FROM J. FISCHER ET AL., *SPECOM'98, INT. WORKSHOP SPEECH AND COMPUTER*, ST.-PETERSBURG, 1998, (PP. 85–90).

A REAL–TIME AND ANY–TIME APPROACH FOR A DIALOG SYSTEM

J. Fischer, H. Niemann, E. Noeth
Universität Erlangen–Nürnberg,
Lehrstuhl für Mustererkennung (Informatik 5)
Martensstraße 3, D–91058 Erlangen, Germany
email: {fischerj,niemann,noeth}@informatik.uni-erlangen.de
Tel.: +49/9131/8527799 Fax: +49/9131/303811

ABSTRACT

In this paper we present a knowledge based approach to a speech understanding and dialog system which possesses any–time and real–time capabilities. Knowledge is represented by means of a semantic–network formalism; the concept–centered knowledge base is automatically compiled into a fine–grained network which allows an efficient exploitation of parallelism. The interpretation problem is defined as a combinatorial optimization problem and solved by means of iterative optimization methods. By using iterative methods, any–time capabilities are provided since after each iteration step a (sub–)optimal solution is always available. At the moment, the real–time factor for interpreting an initial user's utterance is 0.7.

1 INTRODUCTION

Automatic recognition of complex patterns, specifically motion sequences and spontaneous speech, become increasingly relevant for a great number of applications, such as service robots to aid handicapped individuals, multi–modal telecooperation tasks, etc. Real–time as well as any–time capabilities are indispensable for such systems to be used in real–world applications. The combination of parallel and iterative processing seems to be a promising means to achieve these capabilities.

A variety of parallel algorithms for data–driven processing have been developed, especially in image processing [12]. A parallel approach to speech recognition can be found for example in [11]. In contrast, parallel symbolic processing is much less investigated, although some major problems of the field, like e.g. parallel knowledge representation, are discussed in the literature [1] .

In [3] an approach to knowledge based pattern understanding which is based on iterative optimization methods and which allows an efficient parallel computation was developed and successfully tested on a small image understanding task. Current research concentrates on applying this control algorithm to a real–world speech understanding problem, using as a scenario a dialog system which is able to answer queries about the German train timetable.

The following section describes the knowledge representation that we use. Section 3 gives a short overview of the dialog system. In Section 4 we present the iterative control algorithm. Results on the performance of the system are reported in Section 5. We conclude the paper with some final remarks and an outlook to future research in Section 6.

[1]This work was partially supported by the Real World Computing Partnership (RWCP).

2 KNOWLEDGE REPRESENTATION

An *interpretation* $\mathcal{B}$ of a *pattern* $\boldsymbol{f}$ is computed using an internal *knowledge model* $\mathcal{M}$ and an *initial description* $\mathcal{A}$ of $\boldsymbol{f}$. For the representation of $\mathcal{M}$ we use the semantic network formalism ERNEST [10] which provides the following types of network *nodes*:

- **Concepts** representing abstract terms, events, tasks, actions, etc.;
- **Instances** representing actual realizations of concepts in the sensor data;
- **Modified Concepts** representing restrictions arising from intermediate results (i.e, the interpretation of part of the sensor data leads to restrictions on the interpretation of the remaining sensor data).

There are also the following network *links* in ERNEST:

- **Part Links** linking concepts and the parts (which are also represented as concepts) they consist of;
- **Concrete Links** linking concepts on different *levels of abstraction*;
- **Specialization Links** establishing inheritance relations from general concepts to more specific ones.

The task–specific knowledge is thus represented by $\mathcal{M} = \langle C \rangle$, which is a network of concepts linked to each other by the various types of links.

The main components of a concept C are its *parts* P and its *concretes* K. Figure 1 shows an excerpt of a semantic network representing linguistic knowledge about the destination of a trip. One can see that, for example, a Preposition (SY_PREP) is part of a Prepositional Phrase (SY_PP), and a Word Hypothesis (H_WHYP) is a concrete of a Preposition, since it represents it on a lower level of abstraction.

