Behavioral Knowledge Representation for the Understanding and Creation of Video Sequences

Michael Arens and Hans–Hellmut Nagel

Institut für Algorithmen und Kognitive Systeme,
Fakultät für Informatik der Universität Karlsruhe (TH),
76128 Karlsruhe, Germany
{arens|nagel}@iaks.uni-karlsruhe.de

Abstract. The algorithmic generation of textual descriptions of *real world* image sequences requires conceptual knowledge. The algorithmic generation of *synthetic* image sequences from textual descriptions requires conceptual knowledge, too. An explicit representation formalism for behavioral knowledge based on formal logic is presented which can be utilized in both tasks – *Understanding* and *Creation* of video sequences. Common sense knowledge is represented at various abstraction levels in a *Situation Graph Tree*. This form of representation is exploited in order to fill in missing details in a natural language text describing developments for an image sequence to be synthesized.

1 Introduction

An (artifical) *cognitive* vision system is supposed to not only extract numerical results from input images or image sequences. It rather has to associate those quantitative results obtained by machine vision with primitive qualitative concepts which then can be aggregated into more complex concepts [6, 10]. These complex concepts might then be transformed into a (natural language) textual description of what the computer vision system *recognized* in the depicted scene. In this sense, *Cognitive Vision* (CogV) constitutes an algorithmic link between visual and textual representations of information about the same discourse [16]. Examples from image sequences of innercity road traffic [9] and of vehicle maneuvers at a filling station [7] have shown that each step in this algorithmic linkage incorporates a–priori knowledge into the overall vision system.

The generation of synthetic image sequences from a (natural language) textual description of a time–varying scene might be seen as an inversion of CogV as described above: complex concepts mentioned in the input text have to be broken down into primitive concepts [16, 5, 2]. These primitive concepts have to be associated with (time–varying) geometric body configurations [16, 12, 14], which then can be visualized by standard computer graphic algorithms. The steps necessary for image sequence generation require knowledge very similar to the knowledge utilized in the CogV task. Such additional knowledge is definitely needed in

cases where certain constraining information had not been incorporated into the input text due to the fact that a human reader can assume certain global constraints utilizing his *commonsense knowledge*. A good example for such *implicit* constraints is given in [4]. Thus, a–priori knowledge can control the (probably) vast amount of possible image sequences compatible with the input text. This kind of knowledge–supported image sequence generation is referred to, therefore, as *Controlled Imagery Generation* (CIG) in the following.

Accepting that CogV and CIG are inverse processes as motivated above, the question arises wether or not knowledge represented for one process can be utilized easily in the inverse process. This would not only minimize the effort of knowledge engineering or acquisition, but also would introduce the ability of feedback loops between both processes: the conceptual or textual description of image sequences produced by a vision system might be re–visualized. Differences between the original and the synthetic image sequence point out errors or shortcomings in the knowledge deployed [17] or might even be used to re–formulate the output of the vision system in order to impose the correct '*mental image*' to a human listener, as it is named in [3]. Analogously, a synthetic image sequence generated from an input text might be analyzed by a vision system. Again, differences between the original text and the text produced by the vision system might indicate deficiencies in the utilized knowledge or associated processing steps.

In the sequel, Section 2 refines the view of a cognitive vision system as a layered system of separable transformation steps. The knowledge employed in each of these steps will be described with special emphasis on conceptual, in particular behavioral knowledge. Section 3 outlines a possible approach to CIG as an inverted CogV task, focusing on the usability of the *same* behavioral knowledge.

2 Cognitive Vision

Fig. 1 depicts a layered model of a cognitive vision system. Each white rectangle in this figure stands for one representation form of information processed by the vision system. Filled rectangles surrounding one or more layers represent more encompassing sub–systems of the overall vision system. The left lower sub–system (middle–gray) comprises layers which process quantitative information. The upper (light–gray) sub–system contains layers processing conceptual information. The right lower sub–system (dark–gray) contains only one layer (so far): here, natural language information is processed. Arrows between layers indicate the flow of information between two layers, implying appropriate transformation steps. Dashed arrows show the direction in which information is passed during a CogV task, while solid arrows indicate the opposite direction to be taken by a video creation task.

The sub–system concerned with quantitative information (middle–gray in Fig. 1, comprising SAL, ISL, PDL and SDL) obtains signals from a camera configuration, converts these signals into images, and analyzes these images with respect to information about the depicted scene. The knowledge employed in these

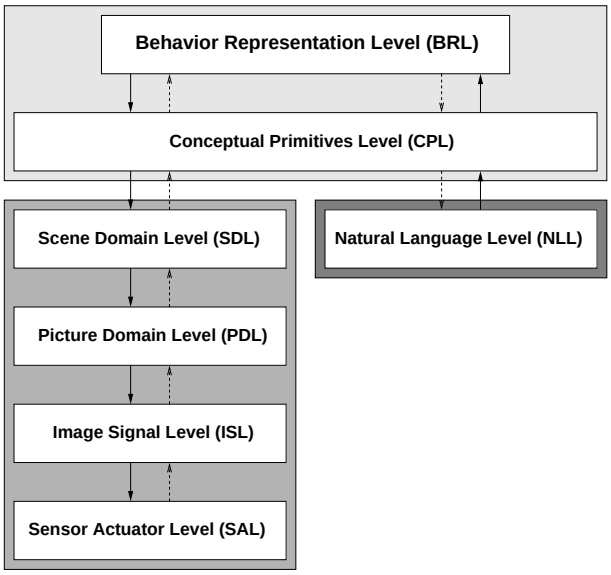

Fig. 1. Layer model of a Cognitive Vision System (adapted from [13]). Further explanation in the text.

