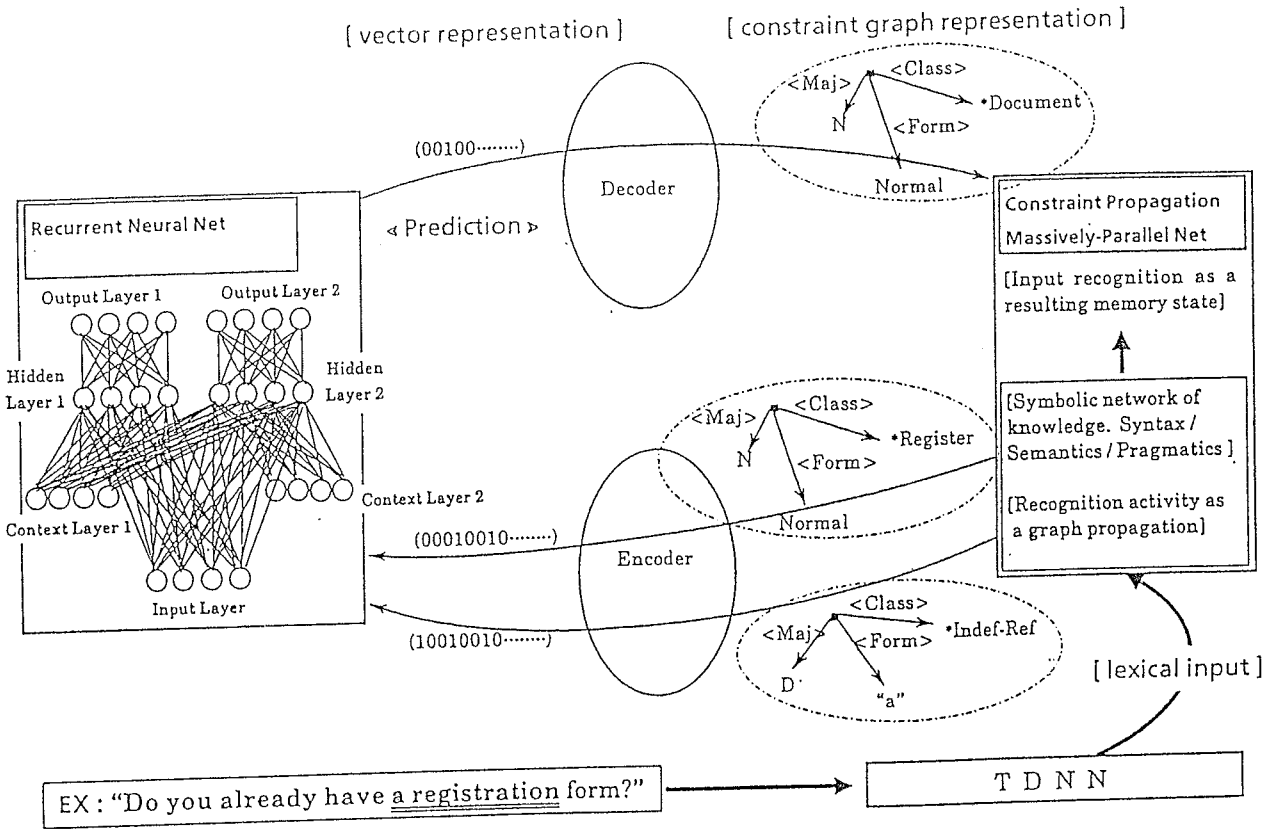

Figure 1: MONA-LISA Architecture

pragmatic. The choice is mainly due to the clarity of presentation by actually demonstrating that existing symbolic models of linguistic cognitive processing can be fully captured in our architecture while maintaining the *a posteriori* learning of the PDP architecture. Once we accumulate enough theoritical formulations of the visual and other cognitive activities at abstract level (similar to the way linguistic activities have been generalized), we hope to demonstrate the architecture's strength in these areas as well.

In the MONA-LISA architeccture, 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 singal 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) Most importantly, the variety of constraints captured in the architecture is not limited to be of that of symbolic and subsymbolic nature. Constraints with very different nature can potentially coexist under this architecture. These constraints may vary in terms of their level of abstraction, aprioricity, compositionality and monotonicity, to name a few[3]. Of course this does not exhaust the variety of constraints potentially capturable in our architeture; however, note that no previous architecture provided an uniform representation and a singular processing of any of these constraints dichotomies. In fact most of the systems simply ignored these problems. (For example, most natural language systems are purely compositional. Also very few inference systems incorporate subsymbolic information, let alone nonmonotonic nones.)

