Automan

28-04-1999

RijksUniversiteit Groningen

Michael Heemskerk

Lennart Quispel

Sven Warris

Table of contents

Part I

Chapter 1
Abstract

This paper describes the design and implementation of an autonomous agent for controlling vehicles in a traffic simulator. This agent is based on recent developments in artificial intelligence, autonomous robotics and cognitive psychology. The goal of the agent is to simulate realistic driving behavior. A cognitive architecture was developed to implement the design of the agent. This architecture is based on rule-base-evaluation and is used in trying to describe human driving behavior and to function in the way prescribed by the model. Fuzzy logic is used to assure natural flow of information and to make human-like reasoning possible.

Chapter 2
Introduction

2.1 From system to agent

The COV at the University of Groningen is specialized in traffic and environmental research. Their main tool for the research is an impressive traffic simulator. It already contains autonomous agents: the cars in the simulator are controlled by a set of rules which allow them to navigate through the system. For a more detailed description, see chapter 3.

The major goal of this research project was to develop a cognitive model of a human driver. Therefore the model should give an adequate description of the driver of a car.

After completion this model should be incorporated in the simulator. This document will give a full account on how this model came about, the choices made and why and how the model was incorporated in the simulator as an autonomous agent.

2.2 Goals

The primary goal is to develop an autonomous agent of a driver based on a cognitive plausible model. This goal can be divided into three subgoals:

1. Development of a cognitive driver model based on cognitive psychology
2. Development of a cognitive architecture based on and suited for the model
3. Implementation of the model in the architecture and evaluating its usefulness as an autonomous agent

These goals were not set immediately. in the beginning, the first subgoal was the major goal of the project. But after long debate and research this proved to be impossible to achieve without subgoals two and three. Later on in this document it will come clear why this is the case.

2.3 Theoretical Background

When you set out to develop a cognitive model one thing should be considered carefully: what is cognition? It is not in the scope of this document to discuss the philosophical issues associated with this question. For those interested in this field see Kim, 1996 ([23]). It is an important question, however, because all functions as defined to be part of the driving process should be incorporated in the model. According to some, cognition is a higher mental activity of the brain(Matlin (1983, [24]), Stillings (1987, [37])). This definition includes acquisition, storage, retrieval and use of knowledge. It excludes however low-level perception and motor control. Recent-

ly, others (Port(1995,[30]) claim the last two functions are also part of the complete cognitive process. We agree with the latter, and will include perceptual and motor processes in our model. Not only does this correspond better to the recent views held in cognitive science, it is also a practical choice; it would be nearly, if not completely, impossible to design a model that could drive a car without perception and motor control.

Having adopted this convenient view of cognition, three fields of research were helpful in developing the model.

2.3.1 Autonomous Agents

In the third sub-goal in section 2.2 autonomous agents are introduced. An autonomous agent is an artificial created 'creature' in a (virtual or real) world, that can act and react without direct interference from a person. All sorts of techniques known from Artificial Intelligence (Rich(1983,[33])) are used to build agents that can do one or more tasks. Commonly used techniques are neural networks (Haykin (1994, [18]), fuzzy logic (Kosko (1992,[24]), and genetic algorithms (Holland (1992,[19]). This field of research is known as Artificial Life (AL) or Autonomous Robotics (in the case the agent is to act in the real world). Because many agents are based on systems found in animals (from insects to dogs) these creatures are known as Automated Animals, or automals for short.

A great deal of the model presented here is based on the research of Blumberg and Galyean ([6]) from MIT. They created a virtual dog named Silas which is used to interact with people. This dog is a perfect example of an automal. Because the model implements a human driver the name automal is not quite right, although a human is an animal. The model of the driver is more an Automated Human and therefore the agent is called Automan[1].

2.3.2 Behavior Based Approach

The design of Automan is also heavily inspired by the so-called *Behavior Based approach (*Braitenberg (1984,[7]),Brooks (1986,[8],1991,[9]),Steinhage(1997,[33])). This approach of designing Autonomous Agents was introduced by Brooks in 1986. The idea is, that to model intelligent behavior in a complex, dynamic environment, one can specify relatively independent modules that perform small parts of the behavior. Traditionally, one would make a central controller, that has access to all sensorial information and incorporates a decision mechanism to decide on the correct action. To do this, the controller needs a central representation of the environment, that has to be extracted from sensorial data. There are several problems with this approach. A central representation is needed, that has to be updated when an action is performed or the environment changes, which is inherently complex (Dennet(1987,[12]),McFarland(1996,[12])). This representation has to contain all the in-

1. This is also a nice playing with words; in dutch, 'auto' means car, and 'man' means man / human.

formation neccessary to decide on an action. Furthermore, for all possible situations, one or more decision rules has to be given in advance. The more complex the task of the agent and its environment gets, the more rules are needed. Also, these rules get increasingly complex.

By contrast, in the behavior based approach the task of the agent is split up in small elementary *behaviors.* To avoid confusion, we will hereafter refer to these behaviors as behavioral patterns. These behavioral patterns are directly coupled to their relevant sensorial inputs.For example, a behavioral pattern would be Lane_Following. This behavioral pattern keeps the agent driving straight on and uses the sensorial information about the position of the vehicle on the lane; no abstract representation of the agent and its environment is needed. All behavioral patterns have a certain *activation level*[1]. Only behavioral patterns with high activation levels will be executed. Mostly, a method is used that only executes the most active behavioral patterns; this can be done by using only two activation levels (either active or inactive), or using dynamical systems (e.g. in Steinhage (1997,[33])). The Lane_Following behavioral pattern will by default have a high activation. The various behavioral patterns can *interact;* they can reinforce or inhibit each other's activation level. For example, two other behavioral patterns would be Obstacle_Detection and Obstacle_Avoidance. The Obstacle_Avoidance behavioral pattern uses sensorial information about the environment immediately in front of the agent, and tries to detect obstacles. Like the Lane_Following behavioral pattern, it has by default a high activation value. If it detects an obstacle, it will inhibit Lane_Following, but will activate the Obstacle_Avoidance behavioral pattern. This behavioral pattern will steer the agent around the obstacle.

Activation of behavioral patterns can not only be done by other behavioral patterns, but also by direct sensor input. For example, instead of having the two behavioral patterns Obstacle_Detection and Obstacle_Avoidance, one can have a behavioral pattern (e.g. Obstacle_Handling) that is activated by an obstacle sensor of some sort and inhibits Lane_Following. How one specifies the behaviors and how they are activated is of course dependent on the task to be performed and the sensors used.

The overall behavior of the agent is now determined by the interaction between the elementary behavioral patterns. By specifying relatively simple behavior patterns complex behavior will emerge from the interaction of the smaller ones. Silas (see section 2.3.1) proved that by finding these smaller elements of the behavior which one tries to model, complex behavior need not to be specified explicitly.

1. In Brooks original conception, behavioral patterns could be either active or inactive. However, in various extensions of the behavior based approach activation values have been used. A behavioral pattern can be more or less active in this way, which makes the approach more flexible.

2.3.3 Fuzzy Techniques

With designing Automan a choice was made which is considered a waste of resources in traditional AL. When an agent resides in a virtual world it can access information from the surrounding with a very high degree of precision. But this sort of perfection is unwanted in the automan because humans do not have access to this kind of information. The real world distorts information. With that the human perception is also not perfect. This makes it that humans need to be able to deal with uncertain and unclear information. Rules humans use are *if the car is close and I'm driving fast then break*. Because of the imperfection of the information the driver does not know *exactly* the distance between him and the other car. But he is capable of reacting in this situation. When making an error in perceiving the car he can make a wrong choice. This happens in everyday life and therefore it must be incorporated in the functionality of the model. This model will be implemented in a computer program so the need for a system that can handle information like *close distance* and *high speed* arises. If the architecture should translate these value to 20 meters or 100 kph the relation with human reasoning will not hold. This is the reason why the architecture makes use of fuzzy logic. For an example of a fragment of a Fuzzy rulebase see table 1.

Table 1 Fuzzy rulebaseFragment of a fuzzy rulebase

Distance	Own speed	Other's speed	Acceleration
small	high	low	break hard
small	mediate	low	break
medium	mediate	medium	none
large	low	medium	medium increase
large	medium	high	medium increase

Table 1 shows a small part of a Fuzzy rulebasefuzzy rulebase. Such a rulebase usually consists of dozens of rules. Depending on the value of each variable the activation for *every* rule is determined. This in contrast with boolean logic. In that case only one rule will be active. See the appendix on Fuzzy Logic for detailed information. The main reason we have chosen for fuzzy logic is that it enables the system to have a more natural way of constructing and evaluating rules.

With the use of fuzzy logic, driver experience can be easily modeled. *Close to curvestone* may range for a novice race car driver from 10 to 20 cm from the curvestone, but an experienced driver may find this already *far* from the curvestones and handle a range of 0 to 3 cm from the curvestones as *close*. By giving different ranges for the same linguistic variables not only different levels of experience can be modeled, but also feelings of safety and risktaking can be described in principle.

2.4 Structure of this document

The structure of this document reflexes the sequence of the goals set in section 2.2. In part 1 terms and parameters of these goals are set. It will give an overview on autonomous agents, fuzzy logic and the simulator. Part 2 will give a full description of the cognitive model, followed by part 3 which will give a description of the cognitive architecture.