transformation steps includes knowledge about the *meaning* of two–dimensional image cues (as edges, optical flow, etc.), but in addition knowledge concerned with the projection from 3D scenes to the two–dimensional image plane. The vision system comprises, moreover, discourse–specific knowledge: in the case of image sequences of road traffic scenes, vehicle models, motion models, illumination models, and geometric models representing the static components of the depicted scene like the lane structure or other static objects are taken into account. The results obtained by this sub–system are converted into *conceptual primitives* and are passed up to the sub–system concerned with conceptual information (light–gray in Fig. 1, comprising CPL and BRL). These conceptual primitives contain the state of each depicted agent, which the quantitative sub–system was able to *recognize*, including, e.g., the agent's position and orientation in the scene and its linear and angular velocity.

2.1 Towards primitive conceptual descriptions

Based on the state–information passed up from the quantitative sub–system, the *Conceptual Primitives Layer* (CPL) can derive more complex concepts by employing a *Fuzzy Metric Temporal Horn Logic* (FMTHL see [15]) on an a–priori given *terminology*. This terminology consists of logic–rules and facts which define from which (combination of) conceptual primitives a certain more complex concept can be derived [8]. Such complex concepts may regard the agent itself (`standing`, `moving`, `accelerating`, `turning`, etc.), the agent and another object

(`overtaking`, `following`, etc.), but also the agent and a conceptual repesentation of a certain location in the scene (`leaving`, `approaching`, etc.). The CPL therefore comprises, too, the conceptual representation of static scene objects.

2.2 Towards abstract *behavioral* descriptions

So far, the complex concepts derived by the CPL appear as temporally and conceptually isolated statements. These concepts are referred to as *occurrences*. To embed occurrences into a temporal and conceptual context, the next layer (BRL) describes the expected behavior of agents in the form of so–called *Situation Graph Trees* (SGTs).

SGTs [15, 1] describe the behavior of an agent in a hierarchical manner: the basic unit of any SGT is the *situation scheme*. Such a scheme describes the state of an agent in terms of occurrencies together with the actions the agent is expected to carry out whenever the situation scheme is instantiated. Directed edges between situation schemes (called *prediction edges*) define a temporal successor relation on situation schemes. Thus, single situation schemes are embedded into (temporal) sequences of other schemes. Single schemes can be marked as start–situation and/or end–situation of such sequences. Situation schemes together with prediction edges build so–called *situation graphs*. These situation graphs can particularize superordinated situation schemes. This is done by connecting the situation scheme to be particularized with a situation graph by so–called *particularization edges*. The situation graph is said to *particularize* the situation scheme, which means it describes the particularized situation in a conceptually or temporally more detailed way. Thus, particularization edges define a partial order on situation schemes: situation schemes inside a situation graph which do not particularize any other scheme are called '*most general situations*' (the graph is called *root graph* of the SGT). Schemes inside any graph reachable by a particularization edge from inside this root graph are *more particularized*, and so on. Circles inside the SGT due to particularization edges are not allowed. Thus, schemes inside situation graphs together with particularization edges connecting schemes with particularizing graphs build tree–like structures, the *situation graph trees* [9, 1].

In a CogV task, SGTs are utilized to recognize situations which can be instantiated for the observed agent by applying a graph traversal on a suitable SGT. For further details on this graph–traversal, see [15, 9, 1]. SGTs are incorporated into the vision system by their algorithmic transformation into FMTHL–logic programs. This allows easy modifications and even the exchange of the complete behavioral knowledge incorporated into the vision system if necessary. In addition, this formulation directly cooperates with the other conceptual knowledge sources such as the terminology and the conceptual representation of static scene objects. The complete conceptual sub–system of the vision system is based, therefore, on formal logic inference.

An example–SGT can be seen in Fig. 2. This SGT is presently used for the textual description of image sequences showing vehicle maneuvers at a filling station. Situation schemes are depicted as rectangles comprising the name