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 graphs[4] and they may point to any location in the network and may contain complex paths including convergent arcs and cycles. 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.,

[3]By different levels of abstraction I mean the differences in the nature of information from the acoustic spectral signal level of speech (i.e. physical level) up to the abstract conceptual knowledge that may not have counterparts in the real physical world.

By level of aprioricity I mean the nature of constraints in terms of (the degree of) the participation of experience in acquiring it, i.e., the epistemological questions about the nature of knowledge captured in our representation especially with regard to the *a priori* / *a posteriori* distinction (i.e., [Kant, 1781]). In natural language processing, some constraints are *a priori* provided to the system while others are acquired through actual recognition of utterances.

By compositional I mean the traditional Fregean/Montagovian notion of compositionality that *meaning of the whole is a function of combining the meaning of the parts*. Many pragmatic constraints in natural language involve noncompositional constraint processing. Also, some constraints are configurational and others are not.

By monotonic I mean that the information is only added as time passes and never gets subtracted. Also that the information once added is never modified as time passes.

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

cannot be complex structures to satify 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 cycles, 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.

Below is the sample lexicon from our current MONA-LISA implementation (simulated massive-parallelism on a 16 CPU shared-memory parallel machine) 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. Note that our choice of HPSG is a purely pragmatic one. The theoretical basis of linguistic formulation can be any other theories that are trivially representable in the graph notations (such as GPSG [Gazdar, *et al.*, 1985] and LFG [Bresnan and Kaplan, 1982]). In fact, the constraints captured in the architecture is not assumed to be limited to linguistic ones. Let us reiterate that we use linguistic information (and HPSG formulation in particular) for the purpose of the clear demonstration of the architecture at work and should not be regarded as the only intended area of actual applications.

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 internally. 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))))
```

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:

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

can easily specify the control constraints lexically in the network. With an addition of a specification for the intermediate subject control VP head *to* 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 mechanism[5]. 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 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 [Tomabechi, *et al.*, 1988] and [Kitano, 1989] for activities of those nodes.)

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

```
function sentential-recognize (input-stream)
```

[5]We use graph-unification as currently desirable scheme of cheking graph-based constraints, but other method may replace unification in the future implementations.

```
 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-hypotesis)
 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*).

When each word is input, an instance of 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

GCPN result state (After the recognition of "John Persuaded Mary to give Sandy Susi")

