SPEECH UNDERSTANDING AND SPEECH TRANSLATION IN VARIOUS DOMAINS BY MAXIMUM A-POSTERIORI SEMANTIC DECODING

Johannes Müller (1), *Holger Stahl* (2)

(1) Institute for Human-Machine-Communication, Munich University of Technology, D-80290 Munich, Germany
mue@mmk.e-technik.tu-muenchen.de
(2) Rohde & Schwarz GmbH & Co. KG, Muehldorfstrasse 15, D-81671 Munich, Germany
holger.stahl@rsd.de

ABSTRACT

This paper describes a domain-limited system for speech understanding as well as for speech translation. An integrated semantic decoder directly converts the preprocessed speech signal into its semantic representation by a maximum a-posteriori classification. With the combination of probabilistic knowledge on acoustic, phonetic, syntactic, and semantic levels, the semantic decoder extracts the most probable meaning of the utterance. Any separate speech recognition stage is not needed because of the integration of the Viterbi-algorithm (calculating acoustic probabilities by the use of Hidden-Markov-Models) and a probabilistic chart parser (calculating semantic and syntactic probabilities by especial models).

The semantic structure is introduced as representation of an utterance's meaning. It can be used as intermediate level for a succeeding intention decoder (within a speech understanding system for the control of a running application by spoken inputs) as well as interlingua-level for a succeeding language production unit (within an automatic speech translation system for the creation of spoken output in another language).

Following the above principles and using the respective algorithms, speech understanding and speech translation front-ends for the domains 'graphic editor', 'service robot', 'medical image visualization' and 'scheduling dialogues' could be successfully realized.

1 INTRODUCTION AND SYSTEM OVERVIEW

Speech is the perfect and common medium for human-to-human information exchange; it has been developed for many centuries. Natural speech has a lot of advantages for human-machine-interaction, too: Since it does not require any learning process, speech can be used without hands and in darkness, being a fundamental contribution for ergonomic multimodality. So why do we not speak with our technical systems in daily routine? Speech understanding can be interpreted as the conversion of a speech signal to the user's intention, which means to solve a huge ambiguity! An utterance can be represented at various abstraction levels – not only at the semantic or pragmatic level, but also as word chain, phoneme chain, as observation sequence, or as speech signal, depending on the speaker, the context and environment.

If these ambiguities are solved step by step, a global optimization of the utterance's meaning is not possible. Instead of connecting a speech recognition stage and an independent syntactic and semantic analysis stage, our approach calculates the meaning S_E of the utterance in a single semantic decoding step from the observation sequence O (preprocessed speech signal). For this purpose, the speech knowledge of the domain is level-specifically stored in acoustic, phonetic, syntactic, and semantic models, see fig. 1.

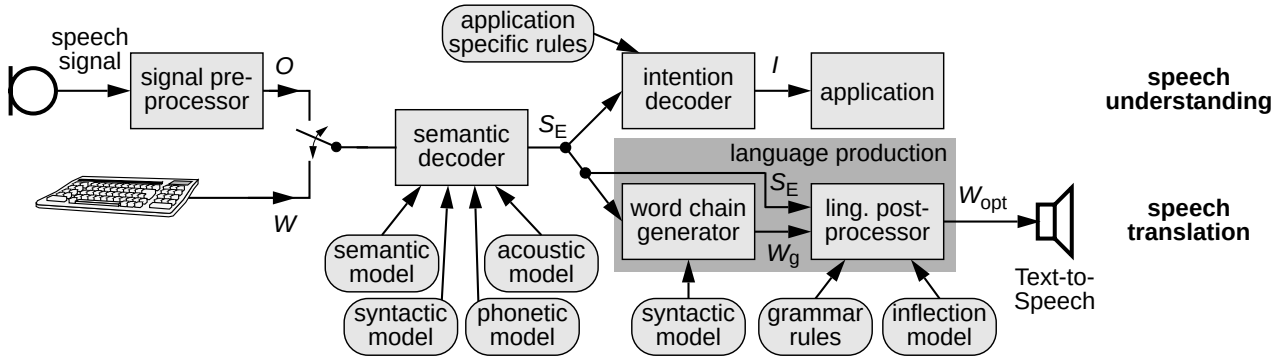

Fig. 1: System overview: The semantic decoder operates in a single step by the help of purely probabilistic knowledge. The semantic output S_E serves as intermediate representation for speech understanding and as interlingua-level for speech translation.

- For **speech understanding**, the meaning S_E is processed by a rule-based intention decoder for obtaining a code to control the behaviour of the application in the user's sense.
- For **speech translation**, 'reverse' semantic decoding is realized for producing a word chain W_g, using a syntactic model of the target language. Possible syntactic mistakes are corrected in the following linguistic post-processing module and spoken output is synthesized by a Text-to-Speech (TTS) system.

2 THE SEMANTIC STRUCTURE

The definition of the semantic representation (i.e. the meaning of the utterance) is essential for the required effort of all modules, for the prospect of realizing it all, and – strictly speaking – for the success of the whole system. Hence, we look for a semantic representation, which should be

- **close to the word level** to keep the dependency between the word chain and the desired meaning representation under control,
- **hierarchical** to cover nested semantic constructs,
- **logically correct** to consistently and comprehensibly represent semantic concepts,
- **generalizing** to identically represent semantically equivalent, but different utterances, and
- **close to the machine level** for a simple transformation of the classified semantic representation into machine comprehensible commands.

2.1 Definition of the Semantic Structure

According to these requirements, we introduce the *semantic structure S* as semantic representation of an utterance within a restricted domain [12][15][20]. It is hierarchic like a tree, which consists of N *semantic units* (abbreviated *semuns*) s_n :

$$S = \{s_1, s_2, \ldots, s_n, \ldots, s_N\} \tag{1}$$

Each semun s_n can be described by (X+2) components, its type $t[s_n]$, its value $v[s_n]$ and $X \geq 1$ references to its successors $q_1[s_n]$, …, $q_X[s_n] \in \{s_{n+1}, \ldots, s_N, \text{blk}\}$:

$$s_n = \left(t[s_n], v[s_n], q_1[s_n], \ldots, q_X[s_n] \right) \tag{2}$$

- The **type** $t[s_n]$ lays down the number X of successors[1)] and restricts the set of possible successor-types $t[q_1[s_n]]$, … , $t[q_X[s_n]]$. Furthermore, it makes a selection of the corresponding values $v[s_n]$.
- The **value** $v[s_n]$ shows the exact meaning of s_n.

1) This means: $X = X(t[s_n])$. At present, we use $1 \leq X \leq 5$.