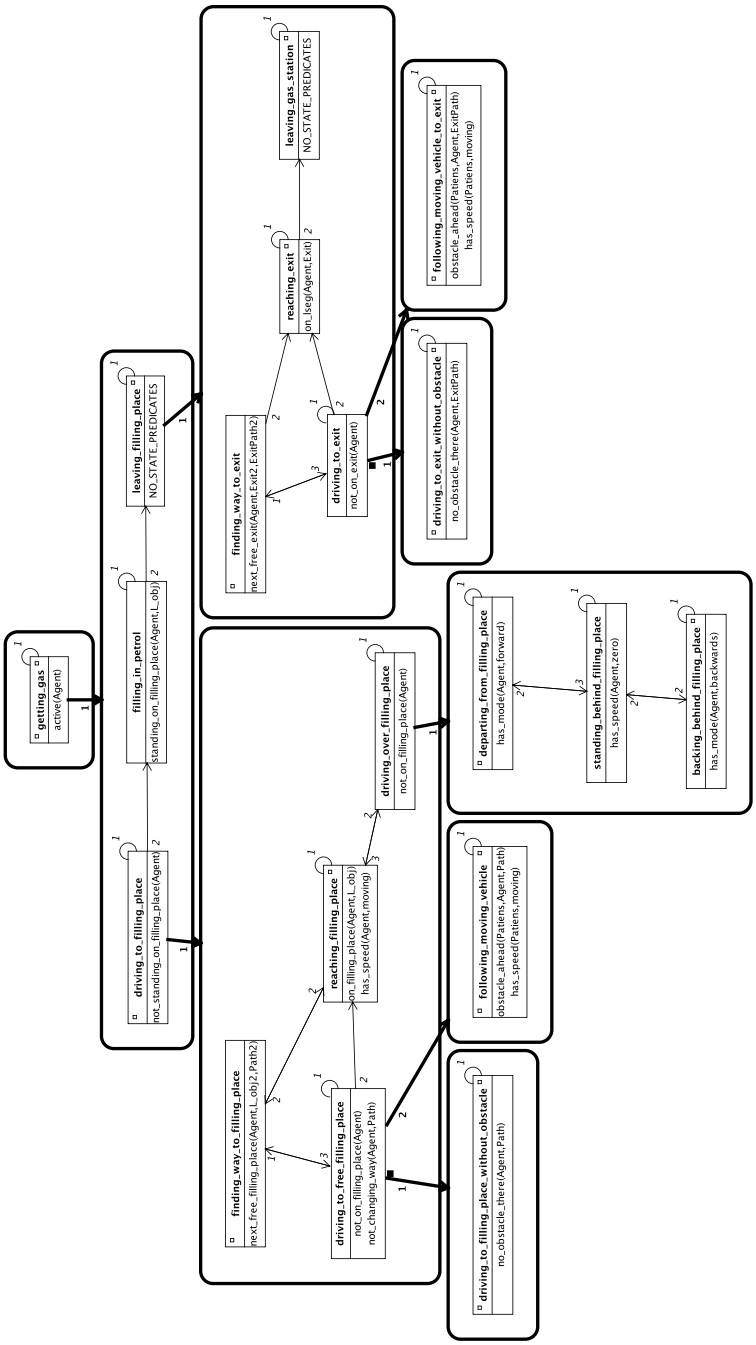

Fig. 2. (Part of an) SGT modelling the expected behavior of (vehicular) agents at a filling station. Further explanation in the text.

of the scheme and all occurrencies which have to be evaluated to instantiate this scheme (note that actions normally associated with situation schemes have been omitted in Fig. 2 due to space limitations). Situation graphs are shown as rounded rectangles. Thin arrows denote prediction edges, while thick arrows indicate particularization edges. Note that small circles in the right upper corner of situation schemes depict predictions from one scheme to itself. These predictions are called *self–predictions*. The root graph (compare Fig. 2) comprises only one situation scheme, namely getting_gas. This scheme is particularized by another situation graph, comprising three schemes (driving_to_filling_place, filling_in_petrol, and leaving_filling_place), whereby the first of these schemes is marked as start–situation and the last one as end–situation (small squares to the left or to the right of situation names indicate start–schemes or end–schemes, respectively). Two of these schemes are particularized further. Note that multiple particularizations are allowed (compare Fig. 2: driving_to_free_filling_place). If multiple particularizations or multiple predictions occur in an SGT, these edges have to be ordered (small numbers in Fig. 2 show the order between particularizations and predictions from any scheme). The traversal algorithm mentioned above tests for further scheme instantiations in this order. Note that some particularizing graphs have been omitted in Fig. 2 due to space limitations. These omissions are indicated by small filled squares beneath situation schemes (compare Fig. 2: driving_to_free_filling_place and driving_to_exit).

2.3 From behavioral descriptions to natural language text

The graph–traversal in BRL based on the employed SGT and on the occurrencies derived in the CPL instantiates one situation for each image frame and each recognized agent in the observed scene. The sequence of these situations can be used in the *Natural Language Layer* (NLL) to formulate a natural language text describing the scene as *seen* by the cognitive vision system [7]. The text generation conducted in NLL is based on the *Discourse Representation Theory* (DRT) treated by Kamp & Reyle [11].

3 Controlled Imagery Generation

The overall task in CIG is to generate *one plausible* image sequence which visualizes the scene characterized by a textual description. Considering CIG as an inverted CogV task, the first step of imagery generation includes the analysis of the input text in order to construct a conceptual description of the scene mentioned there (NLL to CPL in Fig. 1). Natural language processing (NLP) steps along the lines of the DRT as required for CIG have been reported in [2].

The next step towards the generation of a synthetic image sequence is the recognition of situations which have to be instantiated for any agent mentioned in the text (CPL to BRL in Fig. 1). Supposing that appropriate concepts were derived from the text, this instantiation of situation schemes is (at least) comparable to situation instantiation within the CogV task. It therefore is assumed

in the following that a sequence of isolated situations for every agent mentioned in the input text has been derived after these two transformation steps.

One problem arises due to the fact that natural language texts are aimed at a human reader: not every fact has to be included into the text, because the reader is supposed to *assume* certain global facts on the basis of his commonsense knowledge. A text describing a vehicle at a filling station as *reaching* a certain filling place (reaching_filling_place in Fig. 2), would surely make the reader assume that this vehicle was driving towards that (previously vacant) filling place for some preceding time instances (see driving_to_free_filling_place in Fig. 2). The reader can obviously *embed* single situations into *plausible* situation sequences.