Since there may be many different possibilities for the realizations of a concept, *modalities* H_l are introduced with the implication that each individual modality $H_l^{(k)}$ may define the concept C_k. Another possibility for the representation of different realizations of a concept is to define separate concepts for each realization; this, however, prevents a compact knowledge representation. Furthermore,

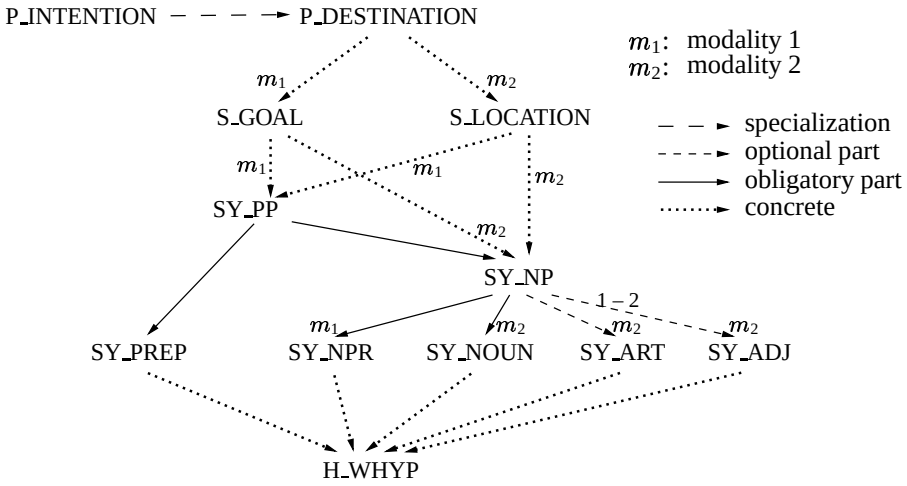

Figure 1. Example of a semantic network representation.

parts of a concept may be defined as being *obligatory* or *optional*. In Figure 1, for example, a Noun Phrase (SY_NP) can be defined by:

- Modality 1: obligatory part: *Proper Noun* (e.g. *"Berlin"*)
- Modality 2: obligatory part: *Noun* optional parts: *Article, Adjective* (e.g. *"the next train"*)

For the definition of properties or features, a concept C has a set of *attributes* A. There may also be *structural relations* S between the attributes of a concept. Each attribute and relation references a function F which computes the value of the corresponding attribute and a measure of the degree of fulfillment of the relation.

The occurrence of a specific pattern in the sensor data is represented by an *instance* $I(C)$ of the corresponding concept C. For the computation of $I(C)$, all attributes and relations of C have to be computed. Furthermore, instances for all the obligatory parts and concretes of the concepts have to exist (i.e., they have to be computed beforehand). Due to errors in the initial description $\mathcal{A}$ (arising from noise and processing errors) and ambiguities in the knowledge model $\mathcal{M}$ (arising, for example, from the various modalities), a *confidence measure* $\boldsymbol{G}$ is also defined. It is computed by a function F, which measures the degree of confidence regarding the occurrence of an instance $I(C)$ in the pattern $\boldsymbol{f}$ and its expected contribution to the success of the analysis of the whole pattern.

The goal of pattern analysis itself is represented by one or more *goal concepts* C_{g_i}. Consequently, an interpretation $\mathcal{B}$ of $\boldsymbol{f}$ is represented by an instance of a goal concept $I(C_{g_i})$. Now, in order to find the 'best' $\mathcal{B}$, an *optimal instance* $I^*(C_{g_i})$ has to be computed. Thus, the interpretation problem is an *optimization problem* and is solved as such. The semantic network formalism of ERNEST provides an A^*–based control algorithm for solving this problem, which we replace by the *parallel–iterative control* (cf. Section 4).

3 THE DIALOG SYSTEM EVAR

As a framework for our approach, the dialog system EVAR [9] is used. This system was initially developed using the semantic network formalism of ERNEST with the A^*–based control algorithm and is able to answer queries about the German train timetable. The knowledge base of EVAR is arranged in 5 levels of abstraction:

- *Word–hypotheses:* represents the interface between speech recognition and speech understanding; requests and verifies word hypotheses from the acoustic–phonetic front–end;
- *Syntax:* represents the level of syntactic constituents; identifies syntactic constituents in the set of word hypotheses;
- *Semantics:* is used to model verb and noun frames with their deep case; verifies the semantic consistence of the syntactic constituents, compounds them to larger ones, and performs task independent interpretation;
- *Pragmatics:* interprets the constituents sent by the semantic module in a task–specific context;
- *Dialog:* models possible sequences of dialog acts; operates in accordance with the level of identified intention of the spoken utterance.