Chapter 3
The Traffic Simulator

3.1 Introduction

This chapter will give a description of the traffic simulator. This might help the reader to understand the environment in which the agent will be placed. It will also give a short overview of the types of research the simulator is used for. Most of the description comes directly from Wolffelaar and Van Winsum ([48]) and Bakker et. al.([4]).

The traffic simulator is situated in a large room in the building of the faculty. This space holds the three essential objects for the simulator: a BMW 518, a powerful computer and three video projection screens. The system has been developed for behavioral traffic research and has incorporated this concept of interactivity as a central principle in the design. The definition of a particular traffic environment starts by defining a logical road network. Next the scenario will be defined for every interactive traffic participant and its specific actions.

3.2 System design and functionality

The hart of the computer consists of two different systems. One system is completely dedicated to the graphical display of the simulated world and the other one calculates all traffic-related tasks. This gives the system a constant refresh-rate (= the amount of frames per second), no matter how intensive the other calculations are. The human driver sees the autonomous agents as cars projected on the screen.

Although the projection screen is best viewed from inside the BMW, the projector can display the environment from almost every angle. Generally this the viewpoint of the driver at eye-height, but viewing the scenario playback from an other angle may give new insights in what happened.

The traffic system is represented as a logical network of nodes. A logical network

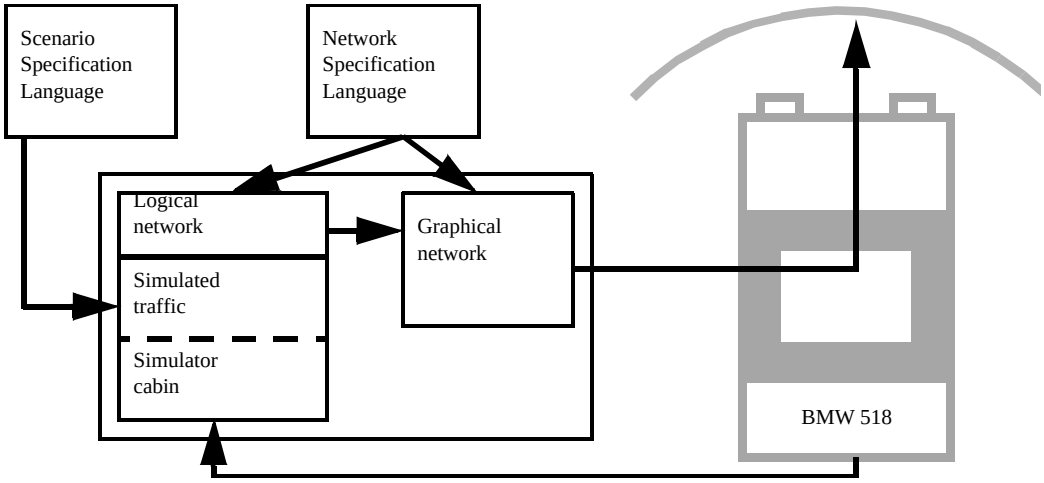

Figure 1: Functional overview of the TRC driving simulator

gives a description of the world in logical terms enabling real-time traversal and computations of traffic encounters. The graphical system uses this network together with a geometric description to visualize the environment in real-time.

In this network all properties of the road-segments (intersections etc.) are stored. Properties can be position, dimension, curve, etc.

- **Intersection**. Several roads join at this node. An intersection has a centre-position and a lay-out as well as a list of all the joining roads.
- **Path**. A path is nothing more than a connection between two nodes, thus between two intersections. It contains a list of segments that together form a road. The list is in consecutive order. This means that when the list is walked through it is just like riding on the road.
- **Segment**. A segment is a part of a road. It has a curve. When its curve is set to zero the segment is a straight road. A segment is always joined at both ends. This can be with other segments or with an intersection(s).
- **Traffic participant**. Every path holds a list with the traffic participants on that road. In this way traffic is connected to a road.

Figure 2 shows how these elements are tide together in a road network design.

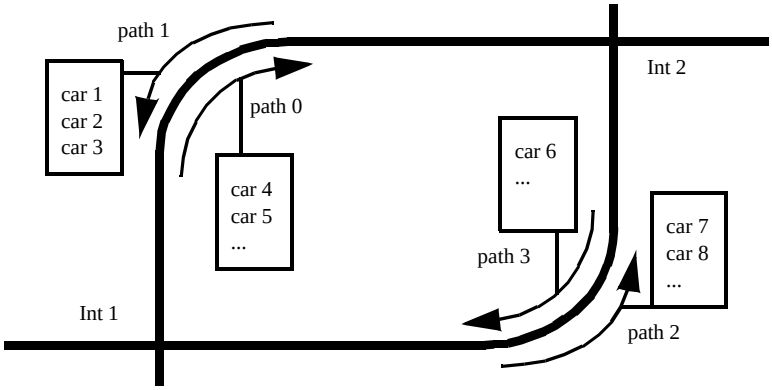

Figure 2: Road network logical design

3.3 Current Autonomous Agents

The autonomous agents that are currently in the system work strictly goal-directed. There are three goals defined which the agent always tries to satisfy:

- Prevent collisions. An agent will never collide with an other traffic participant. The basic setup will not allow the agent to collide with any object in de environment and the agent will do anything to prevent such a collision. This will otherwise disrupt normal traffic flow
- Prevent dangerous situations for others. In a simulator that is used for research with human test-subjects it is usually unwanted that situations could occur which pose a thread to a traffic participant. Situations like these will result in unpredictable behaviour and will disrupt normal traffic flow
- Prevent disruptions in the traffic flow. An agent will not manoeuvre itself in a position that might disrupt the traffic flow. It will not stop at the middle of the road, unless the scenario control says otherwise

At the highest level there is another goal defined. An agent must be able to navigate through the system. The goal can be either to go from A to B or to follow a random path. In the first instance A, B and the path between these two points are given by the scenario that is in use at that time.

The agents currently driving around in the car simulator have some major shortcomings. Three of them are:

- Very little psychological plausibility. This means the agents are not useful as modelling tools. The set-up of the agent did not need it at the time it was developed
- Hard to extent to other situations then the ones currently available. When the system is extended with for example a three-lane highway, the autono-

mous agents can not handle it. The rules the agent are based on can only act and react when driving on a two-lane highway and will probably not notice the third lane. It can be done, but it will take a considerable amount of time

- Too rigid in some situations which might cause oscillating behaviour. Some rules are too crisp. This might result in a behaviour like overtake - don't overtake. When this happens an agent will start overtaking then brake it off, start overtaken, etc. With the use of fuzzy logic in the automan these kinds of behaviours will not occur.

3.3.1 Scenarios

Scenarios are build with the Scenario Specification Language. This language enables the designers to give a precise description of the traffic flow over time. One way of control is to specify the start of a vehicle if the driver passes a point X. In this way you can for example guarantee that the new car will arrive at the intersection the same time as the driver does. Commands can also be given while the simulator is running, thus giving the designer/researcher more dynamic control of the situations. Scenarios are a powerful tool to make sure that the environment is the same, time and again. This is of great important when dealing with statistical research. The deviations in the statistical numbers most come from the driver, not from the simulator.

3.3.2 Hardware

The computer is a fast Silicon Graphics machine. It is a multi-processor Unix system with dedicated graphical hardware. This makes it very suitable for simulators like the one in use here. The car is a complete BMW 518. The motor has been replaced by two servos, which deliver counter forces on the steering wheel and gas pedal. The motor sound is generated by a high quality sound system. The projection screen is divided into three parts. The centre screen has a resolution of 1280 by 1024 pixels, the two outer parts have a lower resolution: 640 by 512. The screen is bent with angles 165 by 45 degrees. The projection itself consists of 3D models of the objects in the simulation. There are also three rectangles projected on the screen, one for each mirror. These 'mirrors' give the driver a rear-view.

3.3.3 Usability

The system can and is used in several ways. The following are a few of them:

- Determination of individual driver-characteristic in specific and controlled traffic situations. In this type of research several people are placed in the same traffic situation. The researcher then monitors the behaviour of every person to see how they act
- Testing and evaluation of onboard navigation and information systems. With this simulator systems like distance-to-car-ahead-estimators can be tested

- Driver training and selection. Like a flight simulator the traffic simulator can be used to train individuals. The simulator can also be used to see if a person is fit to drive a car
- Road-layout testing. If an institute would like to see the effects of different road-settings on the behaviour and mental state of a driver the roads need not to be build in real life. Virtual reality will do just nicely.

Part II

Chapter 4
Automan Overview

4.1 Human driving behavior

Over the years much research has been done on how humans drive. This is not a coincidence: driving is a task handled by many people and one of few tasks that is learned by instruction. Some even did a complete task analysis of this complex task (McKnight en Adams(1970,[29])).

This research indicates that it is useful to distinguish three task levels: strategical, tactical and a control level (Michon (1989,[22])). At the first (thus strategical) level plans are made on which route to take with what kind of transportation, etc. The decisions on this level are influenced by general and personal opinions and attitudes and personal circumstances.

The tactical level describes how a person behaves in the situation at that time. However, while reacting in the current situation higher level plans are still considered. So when the driver needs to decide whether to make a left or a right turn at a intersection, the strategical level is checked to see which one fits best with the route set.

