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

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, apriority, compositionality and monotonicity, to name a few[8]. Of course this does not exhaust the variety of constraints potentially capturable in our architecture; however, note that no previous architecture provided an uniform representation and a singular processing of any of these constraints dichotomically viewed in 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 ones.)

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[5] 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 extended VP is coreferential with the subject of the embedded VP (as in [John persuaded Mary to eat sushi], using the marker-passing scheme (such as in [Toma-bechi and Levin, 1989], and [Kitano, 1989]), we have to explicitly 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.,

[8]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.

[9]By level of apriority 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.

[10]By compositional I mean the traditional Fregan/Montague 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.

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

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

-53-

(== inf) (== +) (<l loc cat subcat 1 loc cat head> == [[maj n] [case nom]]) (<l loc cat subcat 2 loc cat head> == saturated) ;;; must not unify (<l loc cat subcat 3 loc cat head> == saturated) ;;; must not unify (<0 loc cont circumstance> == ❤️ loc cont>) (<0 loc cont reln> == PERSUADE-ACTION) (❤️ loc cont restr reln> == ❤️ loc cont reln>) (❤️ loc cont restr reln> == *action)))

Thus the two equations:

((0 loc cont persuadee) == (2 loc cont para index)) ((0 loc cont persuadee) == (0 loc cont circumstance agent)) can easily specify the control constraints lexically in the network. With an addition of a specification for the intermedidate 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[2]. 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)

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