‘Occurrence’ Extraction from Image Sequences of Road Traffic Scenes

Ralf Gerber and Hans-Hellmut Nagel
Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
{gerber|nagel}@ira.uka.de

Abstract

‘Occurrence’ denotes a perceived *element of a spatio-temporal development. In many languages, for example in English and German, occurrences correspond to* verbphrases. *We converted programmed recognition automata for German occurrence representations related to road traffic scenes into English occurrence specifications suitable for manipulation by a Fuzzy Metric-Temporal Horn Logic (***FMTHL***) inference engine. The approach is illustrated for a subset of agent-location occurrences based on the transformation of geometric tracking results obtained by automatic model-based evaluation of road traffic videos.*

1. Introduction

This contribution addresses the conversion of *geometric tracking results* into *elementary conceptual representations* of relevant aspects of (short-term) developments in video recordings of a road traffic scene. We thus have to be concerned with three disciplines, namely computer vision, knowledge reepresentation, and computational linguistics. Since each of these disciplines already covers several subdisciplines, any system for video-to-text transformation will be complex and, therefore, difficult to present and to analyse. We thus provide an overview of the entire system approach in order to set the frame for subsequent more detailed overviews.

The eventual transformation of video signals into a text describing the recorded temporal development within the depicted scene can be subdivided into three groups of processes:

1. The subsystem which controls video recording and subsequent processing steps up to and including the extraction of 3-D time-dependent geometric descriptions of the scene and, in particular, of visibly *moving* bodies: the computer vision subsystem `Xtrack` evaluates monocular image sequences of road traffic scenes recorded by a stationary video camera, see [Haag & Nagel 1999, Leuck & Nagel 2001].

2. The quantitative 3-D spatio-temporal information provided by the model-based vehicle tracking subsystem is converted into an elementary conceptual representation at the interface between the Scene-Domain-Level and the ‘*Conceptual-Primitives-Level*’. Information about the spatio-temporal developments in the scene represented in form of conceptual primitives is aggregated by abstraction processes into information about, for example, the behavior of agents in the depicted scene at the ‘*Behavior-Representation-Level*’. These two layers constitute the interface between the ‘core computer vision’ subsystem on the one hand and a ‘text generation subsystem’ incorporated into the remaining subsystem, namely

3. the ‘*Natural-Language-Level*’. This latter layer could comprise in principle – in addition to a natural language text generation component – also a natural language question-answering component although such a component has not been studied yet in this context.

In order to base such a system on a reliable methodology, treatment of intermediate results at the conceptual level should *use formal logic* to the extent possible. The remainder of this exposition will concentrate on the subsystem for conceptual representation of occurrences.

2 On the Notion of ‘Occurrence’

In general, already a short observation of a road vehicle allows a fairly good prediction regarding its subsequent motion during a few seconds. Such a prediction will become even more reliable if the movements of other traffic participants in its environment, in particular of other road vehicles, and properties of the road can be taken into account. Since

a reliable prediction of the behavior of other traffic participants is important for safe and smooth road traffic, it is no surprise that a highly specific vocabular has been developed in order to communicate such behavior.

Already a short introspection will reveal that such communication can be characterized as either pertaining to the short-term movement of a single vehicle – possibly supplemented by reference to the road or to some other immediately relevant object – or to some abstraction referring to an entire, usually goal-directed, sequence of such short-term movements. We consider the first type as elementary and denote a 'recognizable movement primitive' as an '*occurrence*': this notion appears as sufficiently neutral to allow its potential extension to other types of motion 'primitives' beyond those relating to road vehicle motion.

3. System Outline

At each layer of a feed-forward systems version shown in Figure 1, an entry into the left column corresponds to the representation of intermediate results at that layer. Entries into the right column explicate the kind of knowledge exploited by the transformation subprocess at that particular layer. The left entry at the bottom layer refers to results obtained by the computer vision subsystem `Xtrack`, i. e. estimates for the time-dependent state of vehicles detected and tracked in an input video sequence by a model-based approach. The entry into the right column of the bottom layer ('Geometric Lane Model') refers to a-priori knowledge about the geometry of the (static part of the) depicted scene, in particular (time-independent) *geometric* lane data. These have to be converted into a *conceptual* representation of the lane structure in the recorded scene for further use by the FMTHL inference engine **F-Limette** (see [Schäfer 96]) activated by the conceptual representation subsystem.

A two-step approach transforms the geometric tracking results from a quantitative, numerical representation into a qualitative, conceptual one: the first step converts the input into discrete values compatible with predefined attribute schemes, the second step then combines the resulting attributes to assert '*occurrences*'. The time scale relating to such 'primitive' conceptual representations of vehicle motion extends from a fraction of a second upwards to several seconds.