The standard operational tasks like steering, gear shifting and other tasks which enable the driver to take part in traffic are on the operational control level. Most of the time, in normal traffic situations these tasks are handled autonomously. When

in a new or hazardous environment the driver will be aware of his actions. It is obvious that this level interacts with the above two levels.

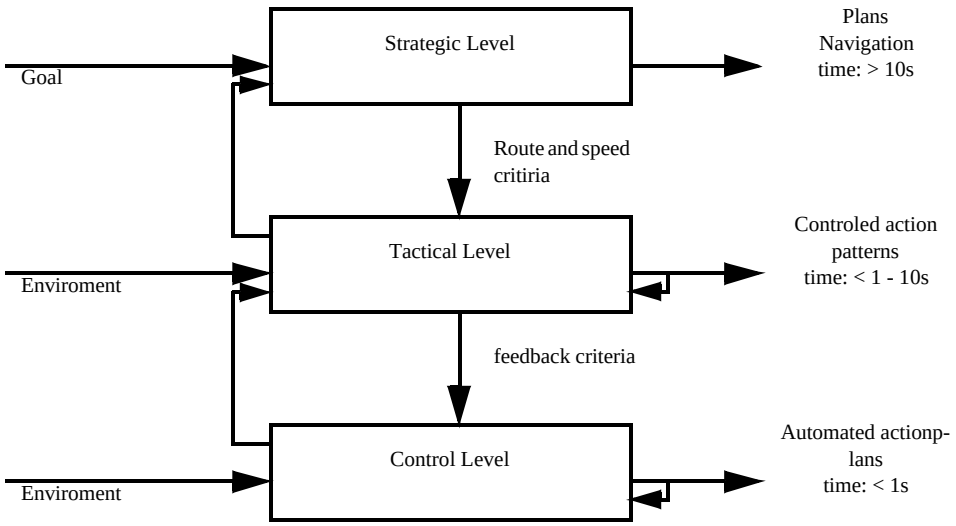

Figure 3: Structure of the driving tasks (Michon (1989,[22])).

Michon argues that a driver model need not to be incorporated all three levels. Within its own parameters each level is independent. If a theory is developed which should model human planning it is not necessary to include low-level driving tasks. This also holds for the low-level behavior-modeling: when maneuvering a car through traffic it is not necessary to have a high plan of the route to be taken. It is for example possible to make a random selection at each intersection.

Although each level is more or less independent, interaction between the separate levels is vital. If for example at an intersection the left turn cannot be made because it is temporally closed, at a higher level a new decision needs to be taken. But the downward flow is even stronger: higher levels control or define the parameters for the lower levels. The decision ‘turn left’ is made and the control level is *forced* to execute this behavior. Or if for example a route for A to B is set by the strategic level, it is not up to the tactical level to decide to go to C.

Michons argument is used to outline and describe the Automan-model. To test for example its low-level behaviors like ‘negotiate intersection’ a high level plan need not to be developed. A model of such a low-level behavior should be able to describe human behavior on an intersection without the need to consider *why* this intersection needs to be taken. This is a high-level decision and only the outcome of this de-

cision is important, not the way it come about. It gives a great deal of flexibility: each separate part of human driving can be modeled and tested independently.

With this in mind, the model of Automan can be presented by example. Michons argument implies that it should hold for any other example.

4.2 Model for Automan

In the next sections the model will be presented and explained. ‘Negotiate intersection’ will be used as an example to clarify the objects and flow of information and control through the model. Imagine the Automan is traveling on a road which ends on a T-shaped intersection. See figure 4 for the picture of the situation.

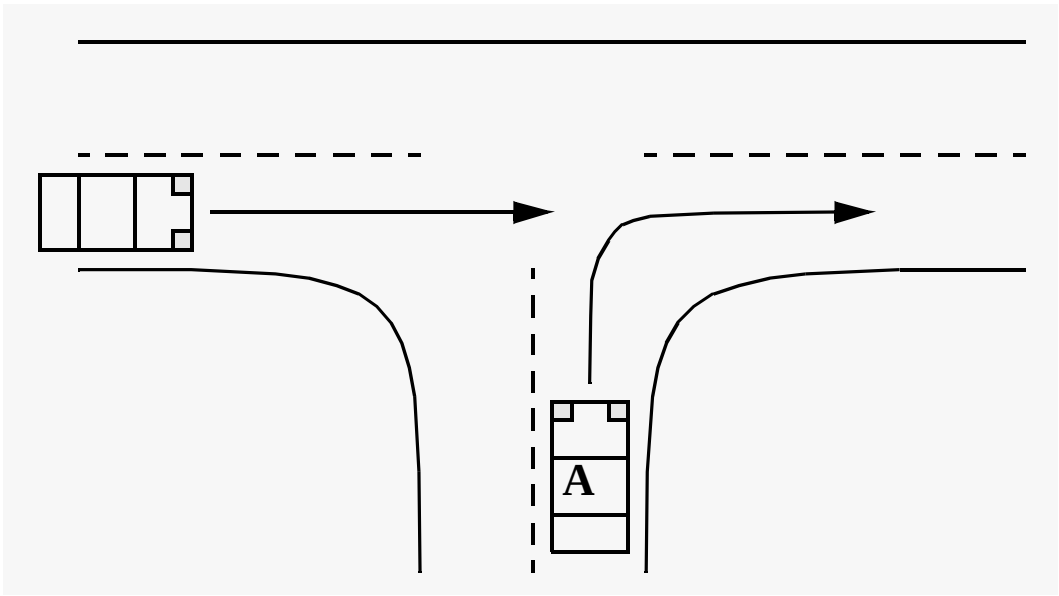

Figure 4: Example situation: Automan(A) is approaching a T-shaped intersection in dense fog

In the picture Automan is coming from the south and is about to travel to the right. From the left another car is approaching. The dots in the picture represent a dense fog. In the following sections some (but not all) aspects of the task of negotiating this intersection will be used to explain the model. Because it is a very complex task, not all relevant behaviors and decisions will be highlighted. That would make this a very long exercise, but it will not clarify the main problems. Every little part of the task needs to be evaluated when designing a working Automan, however.

In figure 5 an overview of the model is given. The grey in this picture area contains the Automan-subsystems. The other objects are environmental conditions.

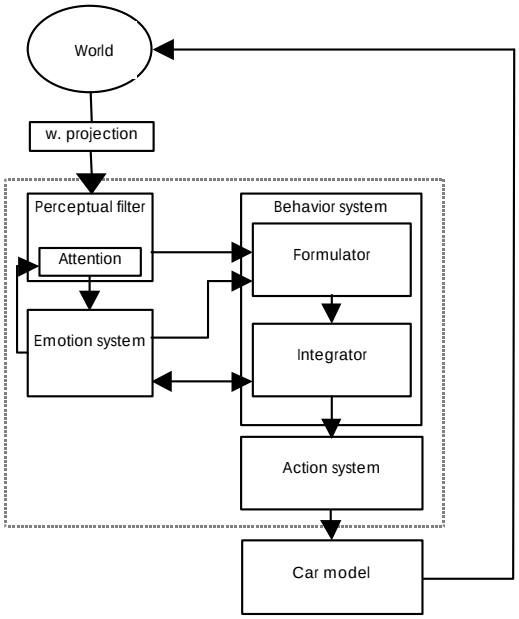

Figure 5: Model of Automan. Objects in the dotted box are part of Automan

4.3 World Projection

To interface the model with the (simulated) environment a *World Projection* queries the simulator and returns the objects in sight. This system isn't really part of the driver model, but must be included to translate the world representation in the simulator to a representation more suitable for the model. It is a kind of simulated projection screen. It determines the objects in sight, which come in two types: foveal objects, objects in the small foveal field of view of the driver, and peripheral objects, objects in the broader peripheral field of view of the driver. Peripheral objects are only seen when they are especially conspicuous. This is the case when they are moving fast, or contrast sharply with their environment. When an object is perceived in the peripheral field of view, its properties are not perceived; to perceive an object properly it must be seen in the foveal field of view.

The world projection takes into account obstructions that block the drivers line of sight (such as buildings, other cars, etc.) and weather conditions to set the vagueness of the fuzzy properties of the objects. This is done by *Reliability functions*. These are simple functions relating vagueness modification to some perceptual condition. The most convenient form of these functions needs to be determined by experiments. The effect of these functions is cumulative. What this means, is that when a car is perceived in dense fog behind another car and some tree all these elements have their impact on the vagueness of the properties of that car.