In addition to the above–mentioned problem of situation–embedding, a second type of potentially incomplete specification arises in association with most natural language texts: usually, a human observer describing, e.g., vehicle maneuvers at a filling station would formulate at a certain level of detail what he sees. For some purposes, it might be sufficient to describe a vehicle as driving to a filling place (driving_to_filling_place), without stating that the vehicle (i.e. the driver) first decides for a vacant filling place (finding_way_to_free_filling_place), drives towards it (driving_to_free_filling_place), and at last reaches that place (reaching_filling_place). Again, the human reader can be supposed to assume this more detailed description of what really happened in the scene on the basis of his commonsense knowledge.

3.1 Derivation of plausible behaviors from situation sequences

A single algorithm for both task – the embedding of situations and the derivation of *most detailed* descriptions – can draw advantage from the fact that the knowledge necessary for both task is already stored in form of SGTs: on one hand, every situation recognized in the input text will appear inside a situation graph, upon a path from some start–situation to some end–situation. Moreover, this situation graph might particularize another, more general situation: again, this particularized situation is not isolated, but may imply an entire sequence of situations, and so on. Thus, the embedding of given situations can be derived.

On the other hand, most situations might be superordinated to particularizing graphs. Each path from a start–situation to an end–situation in such a particularizing graph comprises a more detailed description of the initial situation. Situations occurring in that path might again be particularized by other graphs, and so on. Therefore, the particularization of given situations is also derivable. These relationships will be characterized more precisely in the following six definitions, prior to the description of an algorithm which solves both problems mentioned above.

Definition 1 (SGT–episode). *Any sequence E of situations inside one situation graph G which constitutes a path from any start–situation to any end–situation along prediction edges is denoted as an* SGT–episode *described by G. If G is particularizing a situation scheme* S*, E is called* particularizing SGT–episode *of* S*.*

Definition 2 (behavior). *Given an SGT $\mathcal{B}$, a* behavior *is recursively defined as follows:*

- *each SGT–episode E described by the root graph of $\mathcal{B}$ is a* behavior.
- *Given a behavior E, another behavior can be obtained by replacing one situation* `S` *in E with a particularizing SGT–episode of* `S`.

Definition 3 (particularizing behavior). *Given an SGT $\mathcal{B}$ and a behavior E, any behavior E' is denoted as a* particularizing behavior *of E, iff E' can be obtained by (recursively) replacing situations in E with particularizing SGT–episodes.*

Definition 4 (maximal behavior). *Given an SGT $\mathcal{B}$, any behavior E is called a* maximal behavior*, iff no situation* `S` *in E can be replaced by a particularizing SGT–episode of* `S`.

Definition 5 ($\mathcal{B}$–compatible behavior). *Given an SGT $\mathcal{B}$ and a situation sequence S, any behavior $\widehat{S}$ is called* $\mathcal{B}$–compatible *with S, iff*

- *each element of S is also element of $\widehat{S}$ and*
- *for each pair (s_i, s_j) of elements of S, where s_i lies before s_j in the sequence S, s_i lies before s_j in the sequence $\widehat{S}$.*

Definition 6 (maximized $\mathcal{B}$–compatible behavior). *Given an SGT $\mathcal{B}$ and a situation sequence S, any behavior M is called* maximized $\mathcal{B}$–compatible *with regard to S, iff*

- *M is a maximal behavior and*
- *M is a particularizing behavior of some behavior $\widehat{S}$ which is $\mathcal{B}$–compatible with S.*

The algorithmic solution for situation–embedding and situation–particularization takes an SGT $\mathcal{B}$ and a situation sequence S as input. It derives the desired *maximized $\mathcal{B}$–compatible behavior* in two recursive sub–processes: first, a $\mathcal{B}$–compatible behavior is searched for the given input sequence (compare Def. 5). This sub–process embeds the given situation sequence into a possible behavior as described by $\mathcal{B}$. Then, this $\mathcal{B}$–compatible behavior will be particularized until a maximized $\mathcal{B}$–compatible behavior (compare Def. 6) is found. This second sub–process thus derives a most–detailled description of the desired behavior.

To prepare the first recursion, the algorithm determines an *abstraction path* for each situation in S. These paths lead from the given situation up to a situation inside the root graph of $\mathcal{B}$. The paths are unique, because every situation can only be part of one graph and each graph can only particularize one or – in case of the root graph of $\mathcal{B}$ – no situation. The abstraction paths are used in the first recursion to restrict the search space for $\mathcal{B}$–compatible behaviors. The initial behavior S_0 for this search is given by any SGT–episode (compare Def. 1) inside the root graph (as an initial behavior following Def. 2), which contains all *last* elements of the abstraction paths in correct order. The last elements

Input
- SGT $\mathcal{B}$, situation sequence S

Preparation
- determine abstraction path for each situation in S
- build list L_0 of last elements of those paths in the order given by S
- **determine SGT–episode** S_0 in root–graph containing all elements of L_0
- delete last elements of abstraction paths
- continue with **Recursion 1**

Recursion 1 /∗ derive $\mathcal{B}$–compatible behavior ∗/
- build list L_i of last elements of abstraction paths
- determine graph membership partition P_i of L_i
- **determine list** E_i **of SGT–episodes** for each element of P_i
- **replace** situations in S_{i-1} with particularizing SGT–episode from E_i
- delete last elements of abstraction paths
- all abstraction paths empty ?