The next step ('Situation Analysis') combines primitive conceptual representations with knowledge about conditions in the scene which may influence the switch between particular occurrences as being the most appropriate description of (short-term) vehicular behavior. The dominant scale for temporal intervals of interest thus changes by 1-2 orders of magnitude, i. e. to between several seconds and a minute or more. The 'Conceptual Scene Description' obtained at this level is then prepared for presentation to a

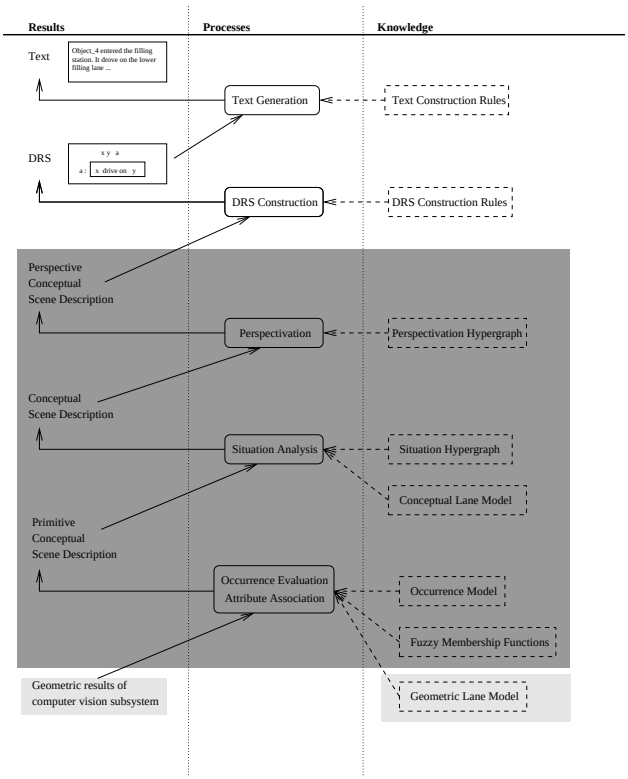

Figure 1. A 'feed-forward' system version designed to convert geometric results obtained by a model-based tracking ('core computer vision') subsystem into a coherent natural language text. The layers underlaid in dark gray use a *Fuzzy Metric-Temporal Horn Logic* in order to represent and manipulate a-priori knowledge together with the results provided by the next lower layer.

human user who is anticipated in our case as a 'reader' of the textual descriptions to be generated. The topmost two layers rely on *Discourse Representation Structures* (**DRS**, see [Kamp & Reyle 1993]).

4 Generation of a Conceptual Representation for Time-Dependent Agent Properties

The 'facts' provided by the computer vision subsystem constitute the link between the 'computer vision' and the 'conceptual representation' subsystems. Some of these facts comprise quantitative numerical values for, e.g., velocity and positions of agents as in

time	!	agent	x	y	θ	v	ψ
[frame#]	!		[m]	[m]	°	[m/s]	°
614	! has_status(	object_4,	8.94,	1.85,	147.0,	2.93,	-2.06).
615	! has_status(	object_4,	8.88,	1.87,	147.1,	2.87,	-2.46).

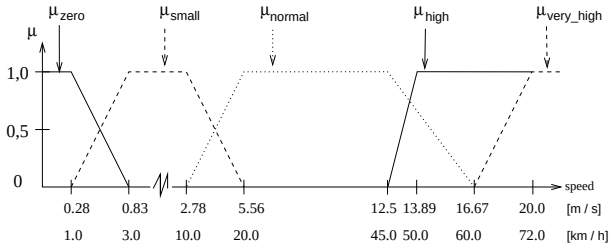

Figure 2. Discretization of continuous speed values into a set of intervals. The upper part shows the fuzzy membership functions $\mu_{speed\ value}$ for the subset { `zero, small, normal, high, very_high` } of discrete conceptual speed values.

This status information about the tracked vehicle 'object_4' provides the domain of definition for an interpretation of the following logical formula

`has_speed(Agent, small)`

which is a logical predicate relating the speed of an agent (e.g., 'object_4') to a discrete conceptual value `small`. To do so, one has to derive the 'Degree of Validity (**DoV**, see Appendix A)' with which the geometric value of the agent's velocity V is mapped to the discrete value `small`. Figure 2 shows one example relating the geometric value for V to the discrete value `small` (and to others). The trapezoidal function μ_{small} can be conceived as a logical interpretation function (see below)

$$\mathcal{I}\{\text{associate\_speed(V, small)}\} = \mu_{small}(V).$$

To be more general, $\mu_{DISCRETE\_VALUE}(V)$ can be written as $\mu_{A,B,C,D}(V)$ where $A < B < C < D$ represent the arguments for which the slope of the trapezoidal function exhibits a discontinuity. According to Figure 2, one obtains for example

$$\mu_{small}(V) = \mu_{0.28,0.83,2.78,5.56}(V)$$

or

$$\mu_{normal}(V) = \mu_{2.78,5.56,12.5,16.67}(V) \quad .$$

In the following, one possible implementation of this predicate in terms of an FMTHL-rule during the import of information from the computer vision subsystem into the conceptual representation subsystem is given:

```
always (has_speed(Agent, small)   :-
     has_status(Agent,X,Y,Theta,V,Psi) ,
     associate_speed(V, small)).
always (associate_speed(V, small) :-
     degreeOfValidity(V,
                0.28,0.83,2.78,5.56)).
```