Every perceptual condition has its own reliability function. Some of these are:

- **Distance.** The further away an object is, the more difficult it is to perceive it. Therefore, vagueness is increased with distance
- **Buildings.** Objects behind a building are not seen at all. Therefore, not the vagueness of those objects is increased, but the objects themselves are not perceived at all. Of course, it could be the case that an object is partially occluded by a building. Humans are mostly able to identify an occluded object, up to certain degree of 'occludedness'. However, properties such as speed of an occluded object are harder to perceive. At this point, the best choice seems to be to define a degree of occludedness at which an object cannot be perceived. If an object is less occluded, its vagueness can be slightly decreased
- **Other Cars.** Other cars hinder perception, but objects can mostly still be seen. People can look through windows of another car, and parts of the object may not be occluded by the car. Therefore, dependent on how much of the object can be seen, vagueness increases up to a point where the whole object is behind the car, where it makes a jump to a higher vagueness level, but not maximum vagueness
- **Trucks.** Trucks totally block the view of a driver. Just like with buildings, the vagueness drops with the amount of object that is not occluded. When the whole object is occluded, it is not perceived any more
- **Weather Conditions**. Dependent on how much fog (or rain, snow or other conditions) there is, vagueness is also increased. As remarked, the vagueness increases are cumulative. Objects at a distance will already have a high vagueness, which will be further increased by the weather condition

In the example a car is coming from the left. If the Automan is looking is this direction, the car is passed through the world projection. This filter sets the intrinsic vagueness of the object (which is very low, because it is close-by). But the weather-conditions are miserable because of the dense fog. This makes the car difficult to perceive, therefore its vagueness is increased. The object is then passed to the perceptual filter.

4.4 Perceptual Filter

The *Perceptual Filter* takes as input the objects in sight, and updates Automan's working memory. This updating can be either the placing of a new perceived object, or updating the properties of an already present object that was perceived again. Every time an object is placed or updated in working memory by the perceptual filter, it will contain a timestamp with the present time. This will be used later on the determine whether properties have to be updated or whether an object was perceived too long ago.

The perceptual filter has a certain minimum time required to perceive an object, the *Minimum Perceptual Time.* An object perceived in the minimum perceptual time will have property values at maximum vagueness, which means that they are too vague to be usable. If more time is spent looking at an object, the perceived properties will have smaller vaguenesses. The vagueness decreases linearly with viewing time. We have chosen a linear function here because we could not find much information in the literature about durations of incomplete perception. It is clear that the longer one looks, the more one sees, but an exact relation between viewing time and perceived properties is difficult to obtain. A linear function seems preferable, because of its simplicity. However, later on one might conclude that another function makes much more sense. Of course, these vagueness decreases are applied after the reliability functions of the world projection have adapted the vagueness.

If the gaze direction stays the same long enough, an object will be perceived with its properties at minimally obtainable vagueness. If this is the case, a new object will be perceived. In this process, closer objects will be perceived first. If all objects in a certain direction are perceived, and the gaze direction still hasn't shifted, again the closest object will be perceived. This will result in updating its properties in working memory, dependent on the time spent looking at it. The longer the interval between successive views of an object, the greater its vagueness will have increased, and the longer it is to be viewed again to perceive its properties with minimally obtainable vagueness.

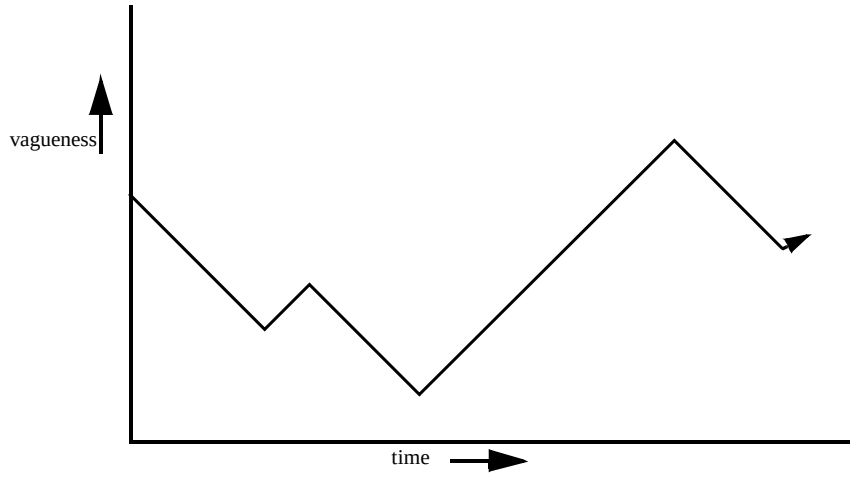

Figure 6: Vagueness of an object: a decrease means time is spend looking at the object and an increase means the object is not perceived

In the example the car is close and because visibility is poor, much time is spend looking to the left. The car is perceived with maximum time, so its vagueness will be not much greater then the value it has received from the world projection. Keep

in mind this is still a high value: although much time is spend looking, the fog makes it still difficult to see.

4.4.1 Shifting Gaze Direction

The output of the perceptual filter is dependent on the time spent looking at an object. This time, in turn, is governed by the *Visual Schemes.* A visual scheme is a sort of behavioral pattern. Its activation is based on the active behavioral patterns. Only one visual scheme can be active at a time. For instance, if approaching an intersection, the Navigate_Intersection behavioral pattern will be active. This behavioral pattern will have a visual scheme associated with it, Check_Intersection, that specifies where the driver has to look to be able to carry out the Navigate_Intersection behavioral pattern. A behavioral pattern can have more than one visual scheme associated with it. For instance, if overtaking a vehicle, it is convenient to have two visual schemes, one for checking whether overtaking can be safely performed, and one to check if, after overtaking, the driver can go safely to his own lane. Just like behavioral patterns, the activation of visual schemes can also be influenced by emotions. The visual schemes are evaluated after the behavioral patterns have been evaluated.

A visual scheme contains priorities for various gaze directions, and rules to update these priorities. The gaze direction will be set to the direction with the highest priority by the action system. For instance, if approaching an intersection, it is more important to know if there is traffic coming from the right than from the left. The rules consult working memory for perceived objects in the various directions. If, in a certain direction, enough objects are perceived recently enough, the priority of that direction will be decreased. Another direction will now have highest priority, and the gaze direction will be shifted. Take, for example, the situations when Automan is (again) approaching an intersection, and has no objects in working memory that represent cars coming from the right. This can have two reasons.

Firstly, Automan has just looked in that direction, and nothing was there. Its working memory will contain only an object for the road coming from the right, which will have a very recent timestamp, and, while looking to the right, this timestamp will be updated again and again because there are no other objects. The priority of looking to the right will then be decreased. Looking to the left will now have a higher priority.

Secondly, Automan has not looked to the right yet. Then, there will be no objects in the specific direction.[1] In this case, nothing will be done with the priorities, and the model will go on looking to the right until it has perceived enough objects. How much is enough is dependent on the situation (and, hence, on the active visual scheme). For instance, when trying to overtake a vehicle, it is enough to know there is one car on the other lane; the cars behind it may become important later, but for the present the driver has to wait until the one car has passed. When approaching a crossing, it is more important that all cars coming from a certain direction have been noticed.

How many objects is enough depends on the visual scheme. It can be the case that only a certain type of object is relevant (e.g. a traffic sign). Then, as soon as this object is perceived, the gaze direction will be shifted. On the other hand it can be the case that a complete mental picture of a certain direction is required, containing all objects (e.g. when approaching an intersection). Then the gaze direction will be shifted only when the closest object is updated again (signifying that all objects in the direction have been perceived.)

It is also possible that an object is perceived in the peripheral field of view (because it is very conspicuous). The active visual schemes will then shift the gaze direction immediately to the direction of that object, after which it can be properly perceived by the perceptual filter.

In figure 7 the various parts of the perceptual process of Automan are sketched. Strictly speaking, the shifting of the gaze direction is done by the action system.

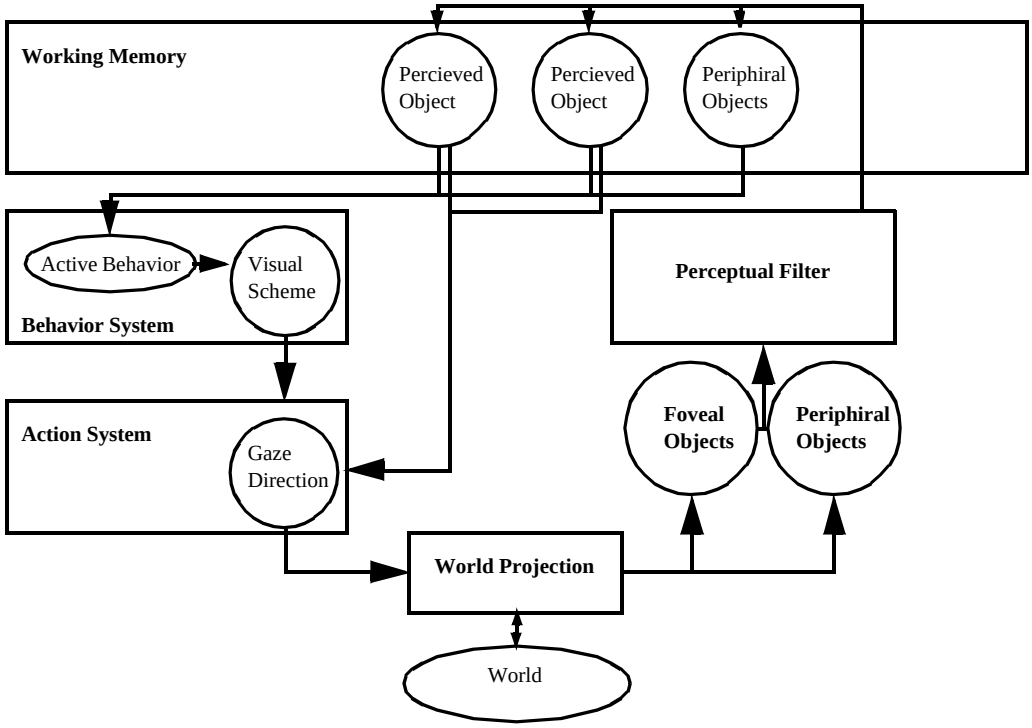

Figure 7: The perceptual process of Automan

1. Well, since Automan knows it is approaching an intersection, it probably will have an object for the road from the right. However, its timestamp will be older, so looking to the right will still keep its high priority. This is because the Navigate_Intersection behavioral pattern only becomes active at a certain distance from an intersection, while intersections are mostly perceived at a much higher distance. This means the visual scheme associated with the driving straight ahead behavioral pattern will be active a certain time after the road from the right is perceived, and will have shifted the gaze direction to other directions in the meantime.