Figure 2 shows an excerpt from the semantic network of EVAR on the dialog level. The ultimate goal of the dialog system is to answer an information query. Therefore, the system must be able to carry out a dialog with the user, since the user might not provide the system with all the necessary information for a database access in one single utterance. Thus, dialog steps are performed, alternately, by the user and by the system. The overall goal, an information dialog, is represented by D_INFO_DIA. It begins with the user's initial utterance, D_U_INF_REQ (e.g. *"hello, I want a train to Hamburg"*). If the system needs more information for database access, it asks for it (D_S_DEMAND, e.g. *"when do you want to leave?"*). The user supplies the requested information (D_U_SUPPL, e.g. *"tomorrow*

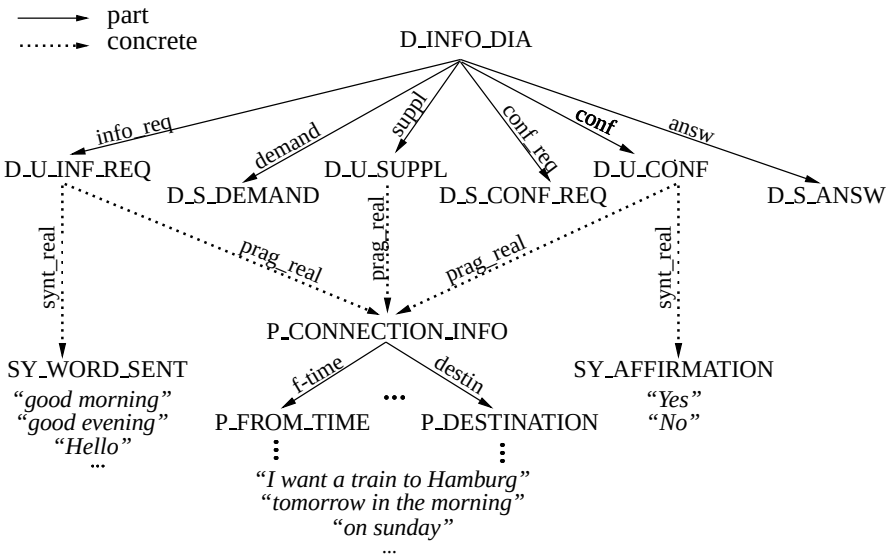

Figure 2. Excerpt of the knowledge base of EVAR, showing part of the dialog model.

in the morning"). The system then may request for a confirmation (D_S_CONF_REQ, e.g. "*do you want to travel to Hamburg tomorrow in the morning?*") depending on the chosen dialog strategy (see below). This is followed by a confirmation or a rejection by the user, D_U_PCONF or D_U_NCONF, respectively. If all necessary information is available, the system accesses the database, retrieves the requested information to the user (D_S_ANSW), and says goodbye (D_S_GOODBYE, not shown in the excerpt). The three different dialog strategies available are: ROBUST: since there may be errors in speech recognition, the system requests for confirmation; NL: the system does not request for confirmation (used e.g. for typed input); RA: only initial user information requests are processed (thus, the dialog consists of a single information request by the user and an answer by the system, if possible). It should be noted that these are only some of the 20 dialog steps represented in the knowledge base of EVAR.

4 A PARALLEL ANY–TIME CONTROL

The control algorithm for knowledge based pattern understanding which we employ (cf. [3]) treats the search for an optimal interpretation as a *combinatorial* optimization problem and solves it by means of *iterative* optimization methods, e.g. Simulated Annealing [7], Stochastic Relaxation [5], and Genetic Algorithms [6]. By using iterative methods one provides the system with *any–time* capability since after each iteration step a (sub–)optimal solution is always available. If the available computation time T is small, only few iterations and hence a coarse result are possible; if T is large, more iterations and a better result can be computed. Furthermore, the control algorithm allows an efficient exploitation of *parallelism* by compiling the concept–centered semantic network into a fine–grained task–graph, the so–called *attribute network* (cf. Figure 3) which represents the dependencies of all attributes, relations, and confidence measures of concepts to be considered for the computation of instances $I(C_{g_i})$.