- Each **successor** $q_x[s_n]$ specifies a certain fact of the semun s_n. If the utterance contains that certain specification, the successor $q_x[s_n]$ is identical with another semun within S. In that case, the successor is denoted as *successor semun*

$$q_x[s_n] \in \{s_{n+1}, \ldots, s_N\}. \tag{3}$$

If the utterance does not contain that certain specification, then it is a *blank successor*

$$q_x[s_n] = \text{blk}. \tag{4}$$

For the consistent description of our stochastic approach, it is necessary to allow a type for the blank successor:

$$t[\text{blk}] = \text{blk} \tag{5}$$

The types of all successors $q_1[s_n]$, …, $q_x[s_n]$, …, $q_X[s_n]$ with $1 \leq x \leq X$ of the semun s_n are denoted as *successor group*

$$t_q(s_n) = \left(t[q_1[s_n]] , \ldots, t[q_x[s_n]] , \ldots, t[q_X[s_n]]\right). \tag{6}$$

A semun s_n together with all references to its X successors $q_1[s_n]$, …, $q_x[s_n]$, …, $q_X[s_n]$ can be graphically depicted as shown in the following figure. The successors are numbered from 1 to X from top to bottom. The reference to a successor semun is marked by the edge "⟶". In contrast to this, the edge "⟙" marks the reference to a blank successor:

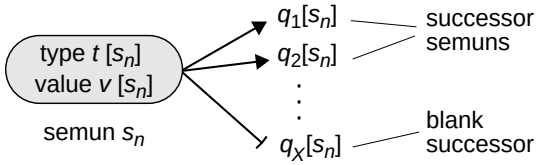

Fig. 2: Illustration of a semun s_n with X successors

The whole semantic structure S forms a 'tree' with the semun s_1 as 'root' and the blank successors as 'leaves'. All other semuns s_2, …, s_N belong to exactly one predecessor semun.

A **branch**, denoted as $S(s_n)$, is formed by a semun $s_n \in S$ with (recursively regarding) all its successor semuns down to all terminating blank successors. The semuns within each branch $S(s_n)$ are consecutively indexed starting from n, blank successors are not included.

$$S(s_n) = \{s_n, s_{n+1}, s_{n+2}, \ldots\} \tag{7}$$

Hence, each branch is a subset of the semantic structure:

$$S(s_n) \subset S \quad \text{for} \quad n \neq 1 \tag{8}$$

According to the previous explanations, the branch $S(s_1)$ of the root semun s_1 is identical to the whole semantic structure S.

$$S(s_1) = S \tag{9}$$

Each single semun represents a small semantic part of the whole utterance. The significance of a semun is specified by its successors, which are in a fixed relation to each other. Each branch $S(s_n) \subset S$ represents a connected part of the total meaning.

A **line** is a direct sequence of m+1 semuns $s_n, \ldots, s_{n+m}$ within a branch, each one with respectively X=1 successor. A line starts with the successor semun of a semun with $X \geq 2$ successors and is terminated by a semun, which has X=1 blank successor, or by the predecessor semun of a semun with $X \geq 2$ successors. The order of the semuns within a line has no influence on the meaning of the whole semantic structure – clearly spoken:

The order of semuns within a line is optional!

Two semantic structures are *equivalent*, if they contain the same information. An equivalent semantic structure comes into being (apart from the restrictions of the syntactic model) by semun permutations within a line.

Fig. 3 shows an example for a semantic structure S_1 ('graphic editor' domain) in a graphic depiction:

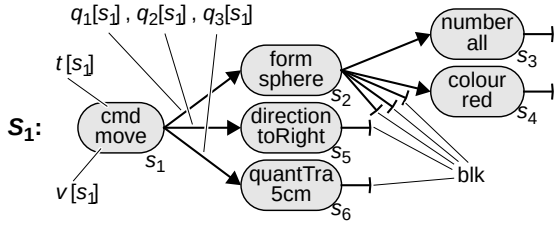

Fig. 3: Semantic structure S_1 corresponding to the utterance *"move all red spheres five centimetres to the right"*

Basically, the structure is quite similar to the notation in Valence/Dependence Grammar [25]. As central part, the respective "ruling" command verb forms the root of the semantic structure. Compared to first order predicate logic [8][10], a single semun can be seen as an X-place relational constant, in which a 0-place relational constant is modelled by X=1 blank successor. However, the connection of single semuns to a semantic structure essentially differs from the representation by predicate logic. Although mathematically not as exact, the semantic structure offers the following advantages:

- The semantic structure S is a meaning representation close to the word level and allows an immediate and probabilistic correlation to a word chain W. It only considers the semantic content of the utterance without any respect of the pragmatic constellation or the dialogue status. Since the semantic structure has not any knowledge about previous utterances, ellipses, anaphoric or kataphoric references cannot be resolved – this must happen in a postprocessing stage.
- For the connection of semuns, there is only one single mechanism, namely referencing to other semuns as respective successor semuns. Therefore, only this one kind of connection has to be considered during the design of the models.

2.2 Relation to the Word Chain

The semantic structure is a meaning representation close to the word level. Its structure is dependent on the word choice and, although conditionally, on the word order of the corresponding word chain. Nevertheless, it has the capability for generalization: Different, but semantically equivalent word chains correspond to identical semantic structures. For this purpose, the following determinations are established:

- Each word of the word chain W corresponds to exactly one semun s_n of the semantic structure S.
- Each semun s_n of the semantic structure S is assigned to exactly one significant word $w^+[s_n]$ and at most one insignificant word $w^-[s_n]$ of the word chain W. In this manner, two or more words (in morphological sense) can possibly be combined to one contiguous "word". Since only the corresponding semantic structure is important as final result, a morphologically correct word representation is not absolutely necessary. Therefore, the design of the syntactic model (estimation of the probability $P(W|S)$ for the occurrence of the word chain W given the semantic structure S) is possible without any further representation levels.
- Each branch $S(s_n) \subseteq S$ corresponds to an unbroken phrase $W(s_n) \subseteq W$.

As an example, the relation between the German word chain *"bitte den schönen grünen quader neben der ähm roten kugel löschen"* (*"please delete the nice green cuboid besides the ahm red sphere"*) and its semantic

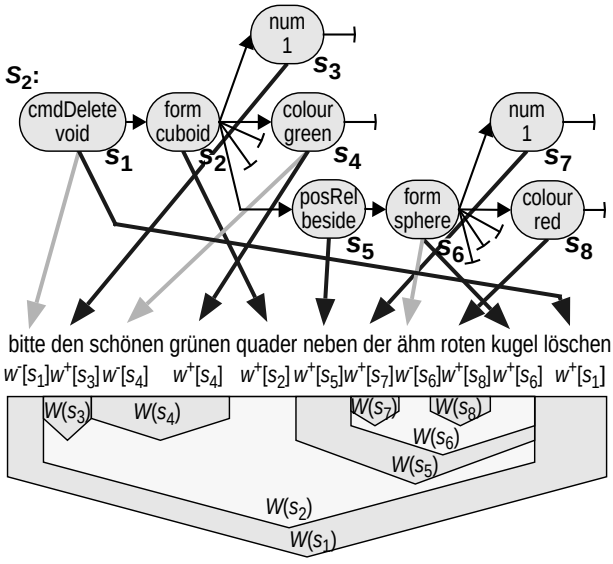

Fig. 4: Semantic structure S_2 and corresponding phrases $W(s_n)$

structure S_2 is shown in fig. 4. The significant words $w^+[s_n]$ of the semuns s_n are marked by dark arrows "⟶", the insignificant words $w^-[s_n]$ (*"bitte"*, *"schönen"*, *"ähm"*) by bright arrows "⟶". The phrase $W(s_n)$ belonging to the respective branch $S_2(s_n)$ is shown below. Please note, that the alignment of these phrases depends on the hierarchy of the underlying semuns.

3 SIGNAL PREPROCESSING

To obtain speaker independence and high performance of the acoustic models, the observation sequence O (input of the semantic decoder) should obtain highly discriminative information about the content of the utterance, but it should not be influenced by the speaker's personality, by the input channel or noise. The next fig. 5 shows the signal preprocessing, which can be well used with Hidden-Markov-Models (HMMs) [1][9][16][19].