YES: ∗ continue with **Recursion 2**
NO: ∗ goto **Recursion 1**

Recursion 2 /∗ extend to maximized $\mathcal{B}$–compatible behavior ∗/
- **determine particularizing SGT–episode** for one situation in S_i
- SGT–episode found ?

YES: ∗ create S_{i+1} by substitution of SGT–episode for situation in S_i
∗ goto **Recursion 2**
NO: ∗ continue with **Output**

Output
- set $M = S_i$, return M

Fig. 3. The algorithm deriving plausible situation sequences in pseudo-code. Steps of the algorithm which might lead to multiple solutions are marked in **bold face**.

of all abstraction paths are then deleted by the algorithm. The first recursive sub–process then particularizes the obtained initial behavior (compare Def. 3). Again, the remaining abstraction paths restrict this particularization: only those particularizing behaviors are investigated which contain the last elements of these abstraction paths. Therefore, a list of those last elements is constructed. This list is partitioned according to the graph membership of comprised consecutive situations. This is necessary for the following reason: in distinction to the initial step, where all last elements of the abstraction paths could be found within the root graph of $\mathcal{B}$, subsequent steps might lead to lists of last elements which lie in different graphs. If two or more consecutive situations lie within one graph, only one SGT–episode in that graph is searched which contains all these situations in the correct order. Thus, for each partition of the current last elements of the abstraction paths, one SGT–episode is searched for particularization. In each recursion step i, the algorithm constructs a particularizing behavior S_i from the result S_{i-1} obtained during the preceding recursion step. The last elements of each (not empty) abstraction path are deleted until all these paths are empty.

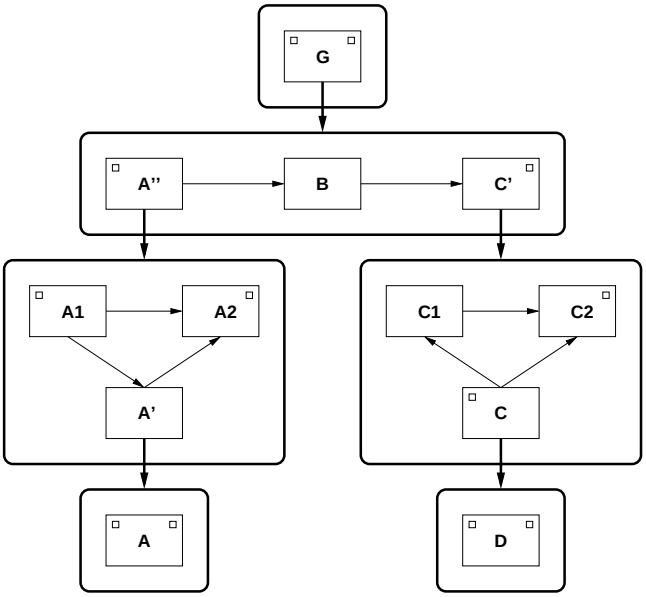

Fig. 4. Example SGT. Situation schemes are depicted as rectangles, situation graphs are shown as rectangles with rounded corners. Small squares in the left (right) upper corner of situation schemes mark them as start–situations (end–situations). Prediction edges are depicted as thin, particularization edges as thick arrows.

The resulting behavior is – due to its construction – $\mathcal{B}$*–compatible* with the initially given sequence S.

The second recursive sub–process of the algorithm takes the previously constructed $\mathcal{B}$–compatible behavior and particularizes this behavior until the resulting behavior is maximal (following Def. 6).

Note that the algorithm described above does not necessarily yield a unique solution. Depending on the SGT and the input situation sequence, the algorithm rather constructs *all possible* maximized $\mathcal{B}$–compatible behaviors with respect to the SGT and the input sequence. The order in which these possible solutions are obtained by the algorithm is defined by the order on prediction edges and particularization edges mentioned in Section 2.2. The following example will illustrate the application of the algorithm and the construction of multiple possible solutions, see Fig. 3 for a pseudo–code formulation of the algorithm outlined above.

3.2 An example

Suppose we describe the possible behavior of agents with the SGT $\mathcal{B}$ depicted in Fig. 4. Suppose further that the text–analysis and subsequent situation instantiation yielded the initial situation sequence $S = \langle \texttt{A,B,C} \rangle$. Thus the following

three abstraction paths result:

$$g_A = \langle \mathtt{A}, \mathtt{A}', \mathtt{A}'', \mathtt{G} \rangle,$$
$$g_B = \langle \mathtt{B}, \mathtt{G} \rangle,$$
$$g_C = \langle \mathtt{C}, \mathtt{C}', \mathtt{G} \rangle$$

according to the three situations of S. Notice that the last element of each path (i.e. $\mathtt{G}$) lies within the root graph of $\mathcal{B}$. Thus, building the list of last elements of all abstraction paths (whereby multiple appearances of the same situation are kept only once) the algorithm produces

$$L_0 = \langle \mathtt{G} \rangle$$

as the list of last elements. Due to the construction of abstraction paths, all elements of L_0 (only $\mathtt{G}$ in this example) lie within the root graph of $\mathcal{B}$. An SGT–episode in this graph is searched, therefore, which contains all elements of L_0 in the order given there, yielding simply

$$S_0 = \langle \mathtt{G} \rangle.$$

We thus proceed with the sequence $S_0 = \langle \mathtt{G} \rangle$. Because all last elements of the abstraction paths have been processed, these elements are deleted. The remaining paths have the form

$$g_A = \langle \mathtt{A}, \mathtt{A}', \mathtt{A}'' \rangle,$$
$$g_B = \langle \mathtt{B} \rangle,$$
$$g_C = \langle \mathtt{C}, \mathtt{C}' \rangle.$$