4.5 Behavior system

The purpose of the behavior system is to activate the 'right' behaviors in every situation. The system checks working memory on active objects and behaviors to activate (or deactivate) new behaviors and objects. Internal states, emotions, motivation and expectations influence the outcome of the processes involved. Conflicting behaviors need to be resolved, like turning left or right.

4.5.1 Hierarchical structure

The behavior system is a hierarchical network of independent, goal-directed elements called behavioral patterns. The purpose of these elements vary a great deal. Some are very specific (turn on signaling), but some are more global behaviors (take next three left turns). An element can therefore be at the strategical, tactical or control level. Not only real world objects in working memory are used to set the activation of a new behavior. Already active (or in the absent of an active) behavior(s) are also used. The behavior 'turn_on_signaling' is only relevant if the higher-level behaviors 'turn' or 'switch_lane' are active. In turn, in the absent of the 'turn_on_signaling'-behavior may result in the activation of 'check_lane'-behavior. The latter will direct the gaze direction to see if the lane is clear.

The hierarchical structure becomes apparent when the following observation is made: a high-level behavior consists of one or more lower-level-behaviors. Overtaking for example consists of amongst others of 'turn_on_signaling', 'switch_lane', 'overtake', 'turn_on_signaling' and 'switch_lane'.

4.5.2 Object-based

Behavior-selection is object-based. This comes from the fact that a driver 'overtakes a car' or 'negotiates an intersection'. If the traffic situation contains more than one car for example, several 'overtake_car'-behaviors may come active. A driver can however only overtake one single car at a time. In this case the most active behavior will be selected. Because the car closest to the driver will create the highest activation, this car will be overtaken. This automatically disables any strange actions like driving through the car in front to overtake the second car.

4.5.3 Activation and success

Selection and control is done based on the current activation of a behavior. How is this activation determined? If a behavior is based solely on real world objects (like traffic-participants) activation is based on the information-value of the referred objects. If an object that perceived poorly (the car from the left in the example) the activation of behaviors based on this object will be adjusted. The activation of 'start_turning' will be fairly low, because you can not be sure where the car is and what the driver is about to do. On the other, activation of 'slow_down' will be high, and therefore shall Automan approach the intersection at a low speed.

Activation of objects will in this way propagate through the complete network, altering the activation of objects (e.g. behaviors) already in memory or setting the activation of an new object.

Success of a behavior is much more delayed then direct activation. And it is not related in such a way that high activation will give a high success. It is more time-related: if a behavior has been selected to be performed, its success-rate will increase. On the other hand, if a behavior is very active, but cannot be performed its success-rate will be decreased. In the example automan has the goals: turn right and drive fast. But due to the fog, both behaviors cannot be performed: because of low visibility he nearly comes to a stop. This decreases the success-rate of both behaviors although they are both very active.

4.5.4 Holding on to a decision

Activation of a selected behavior also influences its own activation in a positive way. This has an effect which is widely know in psychology: humans will stick to their current decision, although more appropriate decisions have arise. Automan has for example decided to start the right turn. It then notices the car from the left. If Automan had seen him in time, he may not have started the turn. But he did, so his urge to stop turning is decreased and therefore could he collide with the other car.

Another example to clarify this: if a person is driving behind a slow driving vehicle, the person will see if he can overtake. The situation does not allow it, so he does not start overtaking. If, however, the lane is free, he may decide to start overtaking. While overtaking a car approaches from the other side. If this situation occurred before he starting overtaking he may not have started the routine. But he did, so he may try to ‘go for it’ and keeps overtaking.

4.6 Action system

The action system is also a kind of decision system. It does not decide what action should be taken however, this is done by the behavior system. The action systems determines how the actions are carried out. In the example the Automan is at the point of turning right. The action system calculates based on current position, speed, etc what the next steering angle should be as well acceleration and signaling. It can be compared with the human motor-cortex: it determines, based on the current position, the next position of the arm while picking up a glass.

The **car model** of the simulator then translates the outcome of the action system in order to the car.

4.7 Memory

In the text some references to sorts of memory systems are made. The next sections will give a more detailed description of all the memory types avaible to Automan.

4.7.1 Working memory

A central part in most cognitive models is working memory. Experiments show that people can hold some piece of information ‘at hand’ and forget it in a short time span. Anderson[2] gives a good overview and explanation of the current view on various types of memory. In the context of this model, working memory is the name for the storage of all currently active objects, although it is assumed that humans have several memory types for the storage of these objects. Maybe they are functional somewhat seperated, conceptual they are all the same.

This is why in Automan all active objects are stored in working memory. This ranges from perceived objects to currently active behaviors. Working memory functions as a storage and retrieval system for the reasoning mechanisms of Automan

4.7.2 Long term memory

Long term memory holds information which can be seen as Automan’s knowledge of the world. In this part of the model, concept definitions, which tell for example what a car looks like, are stored. Long term memory is queried by the perceptual processes and reasoning systems for these definitions. Whether this information is stored in a frame-structure or semantic network does not matter. For Automan only the concept of a type of memory where all knowledge is stored suffices. In the model this specific part of long term memory is called *Declarative Memory.*

4.7.3 Procedural memory

As said before, Automan does not contain any learning mechanisms at this point. But is this not a major disadvantage? The level of driving of Automan is that of a experienced driver. With that, driving a car can been seen as a visual-motor-process. So when learning to drive a car, a person will go through the three steps in acquiring these skills (Anderson, [2]). At the last stage, the skills are stored in a procedural memory. It makes it hard for researchers to retain the information stored here, but it makes it for humans driving a car to a much easier task.

The task of driving a car consists of evaluating a lot of rules and procedures. These are thus stored in the procedural memory. Automan is equipped with a similar kind of memory. Here all available rules can be retrieved and used for reasoning and for the connecting the perceptual processes to the driving task itself.

4.8 Evaluation and updating objects

When an object is perceived, not all its properties are known. It is for example possible that not all traffic participants’ behaviors are evaluated. A driver is about to overtake a car but the lane is not clear: there is much oncoming traffic. It is not relevant to know the behavior of the tenth car, to know it is there is enough. So it would be unnatural and unwise to set all known properties for all objects. The Automan-model is equipped with a system that handles these situations. If a rule in a behavioral pattern requests a property of an object yet unknown, the evaluation

process will use the appropriate rule from procedural memory to calculate the requested value. If for example the behavior of the tenth car becomes relevant (because it is now near), the evaluation process will determine its behavior.

Some properties are set by the visual process, like distance from the intersection, speed, etc. But not every update-cycle all objects in the visual world are seen again. The objects are of course still in memory and are still needed in the behavior system. An update process makes sure that the correct values are set when a behavior element requests it. In the example where Automan is approaching the intersection, he is at this point looking to the right. But while updating the behaviors the current position of the car from the left is needed. By using its last known speed and distance and the time of the last update a new value is calculated. These old values are very uncertain and this means the estimated values based on the old observations are even more uncertain. If Automan can not be sure enough where the car is, he will look at the car again. Because of the dense fog this will occur more often than in normal circumstances.

4.9 Emotion System

Automan is developed to create a virtual human driver that behaves like you would expect from any other real human driver. An intelligent agent like this is therefore called a believable agent. One major influence on human decision making is emotion. An artificial agent with no emotions will behave 'robot-like': analytical and always correct. Hardly believable. This is the reason why Automan has emotions. It is a process not yet completely understood and hard to research. It is however an important part of somebodies personality. So if the model incorporates emotions it not only will result in a more believable agent but with different settings will also give different 'personalities'. Emotions are difficult but necessary for a good model of human driving behavior.

4.9.1 Emotions influence behavioral patterns

A good example of an emotions influencing behavior is that of aggression. If a person is aggressive, this has effect on several processes. Attention is more focused on the participant that irritates the driver most and therefore will not see everything else that happens on the road. Risk-assessment is also affected. Aggressive people tend to take more risk than non-aggressive people do. This makes that they are willing to take more risks in for example overtaking a car, although they just get a few meters ahead.

4.9.2 Behavioral patterns influence emotions

Almost no-one enters a car in an aggressive state. Along the way, some people do get more and more aggressive. This shows that driving a car influences emotions. If a person wants to overtake a car but he can't because traffic is coming ahead, he may get frustrated and perhaps aggressive if it takes too long. This aggression may again lead to more risk-taking.