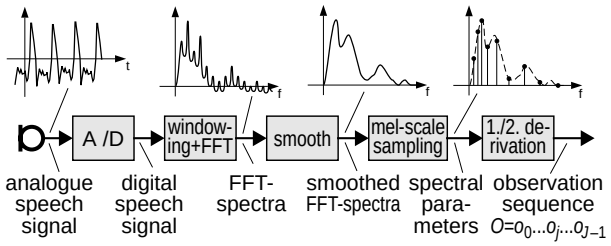

Fig. 5: Block diagram of the signal preprocessing module

The signal preprocessing (or 'feature extraction') module creates 64-dimensional observation vectors (or 'feature vectors' or 'input frames') o_j in intervals of 10 ms, each of them describing the spectral characteristics of the speech signal contained in a 25 ms-wide window [2]. The time alignment of these observation vectors o_j is the observation sequence $O = o_0 \ldots o_j \ldots o_{J-1}$. Similar to the task of stochastic word chain decoding (speech recognition), the semantic decoder uses this observation sequence as input for stochastic pattern matching.

4 SEMANTIC DECODING

Our semantic decoder converts a spoken utterance (given as observation sequence O) into its semantic representation (in our approach denoted as semantic structure S). From the set of all possible S, that one S_E has to be found which is the most probable given the observation sequence O, i.e. which maximizes the a-posteriori-probability $P(S|O)$. The resulting term can be transformed using the Bayes formula.

$$S_E = \underset{S}{\operatorname{argmax}}\, P(S|O) = \underset{S}{\operatorname{argmax}}\, \frac{P(O|S) \cdot P(S)}{P(O)} \quad (10)$$

Since $P(O)$ is not relevant for maximizing, it can be neglected:

$$S_E = \underset{S}{\operatorname{argmax}}\, [P(O|S) \cdot P(S)] \quad (11)$$

Due to the high variety of S and O, it is not possible to estimate $P(O|S)$ directly. Therefore, additional representation levels are necessary. Well defined are the word chain W and the phoneme chain Ph, which can be used for the calculation of S_E [18]:

$$\begin{aligned} S_E &= \underset{S}{\operatorname{argmax}} \max_W \max_{Ph} \left[P(O|Ph)P(Ph|W)P(W|S)P(S)\right] \\ &= \underset{S}{\operatorname{argmax}} \max_W \max_{Ph} P(O, Ph, W, S) \end{aligned} \quad (12)$$

Eq. (12) is the maximum a-posteriori (MAP) classification formula, which is implemented 'top-down' for finding that semantic structure S_E, which is included in the most likely combination of a semantic structure S, a word chain W, a phoneme chain Ph and the given observation sequence O.

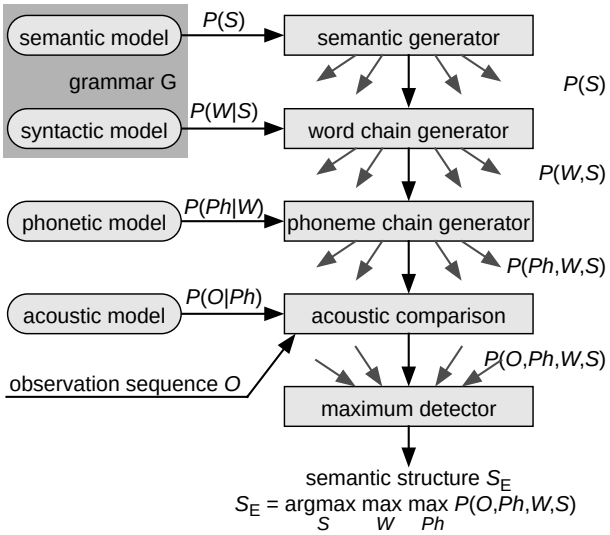

Fig. 6: The 'top-down' principle for semantic decoding

In the above equations, we assume statistical independence of all probabilities, which are stored in four stochastic knowledge bases (called "models"):

- The semantic model delivers the a-priori probability $P(S)$ for the occurrence of a semantic structure S.
- The syntactic model delivers the conditional probability $P(W|S)$ for the occurrence of a word chain W given a certain semantic structure S.
- The phonetic model delivers the conditional probability $P(Ph|W)$ for the occurrence of a phoneme chain Ph given a certain word chain W.
- The acoustic model delivers the conditional probability $P(O|Ph)$ for the occurrence of an observation sequence O given a certain phoneme chain Ph. It contains parameters used in a multi-dimensional HMM.

Phonetic and acoustic models are not necessary to decode written text, since in case of written input (i.e. word chain W) eq. (12) can be simplified to

$$S_E = \underset{S}{\operatorname{argmax}} \; [P(W|S) \cdot P(S)] \; . \qquad (13)$$

These probabilities have to be estimated by counting the occurring frequencies over a large set of training data, which are authentic limited-domain utterances, each represented by semantic structure, word chain, phoneme chain and observation sequence. A detailed description of the semantic decoder, which is implemented as an incremental 'top-down'-parser combining a modified Earley-parsing [3] and a Viterbi-beam-search [26] algorithm, and the consistent integration of all stochastic knowledge can be found in chap. 4.2 and in [24].

4.1 The Stochastic Grammar

4.1.1 Probabilities in the Semantic Model

If statistical dependencies are assumed only inside of each semun, $P(S)$ can be calculated as product of certain first order probabilities.

$$P(S) = f_0 \cdot \prod_{n=1}^{N} (e_n \cdot f_n) \quad \text{with:} \qquad (14)$$

- The **root probability f_0** denotes the a-priori probability that the root semun s_1 has the type $t[s_1]$:

$$f_0 = P(t[s_1]) \qquad (15)$$

  The semantic model has to provide f_0 for all types.

- The **value probability e_n** denotes the conditional probability that the value $v[s_n]$ occurs with a semun s_n of the type $t[s_n]$:

$$e_n = P\Big(v[s_n] \,\Big|\, t[s_n] \Big) \qquad (16)$$

  The semantic model has to provide this probability for all combinations of any type $t[s_n]$ and any value $v[s_n]$.

- The **succession probability f_n** denotes the conditional probability that the successor group $t_q(s_n)$, which consists of the types $t[q_1[s_n]]$, …, $t[q_X[s_n]]$ as defined in eq. (6), belongs to a semun s_n with the type $t[s_n]$:

$$f_n = P(t_q(s_n) | t[s_n]) \qquad (17)$$

  The semantic model has to provide this probability for all combinations of any type $t[s_n]$ and all possible successor groups $t_q(s_n) = \big(t[q_1[s_n]] , \ldots, t[q_X[s_n]]\big)$.

4.1.2 Probabilities in the Syntactic Model

For each semantic structure, a *syntactic network* (a probabilistic finite state automaton) can be set up with states for emitting words and transitions between these states. The word choice is directly influenced by the emissions, the word order by the transitions within the syntactic network.

The semantic structure is a hierarchic tree consisting of N semantic units (semuns) s_n with $1 \le n \le N$. Corresponding to that fact, the respective semantic network is also hierarchically built of N syntactic modules (SM) $A(s_n)$ with $1 \le n \le N$. Each semun s_n corresponds to exactly one SM $A(s_n)$.

Each SM $A(s_n)$ controls the choice and the alignment of the words, which are assigned to the semun s_n. A SM is a probabilistic state automaton, which consists of (X+4) states or nodes beg(s_n), $A[q_1(s_n)]$, ..., $A[q_X(s_n)]$, $B(s_n)$, $C(s_n)$, and end(s_n). X marks the number of successors of s_n, which is fixed by the respective type $t[s_n]$.