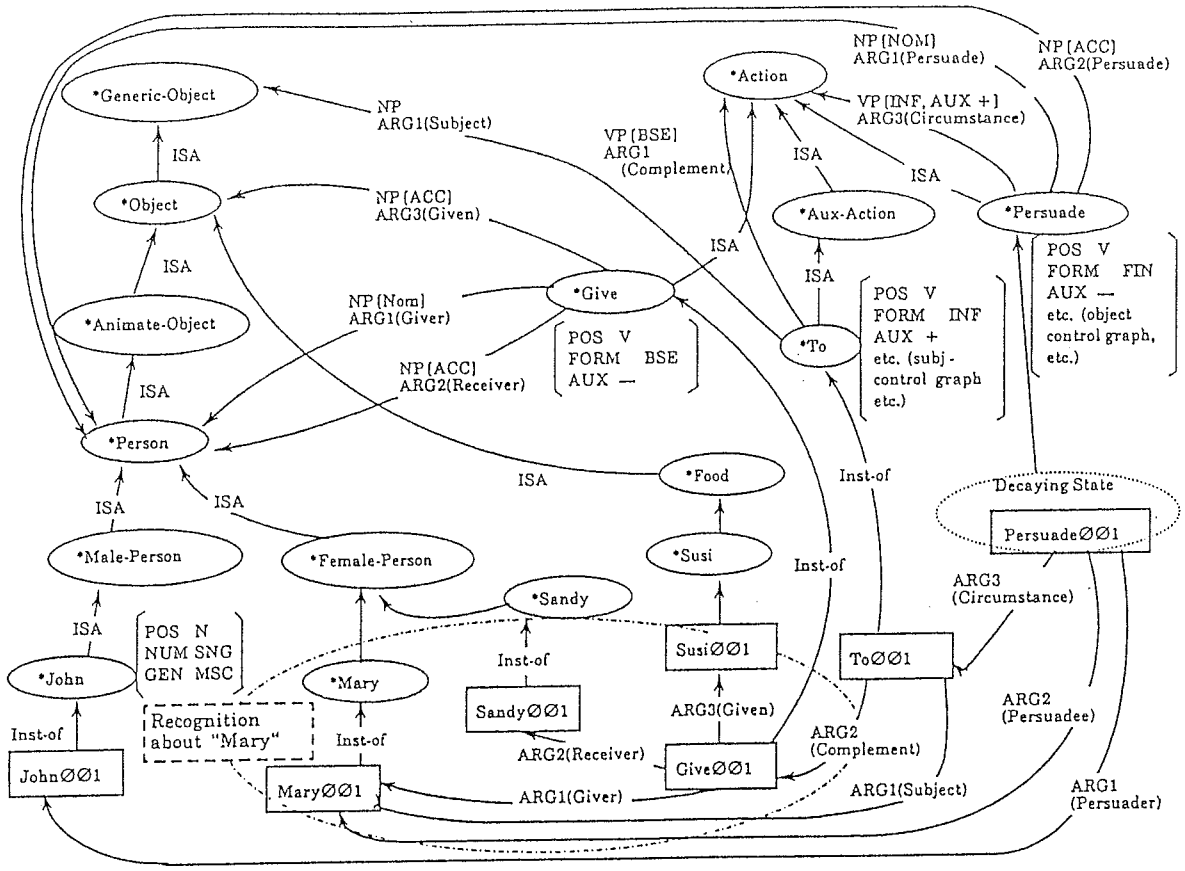

Figure 2: A Portion of GCPN representing *control* relation

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 according 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. 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. We will be reviewing our scheme for dealing with this problem in the second subsection.

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).

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 category[7] (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

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

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

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

pronoun *it* of extraposition[9] and pseudocleft[10] 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 encodes 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], 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 below[14]:

#(0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 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, singular[16].

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 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

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

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

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

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

[13] For 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.*

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

[15] This 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]).

[16] Gender 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.

		:	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); 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 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 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 features[17] 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. In one experiemt, when the current sentence was *I would like*[18] *to register for the conference*, when *to* is input the next inheritance class concept **register* was 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 many 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 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. 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).

6. Discussion:

We have seen in the previous sections that our model is capable of taking advantage of constraints that are postulated in modern cognitive theories by fully supporting arbitrary graphs to be used as constraints in the network. In fact our current sample implementation for natural language uses HPSG which is represented by path equations (which are automatically converted to a directed subsumption-ordering graphs[19]) that are no different from the grammar that we are using for the

[17] We 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.

[19] Subsumption ordering is an assumed condition for the graph-based representation in the unification-based grammar formalisms. Simply put, the information content represented by the directed graph node which is closer to the

unification-based parsers. Although not discussed in this paper, treatment of phenomena such as long-distance dependency is straightforward in our model (no different from the way it is handled in the unification-based grammar systems). This shows the exisiting theories of cognition can directly be represented and manipulated in our architecture. This is a clear contrast to the past models of massively-parallel processing that symbolic and principled representation of constraints were difficult due to the distributed nature of processing. Also note that the architecture supports integration of the neural net PDP processing. This makes the symbolic constraints with the full capacity to capture cognitive theoretical postulations to be cooperating with the fully distributed representation and learning based upon the actual input. In other words, the *a priori* provided knowledge about the world (i.e., symbolic constraints) can help attain focus of attention during the *a posteriori* learning in the neural network based upon the actual input to the system. This cooperation is both ways as we have seen in the capacity of the recurrent network to provide strong contextual predictions to the symbolic constraint propagation network that can be used by the symbolic network to help its disambiguation and other inferential capacities. A practical advantage of this architecture aside from the theoretical ones is that the capacity to perform learning based upon the actual input while taking advantage of the symbolic existing knowledge would mean that the system will perform better as we use the system more. It is because the subsymbolic learning can be performed during the actual usage and the architecture can fine-tune its performance based upon the particular usage patterns of the end users. In other words, *the more we use, the better the system performs.*

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 symbolic and configurational constraints either inevitably involved introduction of arbitrary functional applications to handle configurationality (such as [Tomabechi and Levin, 1989]) or required a separate control structure to handle configurational constraints (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.

7. 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. For example, we could design a visual neural network connected to GCPN so that whenever the visual neural network recognizes particular input it can enhance activity in the other modal channels. The recognition from different modal channels could prime the same instance in the memory network so that the activities in different modal areas can enhance or inhibit constraint processing activities based upon activations in other areas. Typically, when a visual information can help disambiguation activity in the linguistic activity, the architecture will support a very strong pragmatic information processing in the GCPN. 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. The activation in the GCPN is only propagated upward in the inheritance hierarchy and never horizontally. Since the increase in the constraints takes place horizontally, the complexity increase can be countered by an increase in the number of processing units.

---

root of the graph subsumes the information content of the nodes further from the root in the same graph.

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 (or the external world with the mental world).