This corresponds to the following logical interpretation:

$$\begin{aligned}
&\mathcal{I}\{\texttt{has_speed(Agent, small)}\} \\
&== \mathcal{I}\{\texttt{has_status(Agent,X,Y,Theta,V,Psi)} \\
&\qquad\qquad \wedge\ \texttt{associate_speed(V,small)}\} \\
&== \mathcal{I}\{\texttt{has_status(Agent,X,Y,Theta,V,Psi)} \wedge \\
&\quad \texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)} \\
&\qquad \} \\
&== \min\Big(\\
&\quad \mathcal{I}\{\texttt{has_status(Agent,X,Y,Theta,V,Psi)}\}, \\
&\quad \mathcal{I}\{\texttt{degreeOfValidity(V,0.28,0.83,2.78,} \\
&\qquad\qquad\qquad\qquad \texttt{5.56)}\} \Big)
\end{aligned}$$

where the symbol == has been used to indicate that this operation refers to a numeric equality, i. e. neither to a logical equality operator nor to an assignment operator as it is used in many programming languages. The minimum function has been used as the fuzzy version of the conjunction.

Since 'has_status' is considered to have exclusively either the highest possible DoV (i. e. 1) or the lowest one (i. e. 0), one obtains

$$\mathcal{I}\{\texttt{has\_speed(Agent, small)}\} == 0$$

provided that

$\mathcal{I}\{\texttt{has\_status(Agent,X,Y,Theta,V,Psi)}\} == 0$.

Such a value will occur at time points where no trajectory data for the agent is available (for example because the agent is currently not in the field of view of the recording camera). As a consequence, no geometric velocity value V for the agent will be available and thus no association to a discrete value can be performed. Otherwise, one obtains

$$\begin{aligned}
&\mathcal{I}\{\texttt{has_speed(Agent, small)}\} == \min\Big(1,\ \mathcal{I}\{ \\
&\quad \texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)} \\
&\qquad\qquad\qquad\qquad \} \Big) \\
&== \mathcal{I}\{ \\
&\texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)}\} \\
&== \mu_{0.28,0.83,2.78,5.56}(V) \quad .
\end{aligned}$$

As a result, geometric input data can be associated with discrete conceptual values by defining FMTHL predicates such as `has_speed(Agent, small)`. These predicates constitute the head of simple FMTHL rules whose body consists of the non-fuzzy '`has_status`'-predicate in order to access the geometric value and of the '`degreeOfValidity`'-predicate (see Appendix A). The latter represents the compatibility of the value imported from the computer vision subsystem with the vagueness of the discrete concept `small` as expressed numerically by the trapezoidal function.

5 The Instantiation of Occurrences

A systematic search for all verbs in a standard dictionary of the German language yielded a subset of about sixty verbs (from among about 9200) which are related

to road vehicle movements (see [Kollnig & Nagel 1993], [Nagel 1987]). Occurrences related to this subset can be categorized

1. as **perpetuative** if they tend to retain the dominant aspect of a movement without major change,

2. as **mutative** if they characterize the *systematic change of some aspect*, or

3. as **terminative** if they are related to the *beginning* or *ending* of a dominant movement characteristic.

The algorithmic 'recognition' of a particular occurrence will be presented here as based on *logical inference* although the approach exhibits obvious *analogies* to pattern recognition – *provided* the essential *time-dependencies are neglected*. Each occurrence can be characterized uniquely by a conjunction of predicates (see [Kollnig & Nagel 1993]). These in turn consist of a conjunction of up to three (sub)predicates, namely

1. a *PRE-Condition (***PREC***)* which has to be satisfied *before* the occurrence in question could be considered to represent a valid description of the temporal development in which the agent is involved;

2. a *MONotonicity-Condition (***MONC***)* indicating the type of admissible monotonous change which may take place while the occurrence represents a valid description;

3. a *POST-Condition (***POSTC***)* which becomes true once the occurrence in question will no longer constitute an adequate description of the temporal development in which the agent is involved.

As will be seen shortly, evaluation of temporal relations between the validity of these three (sub)predicates are essential for a proper characterization of occurrences. In some cases, the monotonicity condition will be irrelevant and thus can be omitted. In other cases, only the monotonicity condition will be relevant such that pre- and post-conditions could be omitted. In a fourth variant, the pre-condition remains true while an occurrence constitutes a valid description of temporal developments although attribute values may vary during this period, thereby preventing the satisfiability of a monotonicity condition. If the pre-condition is identical with the post-condition, pre- and post-condition are unified into a **COND**ition.

Table 1 presents the definition of predicates which together characterize occurrences referencing both the agent and a specified location in the road traffic scene.

5.1 Taking temporal dependencies into account: *transductors* for occurrence recognition

The a-priori knowledge about temporal relations between the satisfaction of equality attributes has been coded into the *type* of an occurrence (perpetuative, mutative, terminative) and into the way attribute values are required for the PREC, the MONC, and the POSTC. A finite state acceptance automaton ('transductor') has been designed for each type. These transductors determine whether or not the required conditions are satisfied in the prescribed temporal order.