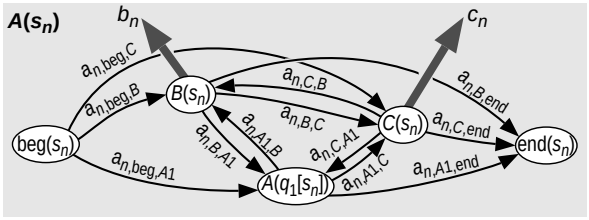

Fig. 7: Synt. module $A(s_n)$ for a semun s_n with X=1 [12][15]

- The **node beg(s_n)** is entered by the predecessor SM. All other states of the SM can only be entered after beg(s_n). For n=1, beg(s_1) is the starting point of the path through the whole syntactic network.
- Each **state $A(q_x[s_n])$** with $1 \le x \le X$ must distinguish:
  - If $q_x[s_n]$ is a successor semun, this state represents (as denoted by the expression) the SM $A(q_x[s_n])$ of the corresponding successor semun $q_x[s_n]$. Entering the state $A(q_x[s_n])$ is equivalent to entering the starting node beg($q_x[s_n]$). As soon as the terminal node end($q_x[s_n]$) is reached, $A(q_x[s_n])$ can be regarded as processed and can be left.
  - If $q_x[s_n]$ is a blank successor, $A(q_x[s_n])$ can be regarded as processed immediately after entering and can be left, without touching any other SM.
- The **state $B(s_n)$** emits a significant word $w^+[s_n]$ of the word chain W with the emission probability b_n.
- The **state $C(s_n)$** emits an insignificant word $w^-[s_n]$ of the word chain W with the emission probability c_n.
- The **node end(s_n)** terminates the path and jumps to this state of the predecessor SM, from where the actual SM has been reached. For n=1, end(s_1) terminates the path through the whole syntactic network.

The transition between two consecutive states is primarily determined by the respective transition probabilities, e.g. the transition probability $a_{n,B,C}$ marks the probability for the transition from the state $B(s_n)$ to the state $C(s_n)$. Due to the determinations made in chap. 2.2, the follow-

ing conditions must be additionally kept along the path from beg(s_n) to end(s_n):

- None of the states $A_1(s_n)$, ..., $A_X(s_n)$, $B(s_n)$ or $C(s_n)$ can be entered twice.
- The node end(s_n) must not be entered before the states $A(q_1[s_n])$, ..., $A(q_X[s_n])$ and $B(s_n)$.

This guarantees that each of the states $A(q_1[s_n])$, ..., $A(q_X[s_n])$ and $B(s_n)$ is entered precisely once and the state $C(s_n)$ at most once, with the consequence, that each successor is entered exactly once as well as one significant word and at most one insignificant word are generated. With these additional conditions, a SM cannot be a first order stochastic automaton.

For the calculation of the conditional probability $P(W|S)$ in a good approximation, we need the following first order conditional probabilities:

- The **path probability a_n** denotes the conditional probability that a certain path is followed within a SM, which corresponds to a semun of the type $t[s_n]$. It is the product of all transition probabilities $a_{n,\mu,\nu}$ along the path through the SM $A(s_n)$ from beg(s_n) to end(s_n), which results in the maximum $P(W|S)$:

$$a_n = P\Big(\text{certain path } \underline{\underline{\text{maximizing}}}\ P(W|S)\,\big|\,t[s_n]\Big) = \prod_{\substack{\text{all transitions}\\ \text{from } \mu \text{ to } \nu\\ \text{along a certain}\\ \text{path through } A(s_n)}} a_{n,\mu,\nu} \qquad (18)$$

  This probability depends <u>only on the type</u> of the respective semun. The syntactic model has to provide a matrix of $(X+2)(X+3)$ transition probabilities $a_{n,\mu,\nu}$ for each type.

  Please note: If the optional state $C(s_n)$ is entered and an insignificant word $w^-[s_n]$ is emitted, there are $(X+3)$ transitions along the path from beg(s_n) to end(s_n). If $C(s_n)$ is missed and no insignificant word is emitted, the number of transitions is reduced to $(X+2)$. For any blank successor $q_x[s_n]$, several paths exist within the SM $A(s_n)$, because the respective successor module $A(q_x[s_n])$ does not cause any further action. Hence, $A(q_x[s_n])$ can be entered at any time. In this case, that path with the maximum $P(W|S)$ is preferred.

- The **emission probability b_n** denotes the conditional probability that the state $B(s_n)$ of a SM, which corresponds to a semun s_n of the type $t[s_n]$ and the value $v[s_n]$, emits the significant word $w^+[s_n]$:

$$b_n = P\Big(\text{emission of } w^+[s_n]\,\big|\,t[s_n], v[s_n]\Big) \qquad (19)$$

  This probability depends <u>on both the type and the value</u> of the respective semun. The syntactic model has to provide this emission probability for all combinations of any type, any value and any significant word.

- The **emission probability c_n** denotes the conditional probability that the state $B(s_n)$ of a SM, which corresponds to a semun s_n of the type $t[s_n]$ and the value $v[s_n]$, emits the insignificant word $w^-[s_n]$. For simplifying eq. (21), c_n is equal to one, if the path through the SM does not contain the optional state $C(s_n)$.

$$c_n = \begin{cases} P\Big(\text{emiss. of } w^-[s_n]\,\big|\,t[s_n]\Big), & \text{if } C(s_n) \\ 1, & \text{else} \end{cases} \qquad (20)$$

  If $C(s_n)$ is entered, this probability depends <u>only on the type</u> of the respective semun. The syntactic model has to provide this emission probability for all combinations of any type and any insignificant word.

The **module probability d_n** of the SM $A(s_n)$ can be calculated by multiplying the path probability and both emission probabilities of the SM:

$$d_n = a_n \cdot b_n \cdot c_n \qquad (21)$$

With the assumption of statistical independence of all module probabilities d_n, the conditional probability $P(W|S)$ results in the product of the module probabilities of all N SMs within the syntactic network:

$$\boxed{P(W|S) = \prod_{n=1}^{N} d_n = \prod_{n=1}^{N} (a_n \cdot b_n \cdot c_n)} \qquad (22)$$

4.2 The Search Algorithm

For the search, an <u>active chart parser</u> [3] is applied. The parsing algorithm (see [22][24]) consists of two layers:

- The *grammar layer* is an active chart-parser which was augmented by a probabilistic dynamic programming mechanism to satisfy the MAP-classification in eq. (12). The parser is consequently realized in a top-down strategy and incrementally processes the input left to right. In this layer, the probabilities $P(S)$ and $P(W|S)$ are evaluated by using the semantic and the syntactic models.
- The *pronunciation layer* contains word hypotheses, each represented by one HMM-trellis, which is processed by Viterbi beam-search [16][26]. In this layer, the probabilities $P(Ph|W)$ and $P(O|Ph)$ are evaluated by using the phonetic model and the acoustic Hidden-Markov-Models on the observation sequence.

The grammar layer is tightly coupled to the pronunciation layer, the edges in the chart predict potential word hypotheses in the pronunciation layer by *push-word operations*. After having consumed a certain part of the observation sequence, each word hypothesis acquires a

score for matching the respective number of feature vectors. Every time the end state of the respective word HMM is reached, a *pop-word operation* is invoked to extend those edges, which formerly triggered the affiliated *push-word operation*.