The first recursion starts with the determination of the list of last elements of the remaining abstraction paths: This yields

$$L_1 = \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle.$$

Partitioning according to graph membership of comprised situations leads to

$$P_1 = \langle \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle \rangle,$$

because all three situations belong to one graph. The search for SGT–episodes in that graph containing all three situations in the correct order produces a single tuple

$$E_1 = \langle \langle \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle \rangle \rangle.$$

We thus obtain one particularizing behavior, namely by replacing the situation $\mathtt{G}$ in S_0 by the sequence $\langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle$ and by deleting every last element in each abstraction path, yielding $S_1 = \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle$ as new behavior and

$$g_A = \langle \mathtt{A}, \mathtt{A}' \rangle,$$
$$g_C = \langle \mathtt{C} \rangle$$

as remaining paths. Notice that the path g_B initially determined for the situation $\mathtt{B}$ is empty now and no longer affects the processing. The next iteration starts with

$$L_2 = \langle \mathtt{A}', \mathtt{C} \rangle$$

as list of last elements and

$$P_2 = \langle\langle \mathtt{A}' \rangle, \langle \mathtt{C} \rangle\rangle$$

as the partition of that list. In this case, we obtain a partition comprising two elements, one containing elements of one graph on the abstraction path initially determined for A, the other containing elements of a graph on the abstraction path for C. Thus, determining possible SGT–episodes (compare Def. 1) in both of these graphs yields

$$E_2 = \langle\langle\langle \mathtt{A1}, \mathtt{A}', \mathtt{A2} \rangle\rangle, \langle\langle \mathtt{C}, \mathtt{C2} \rangle, \langle \mathtt{C}, \mathtt{C1}, \mathtt{C2} \rangle\rangle\rangle.$$

The first element of E_2 contains only one SGT–episode $\langle \mathtt{A1,A',A2} \rangle$ because this is the only SGT–episode in the graph containing $\mathtt{A}'$. The second element of E_2 contains two SGT–episodes: in the corresponding graph, two SGT–episodes can be found which both comprise the situation C (compare Fig. 4). We thus obtain two possible next particularizing behaviors

$$S_2 = \langle \mathtt{A1}, \mathtt{A}', \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C2} \rangle,$$
$$S_2' = \langle \mathtt{A1}, \mathtt{A}', \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C1}, \mathtt{C2} \rangle$$

by replacing the situations $\mathtt{A}''$ and $\mathtt{C}'$ in S_1 by the possible paths from E_2. The only remaining abstraction path after deletion then is

$$g_A = \langle \mathtt{A} \rangle.$$

A last iteration yields the list of last elements

$$L_3 = \langle \mathtt{A} \rangle,$$

the partition

$$P_3 = \langle\langle \mathtt{A} \rangle\rangle,$$

and resulting possible SGT–episodes

$$E_3 = \langle\langle\langle \mathtt{A} \rangle\rangle\rangle.$$

The only possible SGT–episode for replacing $\mathtt{A}'$ in S_2 and S_2' is therefore $\langle \mathtt{A} \rangle$, yielding the particularizing behaviors

$$S_3 = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C2} \rangle$$
$$S_3' = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C1}, \mathtt{C2} \rangle.$$

Because all abstraction paths are empty after this step, the first sub–process of the algorithm terminates here. Notice that both situation sequences S_3 and S_3' are $\mathcal{B}$–compatible behaviors with respect to S. Nevertheless, the two sequences S_3 and S_3' do not describe the behavior of the mentioned agent in the most detailed way obtainable by $\mathcal{B}$: the situation C as part of both behaviors possesses a particularizing graph – consisting only of the situation scheme D (compare Fig. 4). The most particularized description is obtained, however, by the second sub–process of the algorithm: the $\mathcal{B}$–compatible behaviors obtained so far are