4.9.3 Emotions and Fuzzy Logic

A property like ‘speed’ is defined in fuzzy logic by a membership function (see chapter 9). The standard definition can be seen as the subjective definition a person gives in a neutral state. By adding factors to the shape of this function, emotion can be modeled at a low level. By changing the factors, the starting and end point of each single membership can be shifted or can the angle of incline / decline be set. In figure 23 the standard shape for ‘speed’ is given. If for example the subject is aggressive the starting and end points of each member will be shifted to the right, hence giving the person a different sense of ‘speed’, hence giving a different outcome to the rules which use ‘speed’.

Chapter 5
Perception

5.1 Perception in Automan

In this chapter one particular part of the model will be highlighted. Perception is probably the most complex cognitive system in the human brain. Quite a number of braincells are devoted to seeing and perceiving our environment. While driving a car, this sensory information is by far the most used sensory system. For Automan, however, only the results of these processes are relevant. But then why all this attention to perception? The answer is not so complicated: it is not important how human perception comes to the conclusion that the object in sight is a car, but it is very important to know *how* the object is perceived and more over *why* the object is (not) perceived. All sorts of elements, internal and external, influence the possible outcome of how an object is perceived. Attention and visual schemes control the gaze direction of the driver and therefore why and object is perceived. The next sections will elaborate on this subject.

5.2 Perceiving the environment

Human drivers use almost exclusively visual perceptual information. Indeed, it has been argued that a large part of the driving task must be considered a visual task. We think this a little exaggerated; not only are other sources of information also used (e.g. sound, g-forces due to acceleration), but also a large part of the driving task is making decisions and controlling a vehicle. However, visual perception remains an important aspect. Because Automan should be cognitive plausible, it was decided to use visual information for its perception, just like human drivers. Sound or acceleration information is not used at this stage. The simulator does not provide sound information, and it is generally accepted that sound information is not nearly as important as visual information. Acceleration information could be fairly easily obtained, but would be mostly used as an indication of vehicle behavior. Although it is not used at this point, it would not be too difficult to include it later on.

Automan is to function in a simulated environment and has access to a complete representation of this environment. Therefore, the visual output of the simulator is not used. This would imply a range of complex pattern recognition algorithms would have to be used to filter the relevant information out of the images. This would make Automan computationally and conceptually much more complex, while adding nothing to its functionality. The goal is to model realistic driving behavior, not to model human low-level object recognition. Another problem would be, that every Automan used in experiments would have to have access to the visual

information. This poses new computational requirements to the simulator, because for each Automan a complete image would have to be calculated.

Instead, a World Projection is used, that queries the simulator about the objects in the environment of Automan. Based on these objects, it determines what objects Automan actually sees. Just like humans, Automan doesn't have a complete field of view. It has a foveal field of view, in which objects are easily recognized, and a Peripheral field of view, in which objects are perceived if they are very conspicuous. The world projection determines which objects are visible in the foveal and the Peripheral field of view. When using multiple Automen, they all would have their own World Projection which queries the simulator.

Much research has been devoted to the nature of human perception in driving a vehicle (e.g. Gale (1991,[15],1996,[16]). No clear picture emerges from this research; human perception is too broad a subject to give way to a simple explanation. However, broadly speaking two areas of interest can be distinguished: low-level Perception and Attention.

5.2.1 Perception of objects in sight.

Low level visual perception is concerned with the actual recognition of objects. In driving, much perception seems to have to do with optic flow (Berthelon(1991,[3]),Cavallo(1997,[10]),Landwehr(1991,[27])). Optic flow can be described as the apparent movement between two consecutive images. Mostly, this is determined by calculating flow vectors. For instance, when a plane of red pixels has moved a certain number of pixels, the vector describing this movement is the flow vector. These low-level calculations are not done by Automan, so it will be left at this.

Various factors influence perception. For example, when a vehicle is on a collision course, the other vehicle will seem not to move against the background. In maritime environments, this is often used as a useful trick to determine whether a vessel is on a collision course. But because of the lack of apparent movement, the recognition can be severely impaired (Santos (1997,[36])). Properties of objects can also influence perception. In perceiving curves, a slight slope in the curve (to counteract centrifugal forces) or a transition curve seems to enhance perception (Riemersma(1991,[34])). Furthermore, if an object is very conspicuous (e.g. by color or contrast), it is easier to perceive. However, this effect seems stronger if the environment also contains conspicuous objects, at least for perceiving motion (Berthelon(1991,[3])). The size of the object also influences perception. For example, there is evidence that the high accident rate by child pedestrians is not only due to the carelessness of those pedestrians, but also due to a perceptual effect which makes objects harder to perceive if they are as high as the drivers eye level (Stewart (1991,[37])). Also driving in fog, spray, or rain makes perception worse.

A problem in exactly modeling human low level perception is that many phenomena that can be observed only occur in relatively few instances. For instance, look-

ing and failing to see a child pedestrian does not occur every time one is present. Unless one would like to build a model on an even lower level, using images as input and constructing neural pattern recognition algorithms, there are no real models to describe these phenomena. Even for such low level models, it would be an open question whether they would exhibit all human perceptual phenomena. It is not at all clear whether such phenomena are due to low level perception or to higher level attentional processes.

When modeling all the perceptual phenomena in Automan, one could use probability distributions for these phenomena. A new problem arises then, because these distributions have to be reliably estimated. Not much data is available for such an estimate, and individual differences between drivers are fairly large. One could make some assumptions, of course, but the question is whether Automan would really benefit.

To make matters worse, it is not at all clear what exactly is perceived when an object is perceived. In curve perception, various parameters are used together to estimate the curve (Riemersma (1991,[34])); what perceptual mechanisms lie behind this is not known. There is considerable evidence, that drivers use an estimate of Time-to-Contact on the behavioral level when engaging in situations with other vehicles (For example, when following another car, or when approaching an intersection.). This estimate is thought to be derived directly from perception. It is unclear how this is done, (Groeger(1991,[17]),Van der Horst (1991,[20]),Stewart(1991,[37])) although recent research in playing tennis or hitting a baseball seems promising in this direction. However, there are several factors influencing this perception: expansion of the vehicle on the retina, own speed, distance of the vehicle, environment of the vehicle, and the task at hand (Cavallo (1996,[10])). The question arises whether Automan would really benefit from including complex perceptual processes. Theoretically, it is unclear whether some of the phenomena are due to perceptual or attentional processes. Most of the phenomena that can have a large impact on driving behavior can be incorporated in an attentional process (section 5.2.3). Others, for instance time-to-contact estimation, can be included in an inference process later on; the use of fuzzy logic is very well suited for exactly such a thing.

It is generally accepted that humans perceive one object at a time in traffic situations. Also, the more time spent looking at an object, the better it will be perceived. To model the low level perceptual process in Automan a *Perceptual Filter* is included. This filter receives the objects seen from the world projection, and determines which object is currently being perceived. Mostly, this object will be in the Foveal field of view. However, if a very conspicuous object is present in the peripheral field of view (e.g. because it is moving fast, is fairly big, or has conspicuous colors) the Perceptual Filter will also output that object.

The output of the perceptual filter is placed in Automan's *Working memory* (section 4.5). This working memory contains active behavioral patterns, perceived objects and information about Automan itself. It is needed because direct sensor coupling

would not be enough for Automan to function. Many behavioral patterns need information that has to be present for a certain timespan. Also, much information is used by different behavioral patterns. It is therefore convenient to have a central storage mechanism for this information. Furthermore, information about perceived objects must be accessible even if Automan is looking in an entirely different direction.

In figure 8 the flow of visual information is sketched.

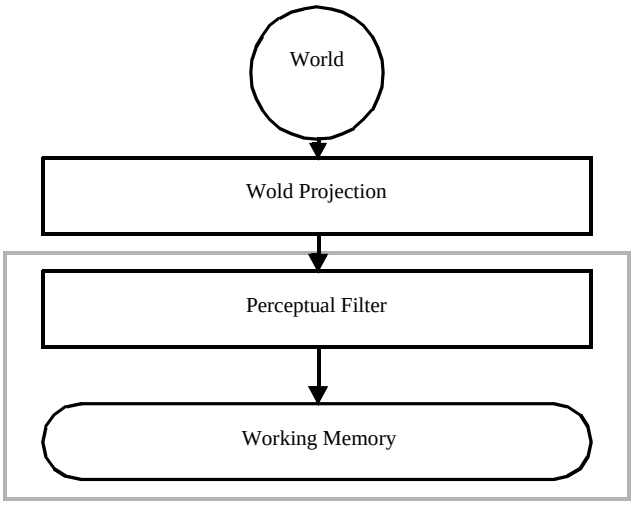

Figure 8: The flow of visual information to and in Automan

5.2.2 Intrinsic Vagueness of perceptual objects.

If a person looks at an object he can perceive its properties. However, several factors (like weather-conditions and human perception capabilities) make it that the person will not see the value of those properties completely correct. For example, at first glance a car is viewed to be driving really fast. How fast? The viewer could say 'between 100 en 175 kph.' But after a closer look (e.g. more time is spend looking at the car) the viewer may narrow it down to 'about 150 kph'. Because human vision is limited, this is as close as the person will get in assessing the value. This is called the intrinsic vagueness.

Every fuzzy object property has an intrinsic vagueness. This is its vagueness under optimal conditions. For example, this would be the vagueness when an object is perceived on a day with clear weather conditions and when much time is spent looking at it. Depending on perceptual conditions, this vagueness is increased. Both the world projection and the perceptual filter are able to increase vagueness.