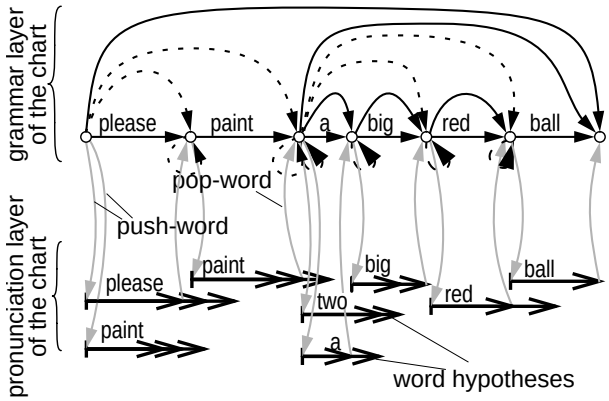

Fig. 8: Integration of word HMMs in an active chart

4.2.1 Representation of Hypotheses in the Search

During the search, the knowledge about open hypotheses is kept in so-called *items* [17], each representing a certain subparse referring to a certain SM $A(s_n)$. An item contains information about the matching between input frames $o_i \ldots o_{j-1}$ and one single semun s_n:

- **Syntactic features:**
  - The type $t[s_n]$,
  - the set of states within $A(s_n)$, which have been already encountered, and
  - the set of states, which still have to be encountered before leaving $A(s_n)$.
- **Position features:**
  - The start index and the end index of the frames of the observation sequence, which already have been matched to the semun s_n, and
  - the indices in the observation sequence, which were matched to the successor semuns $q_1[s_n]$ ... $q_X[s_n]$.
- **Semantic features:**
  - The value $v[s_n]$ and
  - the successor group $t_q(s_n)$.
- **Probabilistic features:**
  - The *inside-score* denotes the score[2)] summed up inside $A(s_n)$. It includes all partial probabilities contributing to the MAP estimation of eq. (12), accumulated along this subparse since the start frame.
  - The *overall-score* contains the inside score as well as the accumulated score for the left context of the item, i.e. the complete history to enter $A(s_n)$.

2) Each probability is transformed into its negative logarithm, which is the *score*, to replace multiplications by faster additions.

4.2.2 Item Lists and Agendas

All items, which have been generated after processing frame o_{j-1}, form the *item list* L_j. Each item list is split into two parts: An *agenda* (the active part) contains items which still have to be processed by the control structure, a passive part contains items which already have been processed. Items can be *checked* in and *retrieved* from the list:

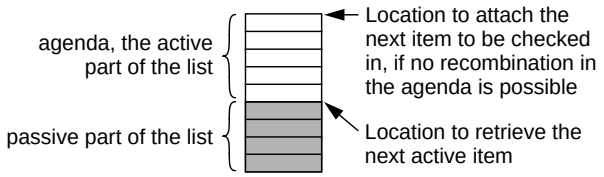

Fig. 9: Organization of an item list and operations on it

Whenever an item is checked in, it is tested for recombination with another item in the list. Items are *equivalent*, if they concur in all their properties except the scores. Equivalent items can recombine! That item with the lower overall-score (i.e. the higher probability) survives, the other is discarded. Instead of attaching each item to be checked in, the following strategy is chosen to maximize the efficiency:

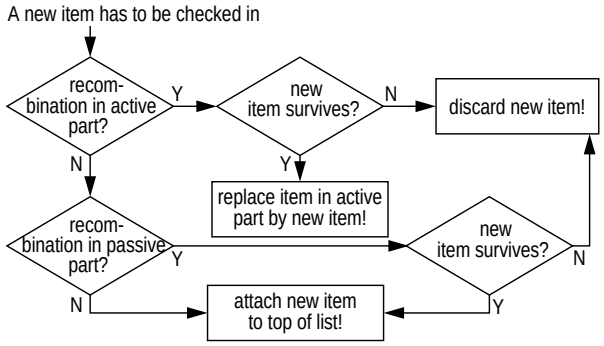

Fig. 10: Strategy for recombination of items to be checked in

After retrieving an item, the pointer on the location to retrieve the next one is incremented. Actually, the item lists can be seen as FIFO, implementing a breadth-first search.

4.2.3 Time Synchronous Construction of the Chart

Fig. 11 illustrates the function of item lists and the operations on them. First, item list L_0 is generated by the *initialization* ①. Subsequently, one item at a time is retrieved from the agenda of list L_0 and is processed by in-list operations ②. New items are checked in again in the same item list. This process continues until the agenda is empty, i.e. no items remain to be processed by in-list operations.

Afterwards, all items in the list L_0 are processed to start up new words in the pronunciation layer. We call this operation *push-word* ③. As a counterpart to this, the *pop-word* operation ④ pops out words which have been com-

pletely matched to a sequence of frames by the pronunciation layer. From these words, the next item list L_1 is then initialized.

Now, another round of ②, ③, and ④ is time-synchronously repeated for each of the input frames, generating one item list after another, until all J frames are consumed. All inactive item lists L_0, …, L_j, which remain after processing frame o_j are kept back in the chart for the completion steps (in ②) and ④.

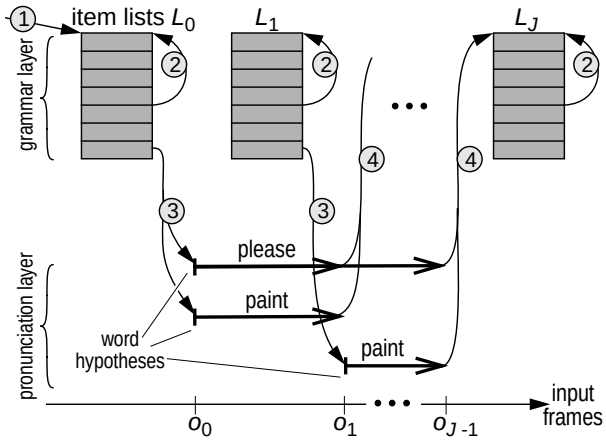

Fig. 11: Illustration of the main steps of the algorithm: The initialization step ①, the in-list operations ②, the push-word operation ③, and the pop-word operation ④.

Note that recursive constructs in the grammar do not cause any problem for the parser, since they will be caught as equivalent subparses, which can recombine to avoid a loop! In contrast to the easy task of parsing deterministic input, both the words itself and their boundaries are ambiguous with our task. Without countermeasures, the search space grows unimaginably huge already after a few input frames.

4.2.4 Improving Efficiency

To reduce both computation and memory effort down to manageable values, we applied a beam search technique [16]. Our incremental algorithm guarantees that all paths ending in the items contained inside one item list L_j have consumed exactly the same number j of input frames. This is an important claim for a direct comparison of the items' overall-scores for pruning, which is carried out both in the grammar layer and in the pronunciation layer.

The number of word hypotheses to be started up, can be further reduced, if the grammar layer is not activated for each frame index. The search effort (computation time) can be significantly reduced by this process, as shown in fig. 12 and fig. 21. In practice, after finishing the push-word operations ③, a number of frames is skipped before invoking the next pop-word operations ④. However, this practice is no longer consistent, since it constrains the words to be forced into a fixed temporal grid.