Implementation

The MONA-LISA architecture assumes massively-parallel processing with a mixture of varying levels of granularity of parallelism from fine to medium grain. For example, constraint processing of graph unification is fine to medium grain; however, processing of the recurrent neural net and the speech recognition neural-net are of fine grain. Also, since the constraint graphs are allowed to point to any location in the memory-network, the hardware needs to support the parallel processes to access any location in a shared memory with least overhead. Currently no massively-parallel hardware designs (such as CM-2 [Thinking Machines Corp., 1989], SNAP [Lee and Moldovan, 1990], IXM2 [Higuchi, *et al*, 1991]) supports this kind of massively-parallel processing in which processing granularity is strictly limited to fine grain[20] and memory is fully distributed. 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. Our current implementation may set a good precedent for demonstrating a simulated massive-parallelism of hybrid granularity using a closely-coupled shared memory parallel architecture.[21] The main technological bottleneck in actually designing a massively-parallel hardware that directly supports MONA-LISA architecture lies in the solution of so called Hillis bottleneck. This is the problem of the limitation of capacity to send large amount of data quickly between distributed processing units in the massively-parallel architecture. Due to this problem, for the next decade or so, MONA-LISA architecture may be only suitable to be run based upon a simulated massive-parallelism on a limited parallelism shared memory parallel hardware (for example upto 256 CPUs or so).

---

[20] Except for the transputer support of IXM2

[21] The problem of varying granularity of parallelism, depending on different types of constraints in the network is countered in my current implementation by inserting intermediate light-weight processes to handle different levels of parallelism. Namely, I have divided 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 units may be dedicated to the scheduling of *lwps*.

Acknowledgments

I would like to thank Masaru Tomita, Jaime Carbonell, David Evans, Alex Waibel, and other members of the Carnegie Mellon community for their comments on the various parts of this research. I would also like to thank Margalit Zabludowski, Hiroaki Saito, Hiroaki Kitano, Tetsuya Higuchi for comments on the early versions of this paper. Personal discussions with Toru Uzawa, Koiti Hasida, Satoshi Tojo, and Kiyoshi Kogure contributed greately to the formulation of the essential notions contained in the architecture. Special thanks are due to Akira Kurematsu, Hitoshi Iida and the members of the ATR Interpreting Telephony Research Laboratories for their support during my two visits to ATR as a visiting research scientist.

References

[Berg, 1987] Berg, G. "A Parallel Natural Language Processing Architecture with Distributed Control". In *Proceedings of the CogSci-87*, 1987.

[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.

[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

[Gazdar, *et al.*, 1985] Gazdar, G., Pullum, G., and Sag, I. *Generalized Phrase Structure Grammar.* Harvard University Press, 1985.

[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.

[Higuchi, *et al*, 1991] Higuchi, T., T. Furuya, K. Hanada, N. Takahashi, H. Nishiyama, and A. Kokubu "IXM2: A Parallel Associative Processor". In *Proceedings of The 18th International Symposium on Computer Architecture*, 1991.

[Higuchi, *et al*, 1991] Higuchi, T., H. Kitano, K. Hanada, T. Furuya, N. Takahashi, and A. Kokubu "IXM2: A Parallel Associative Processor". In *Proceedings of AAAI-91.*

[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.

[Kant, 1781] Kant, I. *Critique of Pure Reason*, trans. J. Meiklejohn, Prometheus Books, Buffalo, New York, 1990.

[Kitano, 1989] Kitano, H. 'Massively Parallel Model of Simultaneous Interpretation: The *Phi*DMDIALOG System,' Technical Report CMU-CMT-89-116, 1989.

[Kitano, Tomabechi, and Levin, 1988] Kitano, H., Tomabechi, H., and Levin, L. "Ambiguity Resolution in the *Dm-Trans Plus*". In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, 1988,

[Lee and Moldovan, 1990] Lee, W. and D. Moldovan "The Design of a Marker Passing Architecture for Knowledge Processing". In *Proceedings of AAAI-90*, 1990.

[Martin, 1989] Martin, C. "Case-based Parsing" in Riesbeck, C. and Schank, R., eds., *Inside Case-based Reasoning*, Lawence 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 empricism". 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. Lenhert G. and Ringle M. Lawrence Erlbaum, 1982.

[Thinking Machines Corp., 1989] Thinking Machines Corp. *Model CM-2 Technical Summary*, Technical Report TR89-1, 1989.

[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 and Kitano, 1989] Tomabechi, H., and Kitano, H. 'Beyond PDP: the Frequency Modulation Neural Network Architecture'. In *Proceedings of the IJCAI-89*, 1989.

[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).