Input: $\mathcal{B}$, $S = \langle \mathtt{A}, \mathtt{B}, \mathtt{C} \rangle$	
$g_A = \langle \mathtt{A}, \mathtt{A}', \mathtt{A}'', \mathtt{G} \rangle$ $g_B = \langle \mathtt{B}, \mathtt{G} \rangle$ $g_C = \langle \mathtt{C}, \mathtt{C}', \mathtt{G} \rangle$ $L_0 = \langle \mathtt{G} \rangle$ $S_0 = \langle \underline{\mathtt{G}} \rangle$ $g_A = \langle \mathtt{A}, \mathtt{A}', \mathtt{A}'' \rangle$ $g_B = \langle \mathtt{B} \rangle$ $g_C = \langle \mathtt{C}, \mathtt{C}' \rangle$	
$L_1 = \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle \Rightarrow P_1 = \langle \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle \rangle \Rightarrow E_1 = \langle \langle \langle \mathtt{A}'', \mathtt{B}, \mathtt{C}' \rangle \rangle \rangle$ $S_1 = \langle \underline{\mathtt{A}'', \mathtt{B}, \mathtt{C}'} \rangle$ $g_A = \langle \mathtt{A}, \mathtt{A}' \rangle$ $g_B = \langle\ \rangle\surd$ $g_C = \langle \mathtt{C} \rangle$	
$L_2 = \langle \mathtt{A}', \mathtt{C} \rangle \Rightarrow P_2 = \langle \langle \mathtt{A}' \rangle, \langle \mathtt{C} \rangle \rangle \Rightarrow E_2 = \langle \langle \langle \mathtt{A1}, \mathtt{A}', \mathtt{A2} \rangle \rangle, \langle \langle \mathtt{C}, \mathtt{C2} \rangle, \langle \mathtt{C}, \mathtt{C1}, \mathtt{C2} \rangle \rangle \rangle$	
$S_2 = \langle \underline{\mathtt{A1}, \mathtt{A}', \mathtt{A2}}, \mathtt{B}, \underline{\mathtt{C}, \mathtt{C2}} \rangle$ $g_A = \langle \mathtt{A} \rangle$ $g_B = \langle\ \rangle\surd$ $g_C = \langle\ \rangle\surd$	$S_2' = \langle \underline{\mathtt{A1}, \mathtt{A}', \mathtt{A2}}, \mathtt{B}, \underline{\mathtt{C}, \mathtt{C1}, \mathtt{C2}} \rangle$ $g_A = \langle \mathtt{A} \rangle$ $g_B = \langle\ \rangle\surd$ $g_C = \langle\ \rangle\surd$
$L_3 = \langle \mathtt{A} \rangle \Rightarrow P_3 = \langle \langle \mathtt{A} \rangle \rangle \Rightarrow E_3 = \langle \langle \langle \mathtt{A} \rangle \rangle \rangle$ $S_3 = \langle \mathtt{A1}, \underline{\mathtt{A}}, \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C2} \rangle$ $g_A = \langle\ \rangle\surd$ $g_B = \langle\ \rangle\surd$ $g_C = \langle\ \rangle\surd$	$L_3 = \langle \mathtt{A} \rangle \Rightarrow P_3 = \langle \langle \mathtt{A} \rangle \rangle \Rightarrow E_3 = \langle \langle \langle \mathtt{A} \rangle \rangle \rangle$ $S_3 = \langle \mathtt{A1}, \underline{\mathtt{A}}, \mathtt{A2}, \mathtt{B}, \mathtt{C}, \mathtt{C1}, \mathtt{C2} \rangle$ $g_A = \langle\ \rangle\surd$ $g_B = \langle\ \rangle\surd$ $g_C = \langle\ \rangle\surd$
$M = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \underline{\mathtt{D}}, \mathtt{C2} \rangle$	$M = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \underline{\mathtt{D}}, \mathtt{C1}, \mathtt{C2} \rangle$
Output: $M = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{D}, \mathtt{C2} \rangle$	Output: $M = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{D}, \mathtt{C1}, \mathtt{C2} \rangle$

Table 1. Example for the computation of plausible situation sequences. For explanations see the text in Section 3.2.

particularized until a maximal $\mathcal{B}$–compatible behavior is reached. In the example above, only one situation (i.e. C) comprised in both behaviors possesses a particularizing graph. In that graph, only one SGT–episode $\langle \mathtt{D} \rangle$ can be found. Thus, for the example discussed above, this operation yields the final result(s)

$$M = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{D}, \mathtt{C2} \rangle$$
$$M' = \langle \mathtt{A1}, \mathtt{A}, \mathtt{A2}, \mathtt{B}, \mathtt{D}, \mathtt{C1}, \mathtt{C2} \rangle.$$

Table 1 recapitulates the example described above.

3.3 From maximal $\mathcal{B}$–compatible behavior to synthetic image sequences

Of course, the task of CIG is not finished by deriving plausible situation sequences in BRL (compare Fig. 1). The next step would be to transform these situations first into complex and then into primitive concepts in CPL. Subsequently, these primitive concepts have to be transformed into quantitative descriptions

of body configurations in the described 3D-scene (SDL). These configurations then have to be visualized.

4 Conclusion

In this article, a view of *Cognitive Vision* and *Controlled Imagery Generation* as inverse processes has been motivated. Moreover, it was suggested to use *one* knowledge base for *both* processes in order to minimize knowledge engineering efforts and in order to allow feedback loops between both processes. Special emphasis was placed on the representation of *behavioral* knowledge. Here it has been shown that SGTs – behavioral representation formalisms developed for Cognitive Vision – can also be used for the derivation of plausible situation sequences as a starting point for the generation of synthetic video sequences.

An algorithm was explained which derives plausible situation sequences by not only particularizing given information – as described, e.g., in [16, 17] – but also incorporates knowledge about plausible behavioral contexts into which the given information can be embedded.

5 Future Work

The algorithmic determination of plausible situation sequences presented in this article is only one step of the process of Controlled Imagery Generation. The extraction of concepts from an input text and their association with situation schemes in a given SGT still will profit from further investigations.