5.1.1 Transductor for *perpetuative* occurrences

The transition diagram for this transductor is given in Figure 3. This transductor is realized by the following FMTHL inferences (see [Schäfer 96] for an explanation of the logical vocabulary of FMTHL):

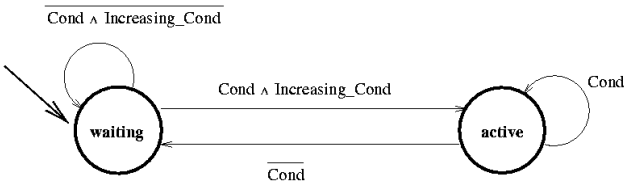

Figure 3. Transductor for the recognition of *perpetuative* occurrences.

```
R1: always (perpetuative(Agent,Loc,Verb)       :-
              waiting_perp(Agent,Loc,Verb)).
R2: always (waiting_perp(Agent,Loc,Verb)       :-
              condition(Agent,Loc,Verb) ,
              increasing_condition(Agent,Loc,Verb),
               ! , active_perp(Agent,Loc,Verb)).
R3: always (waiting_perp(Agent,Loc,Verb)       :-
              has_status(Agent,X,Y,Theta,V,Psi) ,
              1 ? waiting_perp(Agent,Loc,Verb)).
R4: always (active_perp(Agent,Loc,Verb)        :-
              condition(Agent,Loc,Verb) , ! ,
              note(occurrence(Agent,Loc,Verb)) ,
              1 ? active_perp(Agent,Loc,Verb)).
R5: always (active_perp(Agent,Loc,Verb)        :-
              has_status(Agent,X,Y,Theta,V,Psi) ,
              1 ? waiting_perp(Agent,Loc,Verb)).
```

The heads of the rules (implications) correspond to the states of the automaton. Their bodies comprise – in addition to conditions according to each state – the name of the (possible) successor state. Evaluation starts in rule R1. Rule R2 tests (according to the state 'waiting' of the automaton), whether a certain *condition* is currently valid and whether its degree of validity (DoV) increases during five consecutive time points. In this case, the automaton switches into the 'active' state (R4-R5). Otherwise, provided additional

Occurrence	type	**has_speed**(Agent)			**has_course**(Agent,Location)			**has_distance**(Agent,Location)		
		PREC	MONC	POSTC	PREC	MONC	POSTC	PREC	MONC	POSTC
arrive at loc	term	moving	-	moving	-	-	-	small	>	zero
depart from loc	term	moving	-	moving	-	-	-	zero	<	small
drive to loc	mut	moving	-	moving	appr.	-	appr.	not_zero	>	small
leave loc	term	zero	<	moving	-	-	-	zero	-	-
leave loc behind	mut	moving	-	moving	leaving	-	leaving	small	<	not_zero
park at loc	perp	zero	-	zero	-	-	-	zero	-	zero
pass loc	term	moving	-	moving	appr.	changing	leaving	not_zero	-	not_zero
run over loc	perp	moving	-	moving	-	-	-	zero	-	zero
stop at loc	term	moving	>	zero	-	-	-	-	-	zero

Table 1. Time-dependent (!) predicates defining occurrences which refer to both the agent and a location. The symbol '>' indicates a *decreasing* slope for the value subject to the monotonicity condition MONC, the symbol '<' correspondingly an *increasing* slope. 'has_course' denotes the abbreviation of the predicate `has_course_towards_loc`, see Appendix A.4. Similarly, 'has_distance' stands for the predicate `has_distance_to_loc`, see Appendix A.5.

trajectory data are available, the automaton remains 'waiting' (R3).

Perpetuative occurrences which refer only to the agent can be evaluated just by using attributes with agent reference (speed, direction, and mode). The entry for '`park at loc`' in Table 1 can be transformed into an FMTHL implication according to the following example:

```
always (condition(Agent,Loc,park_at_loc)      :-
        has_speed(Agent,zero),
        has_distance_to_loc(Agent,Loc,zero)).
```

In this case, the formulation of a single condition can be extracted directly from Table 1 in analogy to explanations in Section 4. Evaluation is started by a simple logical query, for example in the case of 'park_at_loc(Loc)':

```
?- perpetuative(Agent,Loc,park_at_loc).
```

5.1.2 Transductor for *mutative* occurrences

The inferences performed by this tranductor are illustrated in Figure 4. The predicates appearing in the following are explained in Appendix A.

```
always (mutative(Agent,Loc,Verb)    :-
            waiting_mut(Agent,Loc,Verb)).
always (waiting_mut(Agent,Loc,Verb) :-
            precondition(Agent,Loc,Verb) ,
            mon_condition(Agent,Loc,Verb) ,
            increasing_moncondition(Agent,Loc,Verb) ,
            postcondition(Agent,Loc,Verb) , ! ,
            terminating_mut(Agent,Loc,Verb)).
always (waiting_mut(Agent,Loc,Verb) :-
            precondition(Agent,Loc,Verb) ,
            mon_condition(Agent,Loc,Verb) ,
            increasing_moncondition(Agent,Loc,Verb) ,
            active_mut(Agent,Loc,Verb)).
always (waiting_mut(Agent,Loc,Verb) :-
            has_status(Agent,X,Y,Theta,V,Psi) ,
             1 ? waiting_mut(Agent,Loc,Verb)).
always (active_mut(Agent,Loc,Verb)  :-
```