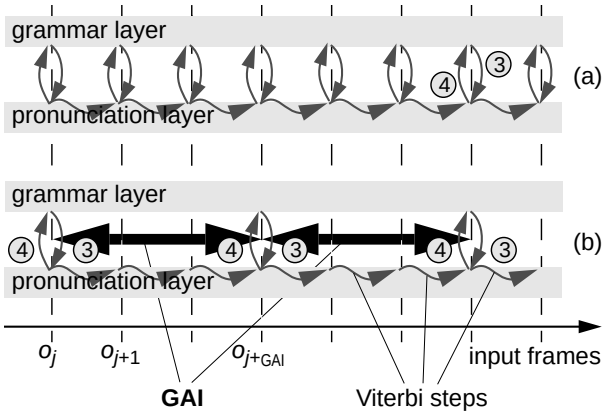

Fig. 12: Scheme of semantic decoding without (a) and with (b) discontinuous grammar activity. The *grammar activity interval* (GAI) indicates the number of Viterbi steps (input frames) without grammar activity.

5 INTENTION DECODING

Generally, it is not possible to use the semantic structure as application input immediately. Hence, it is necessary to transform the semantic structure into an application-specific code, denoted as *user's intention I*. For that purpose, we suggest an application-specific combination of a preprocessor and a compiler:

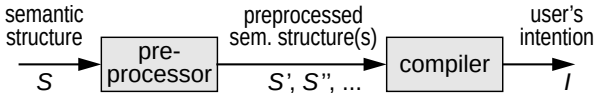

Fig. 13: Block diagram of the intention decoder

5.1 Preprocessor

The preprocessor is necessary to correct inconsistencies in the semantic structure, which occur due to the assumption that each word in the word chain has to be assigned to one single semun. This module provides

- insertion of missing and necessary information (e.g. the semun of the recently modified object),
- deletion of redundant information (e.g. all semuns after an irrelevant "garbage semun"),
- splitting – if possible – a semantic structure S into separate semantic structures S', S'', ... each stating an independent and complete command. E.g. in fig. 14, S_3 corresponding to the word chain *"create a yellow sphere and two cones"* is divided into S_3' and S_3'':

The preprocessed semantic structure(s) S', S'', ... can be seen as programs in the source language of the compiler, its output I is a program in the application-specific language. We call the latter the user's intention I, since it does not only reflect semantic and pragmatic aspects in the utterance, but it is also influenced by the present status of the application.

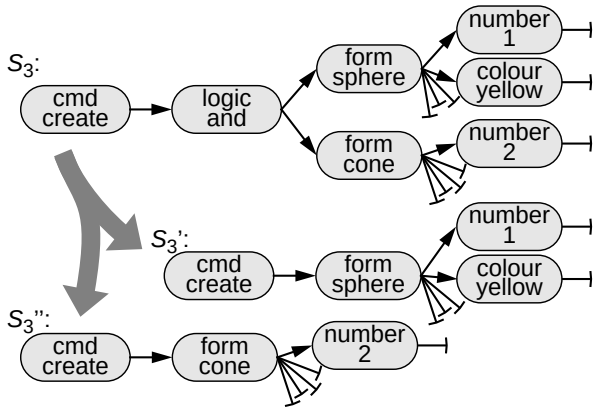

Fig. 14: The preprocessor: Splitting a semantic structure *S*.

5.2 Compiler

The task of the compiler therefore is to translate a nested semantic structure tree into a linear code within the application-specific language. Since a main part of a semantic structure's information is held in the topology of the tree, it is not possible to transform each semun individually into one block of the output code. Instead, a *top-up* approach is introduced, by propagating context knowledge down into the tree from the root to the leaves and subsequently collecting the information required for generating the intention in the reverse way, see [4][5] and fig. 15. Note the three main steps 'DESCEND' for descending the semantic structure, 'CONSTRUCT' for constructing the linearized commands and 'FINISH' for terminating.

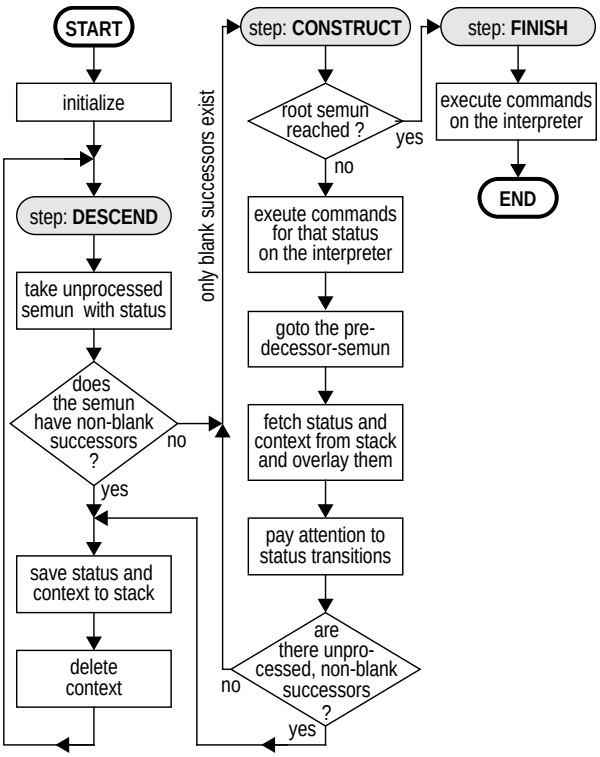

Fig. 15: Flowchart for linearizing the basic command tree [5]

The generated intention *I* for the semantic structure S_3' is shown in the following table. The commands for specifying colour (as *RGB*-components), form, and quantity as well as the insertion into the database can be easily seen:

set r 255	setstring form sphere
set g 255	set num 1
set b 0	insert

Table 1: Intention *I* for the semantic structure S_3' of fig. 14

6 LANGUAGE PRODUCTION

6.1 Word Chain Generator

The word chain generator converts a semantic structure into the corresponding word chain of the target language. For this purpose, we make use of the generative power of our syntactic models by creating just the most likely word chain given a certain semantic structure. In fact, we utilize the same algorithms as in the semantic decoder! Each semun s_n corresponds to a SM $A(s_n)$, which is according to the given semantic structure S_E hierarchically connected with other SMs to a syntactic network as explained in chap. 4.1.2. As in the decoding process, the transitions within the syntactic network affect the word alignment, the emissions affect the respective word choice. $P(W|S)$ is calculated as the product over all path and emission probabilities along a certain path through the entire syntactic network, see also eq. (22):

$$P(W|S) = \prod_{n=1}^{N} (a_n \cdot b_n \cdot c_n) \qquad (23)$$

Unlike the word chain generator in the 'top-down' semantic decoder (see fig. 6 – described in [24]), which has to produce many word chain hypotheses, this one delivers only the most probable word chain W_g given the semantic structure S_E. The concerning syntactic model considers that word chain W_g, which maximizes eq. (23):

$$W_g = \underset{W}{\operatorname{argmax}}\, P(W|S_E) \qquad (24)$$

Fig. 16 depicts one selected path through such a syntactic network, consisting of four SMs.

Since the semantic structure does not contain any grammatical information, case, number, and gender of words in W_g may be wrong. Nevertheless, W_g is usually comprehensible by humans. According to eq. (24), a German word chain

W_g: *"erzeuge zwei rot kugel"*

(*"create two red sphere"*) might be generated. In this case, the emission probabilities for the singular words *"rot"* and *"kugel"* is according to the syntactic model higher than those for the correct plural words *"rote"* and *"kugeln"*.