4.1. Parallel Processing

Parallelism is exploited on the *network* and on the *control level*. On the network level, the attribute network may be mapped to a multiprocessor system for parallel processing (*parallel bottom–up instantiation*). On the control level, several competing instances of goal concepts may be computed in parallel (*parallel search*). This can be done using the PVM (**P**arallel **V**irtual **M**achine) [4] on a local network of heterogeneous workstations (e.g. $WS_1, \ldots, WS_p$ in Figure 3). The *attribute network* is a prerequisite for an efficient exploitation of parallelism and is *automatically generated* in two steps:

- **Expansion** Since each concept is stored exactly once in the knowledge base, but it may be necessary to create several instances for a concept during analysis (consider the example of the wheels of a car), all the instances of a concept needed for the instantiation of the goal concepts C_{g_i} are established by way of a top–down expansion of the C_{g_i}s.
- **Refinement** The expanded network is refined by the determination of dependencies between sub–conceptual entities (attributes, structural relations, confidence measures, etc.) of all concepts in the expanded network. For each sub–entity, a node v_k is created. Dependencies are represented by means of directed links $e_{lk} = (v_l, v_k)$ and express the fact that the computation of v_l must finish before the computation of v_k may start.

Nodes without predecessors represent attributes that provide an interface to the initial segmentation (for example a word–chain or a word–graph), and nodes without successors represent the confidence measures of goal concepts. For the computation of instances of the goal concepts all nodes of the attribute network are processed in a single bottom–up step. This step corresponds to a single iteration in the iterative optimization.

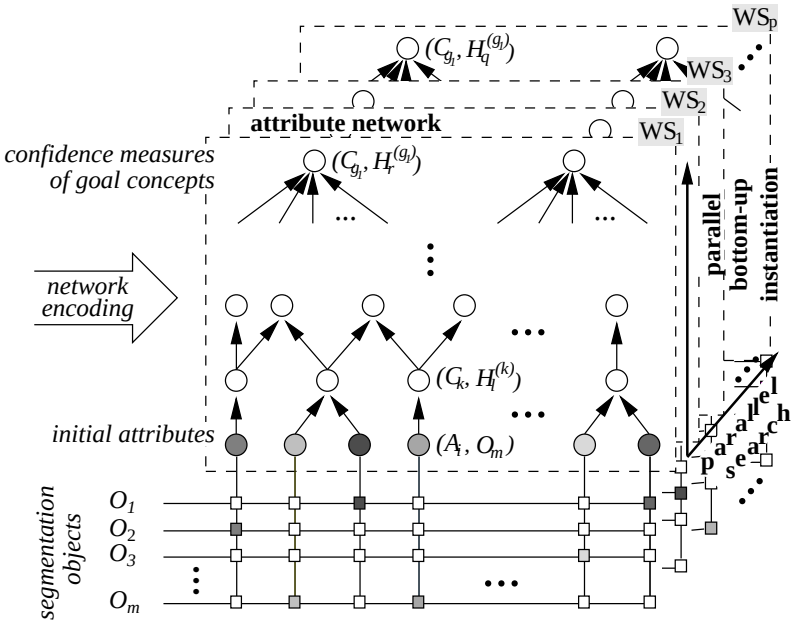

Figure 3. Scheme of the parallel iterative control algorithm.