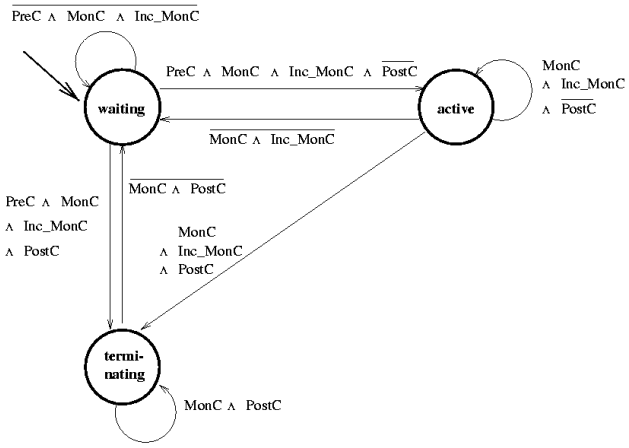

Figure 4. Transductor for the recognition of *mutative* occurrences.

```
            mon_condition(Agent,Loc,Verb) ,
            increasing_moncondition(Agent,Loc,Verb) ,
             ! , postcondition(Agent,Loc,Verb) ,
            terminating_mut(Agent,Loc,Verb)).
always (active_mut(Agent,Loc,Verb)  :-
            mon_condition(Agent,Loc,Verb) ,
            increasing_moncondition(Agent,Loc,Verb) ,
             1 ? active_mut(Agent,Loc,Verb)).
always (active_mut(Agent,Loc,Verb)  :-
            has_status(Agent,X,Y,Theta,V,Psi) ,
             1 ? waiting_mut(Agent,Loc,Verb)).
always (terminating_mut(Agent,Loc,Verb)   :-
            postcondition(Agent,Loc,Verb) ,
            mon_condition(Agent,Loc,Verb) ,
            note(occurrence(Agent,Loc,Verb)) , ! ,
            1 ? terminating_mut(Agent,Loc,Verb)).
always (terminating_mut(Agent,Loc,Verb)   :-
            has_status(Agent,X,Y,Theta,V,Psi) ,
             1 ? waiting_mut(Agent,Loc,Verb)).
```

5.1.3 Transductor for *terminative* occurrences

```
always (terminative(Agent,Loc,Verb)  :-
           waiting_term(Agent,Loc,Verb)).
```

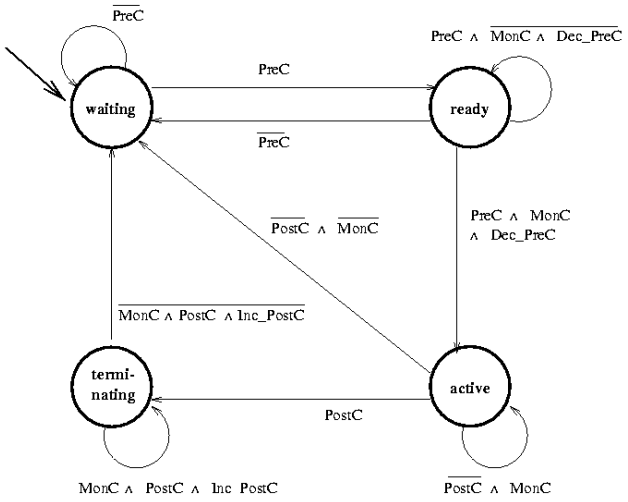

Figure 5. Transductor for the recognition of *terminative* occurrences.

```
always (waiting_term(Agent,Loc,Verb) :-
          precondition(Agent,Loc,Verb) , ! ,
          ready_term(Agent,Loc,Verb)).
always (waiting_term(Agent,Loc,Verb) :-
          has_status(Agent,X,Y,Theta,V,Psi) , 1
           ? waiting_term(Agent,Loc,Verb)).
always (ready_term(Agent,Loc,Verb)   :-
          precondition(Agent,Loc,Verb) ,
          mon_condition(Agent,Loc,Verb) ,
          decreasing_precondition(Agent,Loc,Verb) ,
           ! , 1 ? active_term(Agent,Loc,Verb)).
always (ready_term(Agent,Loc,Verb)   :-
          precondition(Agent,Loc,Verb) , ! ,
           1 ? ready_term(Agent,Loc,Verb)).
always (ready_term(Agent,Loc,Verb)   :-
          has_status(Agent,X,Y,Theta,V,Psi) ,
           1 ? waiting_term(Agent,Loc,Verb)).
always (active_term(Agent,Loc,Verb)  :-
          postcondition(Agent,Loc,Verb) , ! ,
           1 ? terminating_term(Agent,Loc,Verb)).
always (active_term(Agent,Loc,Verb)  :-
          mon_condition(Agent,Loc,Verb) , ! ,
          has_status(Agent,X,Y,Theta,V,Psi) ,
           1 ? active_term(Agent,Loc,Verb)).
always (active_term(Agent,Loc,Verb)  :-
          waiting_term(Agent,Loc,Verb)).
always (terminating_term(Agent,Loc,Verb)  :-
          postcondition(Agent,Loc,Verb) ,
          increasing_postcondition(Agent,Loc,Verb) ,
          mon_condition(Agent,Loc,Verb) , ! ,
          note(occurrence(Agent,Loc,Verb)) ,
           1 ? terminating_term(Agent,Loc,Verb)).
always (terminating_term(Agent,Loc,Verb)  :-
          has_status(Agent,X,Y,Theta,V,Psi) , 1
           ? waiting_term(Agent,Loc,Verb)).
```