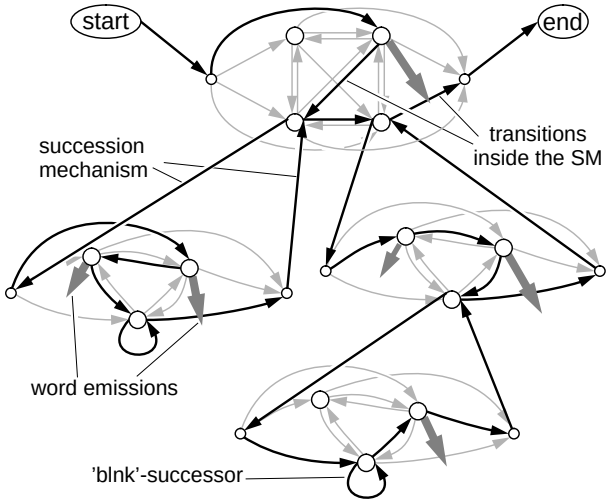

Fig. 16: Origination of a word chain along a certain path by transitions and emissions within the syntactic network [21]

Because choice (i.e. the uninflected word) and alignment of the generated words in W_g are fixed by the transitions and the emissions of the syntactic model and cannot be re-changed in a later stage, it is advisable to use manually created instead of trained syntactic knowledge bases for the target language.

6.2 Linguistic Post-Processing

The post-processing module converts the grammatically wrong word chain W_g into a correct word chain W_{opt} by changing the inflection of some words using the following knowledge bases:

- The grammar rules specify the grammatical features (case, number, gender) of a certain word by the help of both the word chain W_g and the semantic structure S_E.
- The inflectional model delivers the correct inflection of a word given its grammatical features.

The semantic structure S_E is recursively examined semun by semun. If the affiliated word of the current semun is a noun, an adjective, or an article, the correct inflection has to be found according to the respective grammatical features. The gender is extracted depending on the emitted noun, but case and number are extracted depending only on the semantic structure S_E (which is a different and new approach compared to classic linguistics). As an example, we look at following word chain W_g and semantic structure S_E:

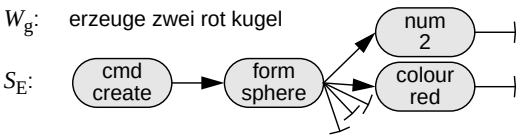

Fig. 17: Word chain W_g and semantic structure S_E

- The semun '(cmd create)' corresponds to the word *"erzeuge"*. As a verb, it remains unchanged.
- The semun '(form sphere)' corresponds to the word *"kugel"*. As a noun, the grammatical features are determined as follows:
- Since the predecessor-semun is '(cmd create)', the case is accusative. Since the first successor-semun is '(num 2)', the number is plural. Since the emitted noun is *"kugel"*, the gender is feminine.
- The inflectional model delivers for the word *"kugel"* with the grammatical features "accusative, plural, feminine" the correct word *"kugeln"*.
- The semun '(num 2)' corresponds to the word *"zwei"*. As a number, it remains unchanged.
- The semun '(colour red)' corresponds to the word *"rot"*. As an adjective, the grammatical features are determined as follows:
- Since the predecessor-semun is '(form sphere)', case, number, and gender are identical to the corresponding word of the predecessor-semun.
- The inflectional model delivers for the word *"rot"* with the grammatical features "accusative, plural, feminine" the correct word *"rote"*.

After gone through the whole semantic structure, the optimized word chain W_{opt} with the correct inflections can be composed as

W_{opt}: *"erzeuge zwei rote kugeln"*

(*"create two red spheres"*). Note, that neither the choice nor the alignment of the words is changed during the postprocessing [14].

7 IMPLEMENTED DOMAINS

Due to the separation of domain-specific and domain-independent knowledge, we attained an easy portability to various domains. Up to now, we developed semantic structures for four different applications:

- **NASGRA** (natural speech understanding graphic editor) for creating, modifying or deleting three-dimensional objects on the screen (understanding German and Slovenian as well as translation into German, English, French, Slovenian) [14][23][24]. An online version of NASGRA for natural German text input is available on WWW. Please be aware of OOV errors: http://www.mmk.e-technik.tu-muenchen.de/~mue/nasgra/
- **ROMAN**[3] (roving manipulator), speech understanding service robot for doing jobs like fetching and bringing things (understanding German) [6][7].

3) In collaboration with the Institute of Automation Control Engineering, Munich University of Technology, Germany.

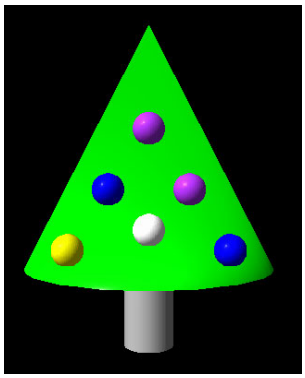

Fig. 18: Typical scenario from the NASGRA domain

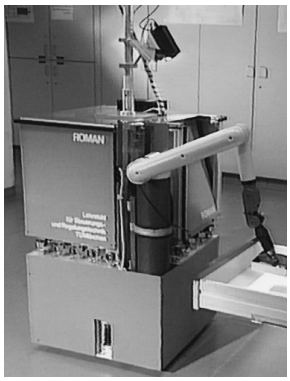

Fig. 19: Speech-understanding service robot ROMAN

- **invitoVR**[4] (interactive visualization tool for virtual reality), speech interaction within a virtual environment for medical image visualization (understanding German) [11] and
- **TERMINator** (*Termin* is the German word for *date* or *appointment*) for the translation of spoken scheduling dialogues from German into English and Greek – in principle, the domain is similar to that one of the project VERBMOBIL [27].

8 RESULTS

8.1 Semantic Decoding

As with HMM speech recognizers, the semantic accuracy stands in direct concurrence with memory and computing effort. A very sensitive parameter is the pruning-offset, which influences the beam width within the search process. Fig. 20 clearly shows the trade-off between the semantic accuracy and the search effort for our one-stage decoder. Maximum accuracy is gained by choosing a pruning offset of 200, taking on average about 22 secs of CPU workload (on a SUN™-*UltraSPARC*™, 168 MHz) for semantic decoding.

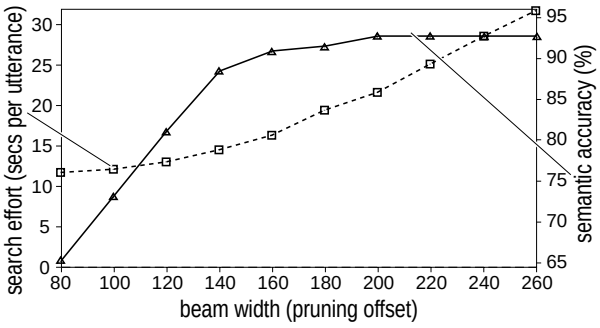

Fig. 20: System performance depending on beam width [23]

4) In collaboration with the Institute of Medical Informatics and Health Services Research, National Research Center for Environment and Health (GSF), Neuherberg, Germany.

Another important parameter to control the search process is the grammar activity interval (GAI), which has been explained in chap. 4.2.4. Search effort can be halved by setting GAI = 8 with a relatively small loss of accuracy. Hence, the discontinuous grammar activity has proven to be an effective method to improve the performance of our semantic decoder as well as of all speech processing systems based on probabilistic models.