The same knowledge representation formalisms employed in the CogV task will be used to facilitate the transformation of situation schemes into conceptual primitives, including the terminology and a conceptual representation of static scene objects. Further steps will have to transform these conceptual primitives into numerical representations, using again the same quantative knowledge as employed in the CogV task, e.g., the illumination model and motion models for vehicles.

References

1. M. Arens and H.–H. Nagel: *Representation of Behavioral Knowledge for Planning and Plan–Recognition in a Cognitive Vision System.* In: M. Jarke et al. (Eds.): Proc. of the 25th German Conf. on Artificial Intelligence (KI-2002), 16–20 September 2002, Aachen, Germany. LNAI 2479, Springer-Verlag: Berlin·Heidelberg·New York/NY 2002, pp. 268-282.
2. M. Arens, A. Ottlik, and H.–H. Nagel: *Natural Language Texts for a Cognitive Vision System.* In: F. van Harmelen (Ed.): Proc. of the 15th European Conf. on Artificial Intelligence (ECAI–2002), 21–26 July 2002, Lyon, France, IOS Press: Amsterdam 2002, pp. 455–459.

3. A. Blocher and J. R. J. Schirra: *Optional Deep Case Filling and Focus Control with Mental Images: ANTLIMA–KOREF.* In: C. S. Mellish (Ed.): Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI–95), Montréal, Canada, 20–25 August 1995, pp. 417–423.
4. B. Coyne and R. Sproat: *WordsEye: An Automatic Text–to–Scene Conversion System.* In: Proc. of the 28th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH 2001), 12–17 August 2001, Los Angeles, CA, pp. 487–496.
5. A. Egges, A. Nijholt, and P. Nugues: *Generating a 3D Simulation of a Car Accident from a Formal Description: the CarSim System.* In: V. Giagourta and M. G. Strintzis (Eds.): Proc. of the Int. Conf. on Augmented, Virtual Environments and Three–Dimensional Imaging (ICAV3D), Mykonos, Greece, May 2001, pp. 220–223.
6. J. Fernyhough, A. G. Cohn, and D. C. Hogg: *Constructing qualitative event models automatically from video input.* Image and Vision Computing **18**:2 (2000) 81–103.
7. R. Gerber: *Natürlichsprachliche Beschreibung von Straßenverkehrsszenen durch Bildfolgenauswertung.* Dissertation, Fakultät für Informatik der Universität Karlsruhe (TH), Januar 2000; see: http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=2000/informatik/8 (in German).
8. R. Gerber and H.–H. Nagel: *Occurrence Extraction from Image Sequences of Road Traffic Scenes.* In: L. van Gool and B. Schiele (Eds.): Proc. of the Workshop on Cognitive Vision, 19–20 September 2002, ETH Zurich, Switzerland; pp. 1–8, see: http://www.vision.ethz.ch/cogvis02/finalpapers/gerber.pdf .
9. M. Haag and H.-H. Nagel: *Incremental Recognition of Traffic Situations from Video Image Sequences.* Image and Vision Computing **18**:2 (2000) 137–153.
10. R. J. Howarth and H. Buxton: *Conceptual descriptions from monitoring and watching image sequences.* Image and Vision Computing **18**:2 (2000) 105–135.
11. H. Kamp and U. Reyle: *From Discourse to Logic.* Kluwer Academic Publishers: Dordrecht, The Netherlands 1993.
12. A. Mukerjee, K. Gupta, S. Nautiyal, M.P. Singh, and N. Mishra: *Conceptual description of visual scenes from linguistic models.* Image and Vision Computing **18**:2 (2000) 173–187.
13. H.–H. Nagel: *Image Sequence Evaluation: 30 Years and Still Going Strong.* In: A. Sanfeliu et al. (Eds.): Proc. 15th Int. Conf. on Pattern Recognition (ICPR–2000), Barcelona/Spain, 3–7 September 2000, Vol. 1, IEEE Computer Society: Los Alamitos, CA 2000, pp. 149–158.
14. H. H. Nagel, M. Haag, V. Jeyakumar, and A. Mukerjee: *Visualisation of Conceptual Descriptions Derived from Image Sequences.* Mustererkennung 1999, 21. DAGM–Symposium, Bonn, 15–17 September 1999, Springer–Verlag: Berlin·Heidelberg·New York/NY 1999, pp. 364–371.
15. K. H. Schäfer: *Unscharfe zeitlogische Modellierung von Situationen und Handlungen in Bildfolgenauswertung und Robotik.* Dissertation, Fakultät für Informatik der Universität Karlsruhe (TH), Juli 1996; Dissertationen zur Künstlichen Intelligenz (DISKI) **135**; infix–Verlag: Sankt Augustin 1996 (in German).
16. J. R. J. Schirra: *Bildbeschreibung als Verbindung von visuellem und sprachlichem Raum.* Dissertation, Fakultät für Informatik der Universität des Saarlandes, Saarbrücken, April 1994; Dissertationen zur Künstlichen Intelligenz (DISKI) **71**; infix–Verlag: Sankt Augustin 1994 (in German).
17. V. T. Vu, F. Brémond, and M. Thonnat: *Human Behaviour Visualisation and Simulation for Automatic Video Understanding.* In: Proc. of the 10th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG–2002), Plzen–Bory, Czech Republic, 2002, pp. 485–492.