Pre-, monotonicity- and postcondition have to be evaluated in order to determine the validity of 'stop_at_loc(Agent,Loc)'. These conditions can be derived directly from Table 1:

```
always (precondition(Agent,Loc,stop_at_loc)
      :- has_speed(Agent,moving)).

always (mon_condition(Agent,Loc,stop_at_loc)
      :-  has_speed_change(Agent,slower)).

always (postcondition(Agent,Loc,stop_at_loc)
      :-  has_speed(Agent,zero),
          has_distance_to_loc(Agent,Loc,zero)).
```

6. Conclusions

An interface between the extraction of geometric information and the formulation of this information as a natural language text has been implemented on the basis of a fuzzy metric-temporal (Horn) logic. The approach has been illustrated for the case of occurrences related to a (vehicular) agent and a location in road traffic. For additional cases related (i) solely to an agent, (ii) to an agent and the lane it drives on, or (iii) to an agent and another (standing or moving) object see [Gerber & Nagel 2002]. The generation of coherent text within the system outlined above is treated in [Gerber 2000].

A. Appendix

A.1 Special predicate 'degreeOfValidity'

`degreeOfValidity` constitutes a special 5-ary predicate symbol. Its DoV corresponds to the function value of the trapezoidal function described by the last four arguments related to its first argument, see Section 4. The 'Meta-Predicate' `sp` overwrites the DoV by the arithmetic expression of its argument, see [Schäfer 96].

```
always (degreeOfValidity(X,P1,P2,P3,P4)  :-
    X >= P1 , X < P2 , sp((X - P1) / (P2 - P1)) ;
    X >= P2 , X < P3 , sp(1.0) ;
    X >= P3 , X < P4 , sp((P4 - X) / (P4 - P3))).
```

A.2 Supplementary Rules

```
always (derivative(A,B,C,D,E,Deriv)   :-
   Deriv is (-2.0*A)-B+D+(2.0*E)).

always (ang_direction(X,Y,Ang)        :-
   X==0 , Y < 0 , Ang is -90 ;
   X==0 , Y >= 0 , Ang is 90 ;
   X>0 , Y>=0 ,
            Ang is atan(Y/X) * 360 / 6.2831853 ;
   X>0 , Y<0 ,
      Ang is atan(Y/X) * 360 / 6.2831853 + 360 ;
   X<0 ,
      Ang is atan(Y/X) * 360 / 6.2831853 + 180).

always (ang_norm(Ang,Norm)            :-
   Ang >= 180 , Norm is 360 - Ang ;
   Ang < -180 , Norm is 360 + Ang ;
```

```
   Ang >= -180 , Ang < 180 , Norm is 1 * Ang).

always (ang_diff(R,T,Diff) :- Diff is R - T).

always (length(A,B,L)                :-
                L is sqrt((A * A) + (B * B))).

always (maximum(A,B,M)               :-
             A < B , M is B ; A >= B , M is A).
```

`increasing_condition` Binary predicate symbol. True, if the DoV of *condition(Agent,Loc,Verb)* is increasing at five consecutive time instants; analogously for *increasing_moncondition* and *increasing_postcondition*.

```
always (increasing_condition(Agent,Loc,Verb) :-
        -2 ? (A1 {condition(Agent,Loc,Verb)} B1) ,
        -1 ? (A2 {condition(Agent,Loc,Verb)} B2) ,
             (A3 {condition(Agent,Loc,Verb)} B3) ,
         1 ? (A4 {condition(Agent,Loc,Verb)} B4) ,
         2 ? (A5 {condition(Agent,Loc,Verb)} B5) ,
        positive_derivative(B1,B2,B3,B4,B5)).

always (positive_derivative(A,B,C,D,E) :-
        Derivative is (-2.0*A)-B+D+(2.0*E) ,
        Derivative > 0).
```

A.3 Attributes related to 'speed'