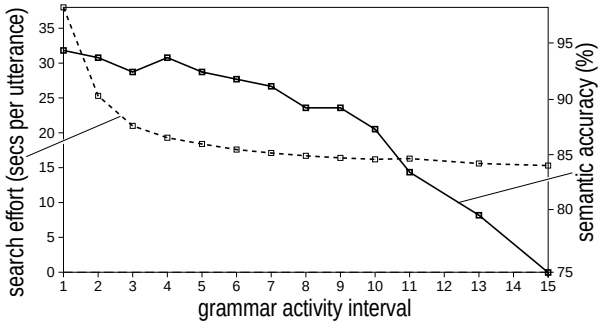

Fig. 21: System performance depending on the GAI [24]

8.2 Intention Decoding

The intention decoder has been tested with 1843 semantic structures of spoken utterances, which have been collected from 33 subjects during a Wizard-of-Oz simulation within the NASGRA domain [13]. The rate for the correct conversion of a semantic structure into its intention (application-specific code) amounts to **97.3%** [4][5].

8.3 Language Production

The language production was tested with 307 real existing semantic structures within the NASGRA domain producing four different target languages German, English, French, and Slovenian. We asked human subjects to judge the semantics (whether it is comprehensible or incomprehensible) and the syntax (whether it is correct, unusual, or wrong) for each optimized word chain W_{opt}.

target language	semantics	syntax	
	comprehensible	correct	unusual
German	95.9 %	84.4 %	5.2 %
English	94.8 %	81.1 %	10.4 %
French	93.8 %	82.4 %	9.8 %
Slovenian	88.3 %	82.1 %	8.5 %

Table 2: Language production performance [14]

The above results with an average comprehensible semantics of 93.2% and an average not-wrong syntax of 91.0% confirm that the introduced translation approach based on the semantic structure can be an alternative to other much more complex rule-based and word-based approaches, if short sentences within a limited domain should be translated.

REFERENCES

[1] J.G. Bauer, H. Stahl, J. Müller: *A One-pass Search Algorithm for Understanding Natural Spoken Time Utterances by Stochastic Models*, Proc. Eurospeech 1995 (Madrid, Spain), pp. 567-570

[2] M. Beham: *An Auditorily Based Spectral Transformation of Speech Signals*, Proc. Eurospeech 1991 (Genoa, Italy), pp. 1437-1440

[3] J. Earley: *An Efficient Context-Free Parsing Algorithm*, Comm. of the ACM, vol. 13 (1970), no. 2, pp. 94-102

[4] M. Ebersberger: *Entwurf und Implementierung eines Compiler-Interpreter-Systems für den natürlichsprachlichen Mensch-Maschine-Dialog*, diploma thesis, Institute for Human-Machine-Communication, Munich University of Technology, 1995

[5] M. Ebersberger, J. Müller, H. Stahl: *A Compiler-Interpreter-System for Decoding the User's Intention within a Speech Understanding Application*, Proc. KI-96 (Dresden, Germany, 1996), in: Lecture Notes in Artificial Intelligence 1137, Springer, pp. 61-65

[6] C. Fischer, P. Havel, G. Schmidt, J. Müller, H. Stahl, M. Lang: *Kommandierung eines Serviceroboters mit natürlicher, gesprochener Sprache*, in G. Schmid, F. Freyberger (ed.): Proc. "Autonome Mobile Systeme 1996" (Munich, Germany), Springer „Informatik aktuell“, 1996, pp. 248-261

[7] C. Fischer, M. Buss, G. Schmidt: *Hierarchical Supervisory Control of Service Robot Using HuMan-Robot-Interface*, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Osaka, Japan), 1996

[8] H. Haugeneder, H. Trost: *Beschreibungsformalismen für sprachliches Wissen*, in G. Görz (ed.): „Einführung in die künstliche Intelligenz“, Addison-Wesley Publishing, Bonn, 1993, pp. 372-424

[9] X.D. Huang, Y. Ariki, M.A. Jack: *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990

[10] R.A. Kowalski: *Logic for Problem Solving*, Elsevier North Holland, New York, 1979

[11] C. Krapichler, M. Haubner, A. Lösch, K. Englmeier: *A Human-Machine Interface for Medical Image Analysis and Visualization in Virtual Environments*, Proc. ICASSP 1997 (Munich, Germany), pp. 2613-2616

[12] J. Müller, H. Stahl: *Stochastic Modelling of Syntax and Semantics*, Proc. KI 1995 Activities: Workshops, Posters, Demos (Bielefeld), pp. 229-230

[13] J. Müller, H. Stahl: *Collecting and Analyzing Spoken Utterances for a Speech Controlled Application*, Proc. Eurospeech 1995 (Madrid, Spain), pp. 1437-1440

[14] J. Müller, H. Stahl, M. Lang: *Automatic Speech Translation Based on the Semantic Structure*, Proc. ICSLP 1996 (Philadelphia, USA), pp. 658-661

[15] J. Müller: *Die semantische Gliederung zur Repräsentation des Bedeutungsinhalts innerhalb sprachverstehender Systeme*, Ph.D. thesis, Herbert Utz Verlag, Munich, 1997

[16] H. Ney, R. Haeb-Umbach, B.H. Tran, M. Oerder: *Improvements in Beam Search for 1000-Word Continuous Speech Recognition*, Proc. ICASSP 1992 (San Francisco, USA), Bd. I, pp. 9-12

[17] A. Paeseler: *Modification of Earley's Algorithm for Speech Recognition*, Proc. NATO ASI, vol. F 46, Springer, 1988, pp. 465-472

[18] R. Pieraccini, E. Levin: *Learning how to Understand Language*, Proc. Eurospeech 1993 (Berlin, Germany), pp. 1407-1412

[19] L.R. Rabiner: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proc. IEEE, vol. 77 (1989), no. 2, pp. 257-286

[20] H. Stahl, J. Müller: *An Approach to Natural Speech Understanding Based on Stochastic Models in a Hierarchical Structure*, Proc. Workshop 'Modern Modes of Man-Machine-Communication' 1994 (Maribor, Slovenia), pp.16.1-16.9

[21] H. Stahl, J. Müller: *A Stochastic Grammar for Isolated Representation of Syntactic and Semantic Knowledge*, Proc. Eurospeech 1995 (Madrid, Spain), pp. 551-554

[22] H. Stahl, J. Müller, M. Lang: *An Efficient Top-Down Parsing Algorithm for Understanding Speech by Using Stochastic Syntactic and Semantic Models*, Proc. ICASSP 1996 (Atlanta, USA), pp. 397-400

[23] H. Stahl, J. Müller, M. Lang: *Controlling Limited-Domain Applications by Probabilistic Semantic Decoding of Natural Speech*, Proc. ICASSP 1997 (Munich, Germany), pp. 1163-1166

[24] H. Stahl: *Konsistente Integration stochastischer Wissensquellen zur semantischen Decodierung gesprochener Äußerungen*, Ph.D. thesis, Herbert Utz Verlag, Munich, 1997

[25] L. Tesnière: *Éléments de syntaxe structurale*, Klincksieck, Paris, 1959

[26] A.J. Viterbi: *Error Bounds for Convolutional Codes and an Asymptotical Optimal Decoding Algorithm*, IEEE Trans. Information Theory, vol. 61 (1973), pp. 268-278

[27] W. Wahlster: *Verbmobil – Translation of Face-to-Face Dialogs*, Proc. Eurospeech 1993 (Berlin, Germany), Addenum "Opening and Plenary Sessions", pp. 29-38