The world projection modifies vagueness based on external conditions. This is done based on *Reliability functions.* Depending on the situation, these functions specify how the vagueness should be increased. These functions are cumulative; the percep-

tion of a car in fog partially occluded by a building is influenced both by the fog and the occlusion functions.

The perceptual functions modify vagueness based on the time spent looking at a certain object. The more time is spent looking, the lower the vagueness. There is a maximum perceptual time, that is needed to perceive objects and its properties at intrinsic vagueness. There is also a minimum perceptual time that is needed to perceive an object, with its properties at maximum vagueness (which would amount to not perceiving the properties at all.)

By varying the time neccessary to perceive an object properly, the perceptual function can reflect the efficiency of a human drivers perception. It is known, that experienced drivers can perceive a situation much faster then inexperienced or older drivers. This can be easily simulated in the model by increasing the perceptual time.

5.2.3 Attention control.

As remarked, attention plays a major role in human driver perception. Attention in this regard expresses itself in the direction the driver fixates his eyes. There is no clear model about how attention processes might function. In Kosslyn(1991, [25]) a model is presented that incorporates the common held views on perception and attention. In this model, information from the retina is first stored in a *Visual Buffer*. In this visual buffer, an *attention window* selects a region for detailed further processing. This processing occurs in two streams, the *what* and the *where* stream. The *what* stream processes object properties, such as shape and color, and is responsible for the recognition of the object. The *where* stream processes spatial properties, such as location and size.

These streams converge in an *associative memory*, where the properties obtained are matched with those of objects in *visual memory*. For this match, both object and spatial properties can be, and have to be, used. For example, a pencil could be identified by two parallel lines; the lines are object properties, parallel is a spatial property. If a good match is obtained, the object is recognized.

If a good match is not obtained, the attention window can be used to *look for additional information* about an object (for example, if only one line was extracted, the attention window might shift to look for another line to identify a pencil). Not only the location of the attention window can be controlled in this way. There is ample evidence that the specific properties extracted from the image in the *what* processing stream are also dependent on what is considered to be useful in identifying it. That is, dependent on one's previous experiences, processing in this stream focuses on different properties (edges, textures, colors, etc.)

Apart from this process, there is also a *stimulus-based attention shifting* process. This process draws one's eyes or one's attention window to a region of space where a novel stimulus appears. This can be detected by looking where large changes in the visual field occur. Furthermore, there is evidence from patients with brain damage that shifting attention consists of three phases: disengaging from a particular loca-

tion, moving and engaging at another location. The attention window in this model can also be used to identify objects that are not directly pointed to by the eyes, by shifting the attention window to specific regions of the visual buffer.

5.2.4 Attention in driving a vehicle

Although there is some controversy in the literature the general idea is that in driving, people mostly pay attention to the objects in the direction where their eyes are pointed, their *foveal field of view* (Cohen (1996,[11])). Just as the attention window in Kosslyn's model, the direction of this field is mostly controlled by the information needed at a particular time (Theeuwes(1991,[42],1996 [43]). The driver actively searches the scene to gather information about his environment. It seems perception of a certain object when driving depends more on whether a driver is searching for that object than on the conspicousity of it. When looking for a particular object, it is more difficult to find it when it is not on an expected location.

In driving, the expectation about where an object might be is mostly given by the particular traffic situation. For example, when approaching an intersection, the need arises to know if there is traffic coming from the other roads. This will prompt a visual search for that traffic. When there are no expectations, for example when driving ahead on a straight road, the eye movements will concentrate around the so-called focus of expansion. Therefore, when modeling this proces, a mechanism is needed that incorporates a kind of goal directed search for needed information, dependent on the traffic situation. Indeed, it seems that if drivers engage in a familiar situation (e.g. an intersection they have encountered many times), their search will be constricted. They will only look in directions in which they have experienced traffic before (Van Elslande (1991,[13])), possibly neglecting directions that would also be important. In negotiating curves, drivers exhibit a kind of sawtooth scanning pattern, which suggests they are tracking certain points (or certain textures or markings) on the curve. This depends on curve radius (Jurgensohn (1991,[21])). All this suggests that human drivers employ certain looking strategies, that function similarly to Kosslyn's attention window.

5.2.5 Gaze Direction

Research has shown, that eye- and head movements in driving are closely related (Land (1991,[26]). When a driver shifts his looking direction, this is composed of three phases: a fast movement of the eye, followed by a slower but larger movement of the head, followed by a movement of the eye to compensate for the head movement. Also, when the head direction stays constant during a certain period of time, still eye movements occur, although the fixation points will be fairly close to each other. This is presumably an attention effect, to focus on different objects or part of objects for recognition. Mostly, the direction of the eye and the direction of the head will be roughly similar.

The foveal field of view can thus be conveniently approximated by using just one looking direction, the *Gaze Direction*, which is an aggregate of both eye and head

direction. This has several advantages when trying to model looking behavior. The *where* processing stream becomes implicit in the Gaze direction. Whole objects with their properties are perceived at a time, and information about where they are is already available. If assuming only an object in the foveal field of view can be perceived, there is no need for such a stream. Furthermore, the mechanisms behind eye movements are poorly understood, and are only partly consciously controlled. Therefore, it is very difficult to make a realistic model of these movements. These movements occur very frequently and swiftly, which makes it computationally hard to model in real-time. Also, it seems that eye movements occur at a constant speed, being controlled by their duration, while head movements occur with a fixed duration, and are controlled by speed. In a digital model, it is mostly easier to use fixed durations for low level actions, because the duration of the action does not have to be calculated each time. Also, when modeling it is very convenient to keep the model as simple as possible. When one can get the same results using just one looking direction, this would be a good thing. In Automan, it is not neccessary to model precisely where an eye looks to. Because it is already known which objects are in sight, it is more convenient to use one looking direction, and define the foveal field of view to be somewhat larger than in humans (incorporating the area scanned by a few consecutive saccades). One can then treat all objects in this foveal field of view as if they were recognized by saccades, without having to model the saccades explicitly. Lastly, incorporating a visual buffer with an attention window becomes neccessary. Because the input to Automan will be already 'recognized' objects, there is no need to extract spatial and object properties out of an image. Attention can be modeled by changes in Gaze direction, that are dependent on expectations the driver has, or strategies the driver has learned. By using vagueness with the properties perceived, the perceptual filter incorporates a sort of *what* processing stream. It might be possible to include attentional effects also in the perceptual filter, but at present it is unclear whether this would gain anything.

It is known that inexperienced drivers shift their gaze more often than experienced drivers (see Aasman (1995,[1]), De Velde Harsenhorst (1987, [45] en [46])). Although a goal driven visual search seems to be an accurate description of driver looking behavior, it is difficult to devise a standard strategy used([41]). Therefore, for controlling Automan's looking behavior, we needed a way to make flexible looking strategies possible.

5.2.6 Visual Schemes

The *Gaze Direction* of Automan is controlled by *Visual Schemes.* These schemes represent the driver's expectations about a situation, and specify which directions are important to look at. For example, when approaching a left turn traffic from other directions are more important then when approaching a right turn. Given a certain situation, a certain visual scheme will be active. This scheme contains priorities for the various directions, that are updated dependent on what is already seen. In this updating process, the vagueness of properties of objects is used. If an object is seen

some time ago, the vagueness of its properties will be increased, representing the fact that the information is less reliable. (For instance, a car may have slowed down or turned, is no longer at the same position, etc.) If the properties of an object perceived in a relevant direction are too vague, the priority of that direction will be increased.

In figure 9 this is illustrated.

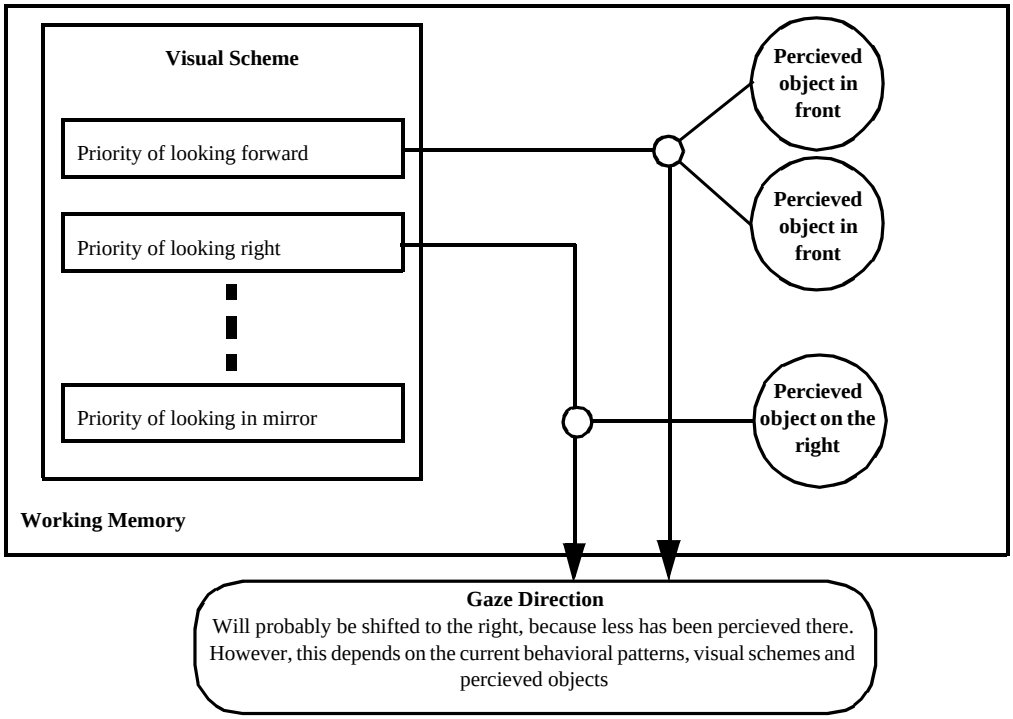

Figure 9: The determination of the gaze direction

The difference in gaze direction shifting between experienced and inexperienced drivers can be modeled in a couple of different ways. Firstly, by greater updates of the priorities in the visual schemes. This would represent that inexperienced drivers are not really confident to which direction they have to pay attention. Their experience is too limited to form good expectations about the situation; rather, they swiftly form a series of expectations, not being sure which is the right one. Secondly, by greater increases in vagueness of properties of objects. This would represent a lack of *Situational Awareness* (the ability to maintain a coherent representation of a changing situation). Drivers can have good expectations where danger might be coming from, but are not able (by lack of experience or otherwise) to keep track of their perceptions when looking at the various directions.