4.2. Iterative Optimization

Since the results of the initial segmentation are usually erroneous and due to the presence of ambiguities in the knowledge base, the computation of an instance and its judgment (i.e. confidence measure) are completely determined by

- the assignment $(A_i, O_j^{(i)}), i = 1, \ldots, m$ of a segmentation object O_j to each attribute A_i of concepts representing the interface to the initial segmentation and
- the choice $(C_k, H_l^{(k)}), k = 1, \ldots, n$ of a modality H_l for each instance of a concept C_k which enables multiple definitions of the same object (cf. Section 2),

where m is the number of attributes of interface nodes and n the number of concepts having more than one modality associated to them. This allows us to characterize the *current state of analysis* by way of the following vector:

$$\boldsymbol{r}_c = \left[(A_i, O_j^{(i)}); (C_k, H_l^{(k)}) \mid i = 1, \ldots, m \; ; \; k = 1, \ldots, n \right].$$

The task of the control algorithm is now to find *that* state of analysis $\boldsymbol{r}_c^*$ which leads to an optimal instance of a goal concept $I^*(C_{g_i})$. Therefore, a cost function ϕ is introduced and $\boldsymbol{r}_c$ is treated as the current state of a combinatorial optimization problem.

In each iteration step i, instances for the goal concepts $I_{r_i}(C_{g_1}), \ldots, I_{r_i}(C_{g_n})$ and the corresponding costs ϕ_{r_i} are computed for the current state $\boldsymbol{r}_i$. Iterations are performed until an optimal solution is found or until no more computing time is available. Please recall that after each iteration step a (sub–)optimal solution is available.

In Figure 3 it is shown, for example, that in the current state of analysis for which the attribute network is computed on WS_1, the segmentation object O_m is assigned to the initial attribute node A_i, and modality $H_r^{(g_1)}$ is assigned to the goal concept C_{g_1}. On WS_p, modality $H_q^{(g_1)}$ is assigned to C_{g_1}, indicating that competing instances are computed for different states of analysis on the various workstations.

4.3. Expanding the Control for Dialog

Since in the context of a dialog not only the interpretation of the user's utterance is necessary, but also the management of the dialog as a whole, the control has to be extended. In our application (cf. Section 3), the overall goal concept is D_INFO_DIA, which we will call C_G. Let's consider, now, an attribute network computed for C_G. Since C_G models all user and system dialog steps, an instance for C_G can not be computed in one step (i.e., a bottom–up processing of all nodes of the attribute network for C_G is not possible). Thus, the instantiation of user dialog steps and system dialog steps have to be separated in the attribute network. Therefore, we divide our global optimization task, which is to find an optimal instance for C_G, into several local optimization tasks C_{g_i}, C_{a_i} (this notation was chosen in order to preserve the generality of the control). Now, for the computation of an instance of C_G, instances for subgoals C_{g_i} on the interpretation level (user dialog steps) have to be considered alternately with instances of subgoals C_{a_i} on the action level (system dialog steps). In order to handle this closed loop of alternating interpretations and actions over the time, a *control–shell* (cf. Figure 4) was implemented.

It computes, in a pre–processing step, sub–networks $dsub_{g_i}$ and $dsub_{a_i}$ out of the attribute network for all C_{g_i}s and C_{a_i}s, respectively. This is done off–line, before the analysis begins. Thus, for the computation of an instance $I(C_{\{g,a\}_i})$, only the corresponding sub–networks $dsub_{\{g,a\}_i}$ have to be bottom–up processed. Furthermore, a task–dependent function *next_dsub* is provided by this shell which decides whether an action or an interpretation (and which) is to be performed next (this decision is taken by means of some decision rules which were extracted from the former dialog–model implemented with the A^*–based control; it may be substituted by statistical methods in the future). It also has to decide which subgoal concepts are concurring at a point in time given the actual progression

Compute attribute subnets $dsub_{\{g,a\}_i}$ for all $C_{\{g,a\}_i}$
Initialize memory
Determine the set of initial subgoals $C_{\{g,a\}_{init}}$ and the corresponding subnets $dsub_{\{g,a\}_{init}}$
$dsub_{\{g,a\}_c} := dsub_{\{g,a\}_{init}}$; $C_{\{g,a\}_c} := C_{\{g,a\}_{init}}$
WHILE $dsub_{\{g,a\}_c} \neq$ NIL
Find an optimal instance for $C_{\{g,a\}_c}$ by an iterative bottom–up computation of $dsub_{\{g,a\}_c}$
Actualize memory
Compute with *next_dsub*: $dsub_{\{g,a\}_{next}}$ and $C_{\{g,a\}_{next}}$
$dsub_{\{g,a\}_c} = dsub_{\{g,a\}_{next}}$; $C_{\{g,a\}_c} := C_{\{g,a\}_{next}}$