The fuzzy membership functions underlying the conversion of numerical speed values into 'conceptual constants' are shown in Figure 2.

```
always (has_speed(Agent,Value)                :-
     has_status(Agent,X,Y,Theta,V,Psi) ,
     associate_speed(V,Value)).
always (associate_speed(V,Value)              :-
     degreeOfValidity(V,0,0,0.28,0.83) ,
                                 Value = zero ;
     degreeOfValidity(V,0.28,0.83,2.78,5.56) ,
                                 Value = small ;
     degreeOfValidity(V,2.78,5.56,12.5,16.67) ,
                                Value = normal ;
     degreeOfValidity(V,12.5,13.89,100.0,100.0) ,
                                 Value = high ;
     degreeOfValidity(V,16.67,20.0,100.0,100.0) ,
                              Value = very_high ;
     degreeOfValidity(V,0.28,0.83,100.0,100.0) ,
                                Value = moving).
always (has_speed_change(Agent,Value)          :-
-2 ! has_status(Agent,X_2,Y_2,Theta_2,V_2,Psi_2) ,
-1 ! has_status(Agent,X_1,Y_1,Theta_1,V_1,Psi_1) ,
     has_status(Agent,X,Y,Theta,V,Psi) ,
 1 ! has_status(Agent,X1,Y1,Theta1,V1,Psi1) ,
 2 ! has_status(Agent,X2,Y2,Theta2,V2,Psi2) ,
 derivative(V_2,V_1,V,V1,V2,Deriv) ,
 associate_speed_change(Deriv,Value)).
always (associate_speed_change(Deriv,Value)    :-
  degreeOfValidity(Deriv,-100,-100,-1.11,-0.56) ,
                                Value = slower ;
  degreeOfValidity(Deriv,-1.11,-0.56,0.56,1.11) ,
                              Value = constant ;
  degreeOfValidity(Deriv,0.56,1.11,100,100) ,
                                Value = faster).
```

A.4 Attributes w.r.t. 'course_towards_location'

The 'conceptual constants' expressing the different admitted descriptors for the 'course' of an agent with respect to a location are illustrated by Figure 6.

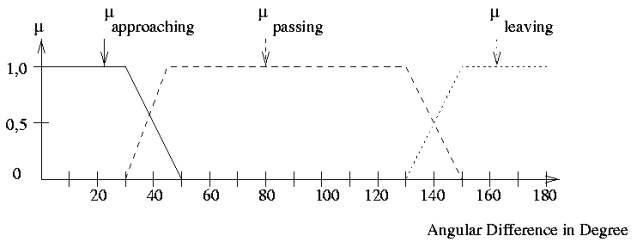

Figure 6. Mapping the angular difference between the vehicle orientation and the line connecting the current vehicle position with a specified location in the environment into the conceptual 'course descriptions' `approaching`, `passing`, and `leaving` the location.

```
always (has_course_towards_loc(Agent,Loc,Value)
                                                :-
   course_of_agent_towards_loc(Agent,Loc,Course),
   associate_course_towards_loc(Course,Value)).
always (course_of_agent_towards_loc(Agent,
                                    Loc,Course) :-
   has_status(Agent,X,Y,Theta,V,Psi) ,
   location(Loc,XO,YO) ,
   ang_direction(XO-X,YO-Y,R) ,
   ang_diff(Theta,R,Diff) ,
   ang_norm(Diff,Course)).
always (associate_course_towards_loc(Course,
                                        Value) :-
   degreeOfValidity(Course,-100,-100,30,50) ,
                            Value = approaching ;
   degreeOfValidity(Course,30,50,130,150) ,
                                Value = passing ;
   degreeOfValidity(Course,130,150,360,360) ,
                                Value = leaving).
always (has_course_change_towards_loc(Agent,
                                     Loc,Value) :-
   -2 ! course_of_agent_towards_loc(Agent,
                                    Loc,Course_2) ,
   -1 ! course_of_agent_towards_loc(Agent,
                                    Loc,Course_1) ,
        course_of_agent_towards_loc(Agent,
                                    Loc,Course) ,
    1 ! course_of_agent_towards_loc(Agent,
                                    Loc,Course1) ,
    2 ! course_of_agent_towards_loc(Agent,
                                    Loc,Course2) ,
    derivative(Course_2,Course_1,Course,Course1,
                                  Course2,Deriv) ,
    associate_course_change_towards_loc(Deriv,
                                        Value)).
always(associate_course_change_towards_loc(Deriv,
                                        Value) :-
```

```
      degreeOfValidity(Deriv,-15,-5,5,15) ,
                              Value = constant ;
      degreeOfValidity(Deriv,-100,-100,-15,-5) ,
                              Value = changing ;
      degreeOfValidity(Deriv,5,15,100,100) ,
                              Value = changing).
```

A.5 Attributes related to 'distance_to_location'