Figure 4. Control–shell for dialog–steps.

of the dialog and the knowledge about the application domain. For example, a user may give a positive (e.g. "*yes, tomorrow*") or a negative ("*no, I want to go today*") confirmation to a system's confirmation request (cf. Figure 2). Thus, after the concept D_S_CONF_REQ was instantiated, *next_dsub* computes D_U_PCONF and D_U_NCONF as being the concurring goal concepts to be considered next. $dsub_{\{g,a\}_{next}}$ is NIL if the dialog represented by C_G has been completed. The information extracted in each instantiation and the progression of the dialog is stored in the *memory* of the system. In Section 5, results are cited regarding the user's initial utterance to show the efficiency of the control algorithm in this application domain. At present, the dialog steps D_U_INF_REQ, D_S_ANSW, and D_S_GOODBYE have been fully implemented for the new control.

5 EXPERIMENTAL RESULTS

The goal of initial experiments (cf. also [2]) was to evaluate the performance of the system with respect to processing speed and the percentage of correctly analyzed pragmatic intentions[1]; for example:

We want to go to Hamburg today
TRAVELLER DESTINATION DEP_TIME

Thus we only considered the user's first request, D_U_INF_REQ. Further experiments will be carried out to evaluate the dialog capabilities of the system. The context for our experiments was as follows:

- goal concept C_G of the attribute network was D_INFO_DIA;
- the attribute network itself consisted of about *10 500 nodes* and was generated for the dialog steps D_U_INF_REQ, D_S_DEMAND, D_U_SUPPL, D_S_ANSW, and D_S_GOODBYE;
- the *dialog strategy* was RA (cf. Section 3), i.e., the dialogs consisted only of the steps D_U_INF_REQ, D_S_ANSW, and D_S_GOODBYE;
- the optimization method used was *stochastic relaxation*;
- the optimization criterion was the *maximization of the number of words covered*;
- as input for the linguistic analysis we used the *transliterated* utterances (simulating a 100% word–accuracy).

Parallelization on the control level was simulated on a single processor (parallelization with PVM is being implemented at the moment). The attribute network was processed sequentially.

Two test corpora were used: a corpus of 146 *thought up* and *read* (i.e., not spontaneous) user's first utterances, 8.3 words (2.7 seconds) per utterance and a total of 447 pragmatic intentions (part of the ASL–SÜD corpus) and a corpus of *spontaneous speech* consisting of 327 user's first utterances collected over the public telephone network, 8.7 words (3.0 seconds) per utterance and a total of 1 023 pragmatic intentions (part of the EVAR–SPONTAN corpus). Table 1 shows the number of correctly analyzed pragmatic intentions (in %) for EVAR–SPONTAN and ASL–SÜD.

	ASL–SÜD				
n	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
1	88.4	92.3	92.7	95.9	95.9
5	87.4	92.1	94.4	96.4	97.2
10	91.9	94.6	95.5	97.4	97.9
25	93.1	96.1	97.0	98.5	99.6
50	96.6	98.9	99.4	100	100

	EVAR–SPONTAN				
n	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
1	73.6	76.7	77.6	87.8	88.0
5	75.6	82.5	83.4	88.3	88.9
10	83.3	88.0	89.1	90.0	90.1
25	85.3	89.4	90.8	91.1	91.6
50	89.5	91.3	92.0	92.0	92.2

Table 1. Percentage of correctly analyzed pragmatic intentions for n iterations and p processors on ASL–SÜD and EVAR–SPONTAN.

These results were obtained after a careful choice of an initial state of analysis vector (cf. Section 4.2.) based on the incoming word–chain and some heuristic rules (this initialization is presently being replaced by a statistical approach which showed to be successful in former experiments). On this account, quite good solutions were found in the first iteration step already. One can see that after $n = 5$ iterations using $p = 5$ processors 97% and 89% of all pragmatic intentions of the ASL–SÜD and EVAR–SPONTAN corpora, respectively, were found. The majority of the pragmatic intentions not identified were time specifications, which are syntactically more complex than the expressions of other intentions. Furthermore, after $n = 5, p = 5$, 97% and 94% of all destinations were correctly recognized for ASL–SÜD and EVAR–SPONTAN, respectively. This means that in almost all cases the system is able to maintain a dialog with

[1] These are pragmatic information units the system needs to recognize in order to react to the user's request; in EVAR they are represented by concepts on the pragmatic level, such as P_DESTINATION (cf. Figure 2).

the user by confirming the destination location and asking for information it has not yet acquired, e.g. the departure time.

The mean processing time for a single iteration was 0.2 seconds (on a 9000/735 HP–Workstation). This can be translated to a *real–time* factor of ≈ 0.7 for $n = 1$, $p = 5$. As mentioned before, these results were obtained for the user's initial utterance. Nevertheless, first experiments for the dialog steps D_U_INF_REQ, D_S_ANSW, and D_S_GOODBYE were carried out. The additional processing time, which comprises the prediction of the new dialog step to be performed, and the instantiation of D_S_ANSW (including access to the database) and D_S_GOODBYE is about 0.3 seconds. A speed–up by a factor of approx. 10 could be achieved in first rudimentary experiments, comparing the system with the new control to the former system with A^*–based control. This has to be confirmed by further experiments.

6 CONCLUSION AND FUTURE WORK

In this paper we presented a dialog system based on a semantic network formalism for knowledge representation and on a parallel, iterative control algorithm with any–time and real–time capabilities. Experimental results have proved the feasibility of the approach.

Future work will concentrate on completing the implementation of the approach for all dialog steps. We will also consider the implementation of incremental speech analysis, which will result in a further performance improvement. In the long term, the integration of speech and image analysis will be of great importance for pattern analysis applications such as the control of a robot for the household chores. The approach presented here can provide the appropriate uniform formalism.

REFERENCES

[1] M. Evett, J. Hendler, and L. Spector. Parallel Knowledge Representation on the Connection Machine. *Journal of Parallel and Distributed Computing*, 22(2):168–184, 1994.

[2] J. Fischer and H. Niemann. Applying a parallel any–time control algorithm to a real–world speech understanding problem. In *Proceedings of the 1997 Real World Computing Symposium*, pages 382–389, Tokyo, 1997. Real World Computing Partnership.

[3] V. Fischer and H. Niemann. A Parallel Any–time Control Algorithm for Image Understanding. In *Proceedings of the* 13^{th} *International Conference on Pattern Recognition (ICPR)*, Wien, October 1996. IEEE Computer Society Press.

[4] G. Geist and V. Sunderam. The PVM system: Supercomputing level concurrent computations on a heterogenous network of workstations. In *Proc. of the 6th IEEE Conference on Distributed Memory Computing*, pages 258–261, Portland, Oreg., 1991.

[5] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[6] D. Goldberg. *Genetic Algorithms: Search, Optimization and Machine Learning*. Addison–Wesley Publ. Co., Reading, Mass., 1989.

[7] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[8] F. Kummert, H. Niemann, R. Prechtel, and G. Sagerer. Control and explanation in a signal understanding environment. *Signal Processing*, 32(1-2, Special Issue on Intelligent Systems for Signal and Image Understanding):111–145, 1993.

[9] M. Mast, F. Kummert, U. Ehrlich, G. Fink, T. Kuhn, H. Niemann, and G. Sagerer. A Speech Understanding and Dialog System with a Homogeneous Linguistic Knowledge Base. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2):179–194, 1994.

[10] H. Niemann, G. Sagerer, S. Schröder, and F. Kummert. ERNEST: A semantic network system for pattern understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9:883–905, 1990.

[11] H. Noda and M. N. Shirazi. A MRF–Based Parallel Processing Algorithm for Speech Recognition Using Linear Predictive HMM. In *Image and Multidimensional Signal Processing*, volume 1, pages 597–600, Adelaide, 1994.

[12] I. Pitas, editor. *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*. Wiley Series in Parallel Computing. John Wiley & Sons, 1993.