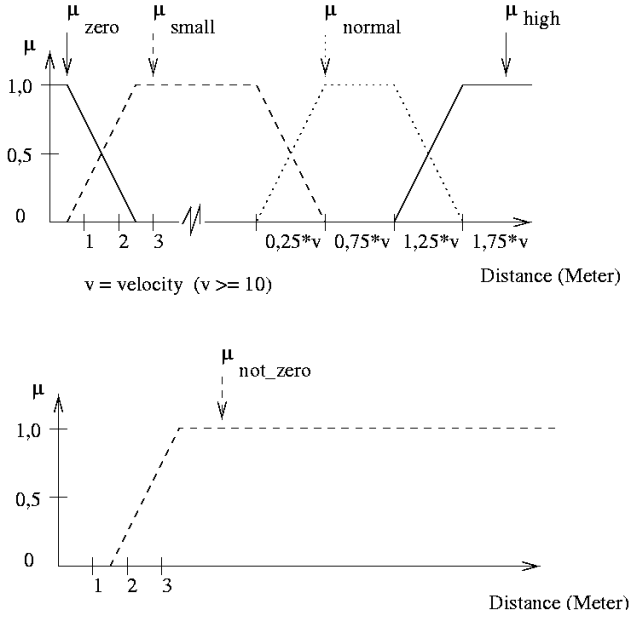

Figure 7. Mapping the distance between the current estimate for the vehicle position and a specified location in the environment into a set of conceptual values. Note that the conceptual values for the attribute `distance_to_location` depends, too, on the estimated velocity: the 'threshold' for the assignment of, e. g., the conceptual value `normal` increases with increasing speed.

```
always (has_distance_to_loc(Agent,Loc,Value)  :-
   distance_of_agent_to_loc(Agent,Loc,Distance,
                                      Offset) ,
   associate_distance_to_loc(Distance,Offset,
                                       Value)).
always (distance_of_agent_to_loc(Agent,Loc,
                                    Distance)  :-
   has_status(Agent,X,Y,Theta,V,Psi) ,
   location(Loc,XO,YO) ,
   length(X-XO,Y-YO,Distance) ,
   maximum(V,10,Offset)).
always (associate_distance_to_loc(Distance,OS,
                                     Value)  :-
   degreeOfValidity(Distance,-5,-5,0.5,2.5) ,
                                  Value = zero ;
   degreeOfValidity(Distance,0.5,2.5,0.25*OS,
                                0.75*OS) , Value = small ;
   degreeOfValidity(Distance,0.25*OS,0.75*OS,
            1.25*OS,1.75*OS) , Value = normal ;
   degreeOfValidity(Distance,1.25*OS,1.75*OS,
                             100,100) , Value = high ;
   degreeOfValidity(Distance,1.5,3.5,100,100) ,
                                  Value = not_zero).
always (has_distance_change_to_loc(Agent,Loc,
                                       Value)  :-
   -2 ! distance_of_agent_to_loc(Agent,Loc,
                              Distance_2,Offset_2) ,
   -1 ! distance_of_agent_to_loc(Agent,Loc,
                              Distance_1,Offset_1) ,
        distance_of_agent_to_loc(Agent,Loc,
                                Distance,Offset) ,
    1 ! distance_of_agent_to_loc(Agent,Loc,
                             Distance1,Offset1) ,
    2 ! distance_of_agent_to_loc(Agent,Loc,
                             Distance2,Offset2) ,
    derivative(Distance_2,Distance_1,Distance,
                       Distance1,Distance2,Deriv) ,
    associate_distance_change_to_loc(Deriv,
                                             Value)).
always (associate_distance_change_to_loc(Deriv,
                                       Value)  :-
   degreeOfValidity(Deriv,-1.5,-0.5,0.5,1.5) ,
                                  Value = constant ;
   degreeOfValidity(Deriv,-100,-100,-1.5,-0.5) ,
                                   Value = smaller ;
   degreeOfValidity(Deriv,0.5,1.5,100,100) ,
                                   Value = higher).
```

References

[Gerber 2000] R. Gerber: *Natürlichsprachliche Beschreibungen von Straßenverkehrsszenen durch Bildfolgenauswertung* (in German). Dissertation, Universität Karlsruhe (TH), Januar 2000; `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=2000/informatik/8`

[Gerber & Nagel 2002] R. Gerber and H.-H. Nagel: *Text Generation from Image Sequences of Road Traffic Scenes*. Internal Report, Institut für Algorithmen und Kognitive Systeme, June 2002.

[Haag & Nagel 1999] M. Haag and H.-H. Nagel: *Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Image Sequences*. International Journal of Computer Vision **35**:3 (1999) 295-319.

[Kamp & Reyle 1993] H. Kamp and U. Reyle: *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht Boston London 1993.

[Kollnig & Nagel 1993] H. Kollnig und H.-H. Nagel: *Ermittlung von begrifflichen Beschreibungen von Geschehen in Straßenverkehrsszenen mit Hilfe unscharfer Mengen* (in German). Informatik - Forschung und Entwicklung **8** (1993) 186-196.

[Leuck & Nagel 2001] H. Leuck and H.-H. Nagel: *Model-Based Initialisation of Vehicle Tracking: Dependency on Illumination*. Proc. 8th Intern. Conf. Computer Vision (ICCV 2001), 9-12 July 2001, Vancouver/BC, Canada, Vol. I, pp. 309-314.

[Nagel 1987] H.-H. Nagel: *From Image Sequences Towards Conceptual Descriptions*. Invited Lecture presented at the Third Alvey Vision Conference, 15-17 September 1987, Cambridge, UK; see Image and Vision Computing **6**:2 (1988) 59-74.

[Schäfer 96] K.H. Schäfer: *Unscharfe zeitlogische Modellierung von Situationen und Handlungen in Bildfolgenauswertung und Robotik* (in German). Dissertation, Univ. Karlsruhe (TH), Juli 1996. DISKI **135**, Sankt Augustin: infix 1996.