Also, the model can “look and fail to see”. If it takes a quick look to the right, but shifts his gaze direction swiftly for whatever reason, it can perceive things so vague they don't influence the overall behavior. This swift shifting can occur, for example,

because the perceptual filter has perceived a peripheral object (because, for instance, it is moving fast, implying a large change in the visual field). In the visual scheme can be specified that if there is such a peripheral object, the gaze direction has to be shifted immediately to that object. This can be described as a discrepancy between the driver's expectation (namely, no significant change in the periphery) and his perception, triggering an attention shift. Alternatively, one could use a peripheral object to directly influence or activate a behavioral pattern, ensuring fast reaction to a danger. Of course, this does not neccessary have to be done. Effects like tunnel vision or inability to perceive swift changes because of distractions can be incorporated by specifying a corresponding visual scheme. How to best model drivers reactions to Peripheral objects needs to be determined by experiments.

In figure 10 the whole proces is sketched.

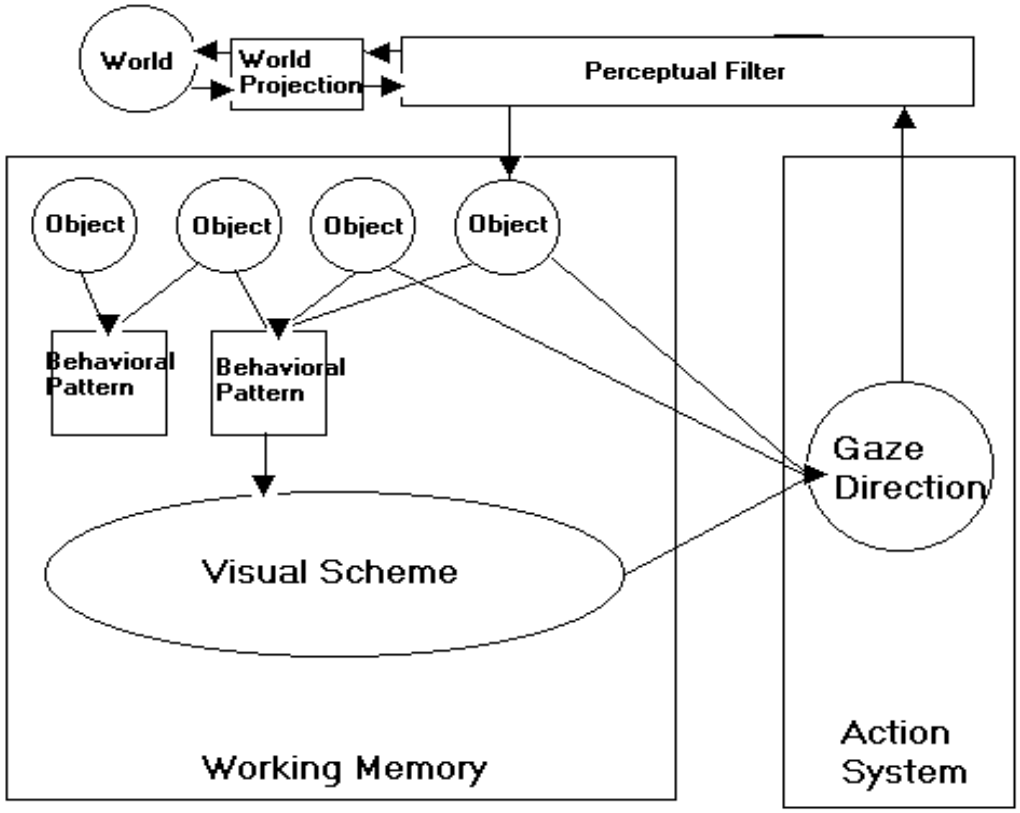

Figure 10: Automan's perceptual process.

The attention in the visual system thus consists of a time resource, the time spent looking in a certain direction. A visual scheme controls this attention by controlling the gaze direction. This control represents the expectations a driver has about the situation at hand. A driver will devote attention to directions where he expects rel-

evant information. To keep track of which information he already has and what additional information he needs, a sort of feedback loop is present. The gaze direction depends on the perceived objects present in working memory, which are dependent on the output of the perceptual filter, which is in turn dependent on the gaze direction.

A visual scheme can therefore devote all attention to a specific direction, by specifying that all other directions are not so important, or by specifying that the information about that direction has to be very recent. Alternatively, it can distribute this attention, by specifying it is only important to know some things about a certain direction, represented by a set of required objects or properties of objects. As soon as this required information is present in working memory, attention can be devoted to another direction, and the gaze direction will be shifted.

Part III

Chapter 6
Architecture for Cognition

To implement the various subsystems of Automan in a convenient, easy-to-modify way, a new cognitive architecture was developed. This architecture, although specially developed for Automan, is more generic in nature and would be useful for other purposes than controlling a vehicle. In this chapter the development and functions of this achitecture are explained.

6.1 The need for a cognitive architecture

The Behavior, Action, Emotion and Perceptual systems of Automan are all fuzzy rule-based systems. They all work with the contents of Working Memory, and some systems even work with the same objects in Working Memory. All systems apply fuzzy rules to update or infer properties of objects, or create new objects. They all have to use Long Term memory for their rules and objects definitions. Furthermore, the behavior and perceptual systems both work with vagueness. Although the systems perform fairly different functions, their inner workings are the same. All of the systems perform more or less *cognitive* functions; they manipulate the representations Automan has of the environment.

Also, if Automan is to be generic enough to be applicable in a wide range of traffic situations, the rules and objects used have to be easily modifiable. If it turns out certain rules do not perform as well as planned, it has to be fairly easy to change those rules. This implies that the rules cannot be hardcoded in the system, but must be specified in structure files that can be edited by hand.

Therefore, for an implementation of Automan it would be very convenient to have a generic system that can handle fuzzy rulesets. All systems can then be implemented by composing rulesets that define and structure the systems, and the generic system used will take care of the rest. What is needed for this is what is called a ***Cognitive Architecture*** (Taatgen (1995,[41],VanLehn(1991,[44]).

6.1.1 Overview of existing cognitive architectures

Cognitive Architectures grew out of the idea that humans are capable of performing a wide variety of cognitive functions using the same basic setup. This would imply that if one can specify this basic setup, one would have a system in which it is easy to implement various cognitive processes. An analogy can be made with a computer. If one knows the set of instructions the machine understands, one can program a wide range of applications using those instructions, from word processing to shooting down space invaders. Similarly, if one has the basic architecture of the hu-

man cognitive system, one can implement a variety of cognitive processes, for example playing chess, understanding language, or making decisions about traffic situations.

Cognitive Architectures are sometimes linked to what is called a *Unified theory of Cognition* (Anderson & Lebiere, (1998,[3])). Such a theory functions as a framework to integrate the various theories that have been developed over the years to explain isolated cognitive phenomena. The architecture is used as an implementation of the theory, in which various cognitive phenomena can be modeled.

Most cognitive architectures are based on the use of *Production Rules*. These are basically IF condition THEN action rules. The condition part of these rules are mostly compared to some information in the architecture, and if this condition is fulfilled, the action part is executed. This process is called *matching.*

A variety of cognitive architectures exist, each with its own underlying theory. In the following paragraphs a short overview of the most important architectures currently used is given.

6.1.2 ACT-R

ACT-R is the result of a sequence of models and theories based on the use of production rules in long term memory. It is an implementation of ACT*, which was proposed by John Anderson as a unified theory of cognition. ACT-R consists of two memory systems.

The *Declarative Memory* contains *chunks.* Chunks are the representation of facts. They are roughly comparable to what is called *frames* in Artificial Intelligence; they consist of a name and type, and a number of slots with additional information. (For example, a chunk for the fact that one plus one equals two would have a name like addition_fact_one, a type addition fact, and three slots, containing one, one (the addenda-slots) and two (the answer slot))

In the *Production Memory* the rules are stored. The chunks in working memory are compared with the preconditions of these rules. If there is a rule that matches the condition part of a chunk, that rule is executed. A rule can modify slot values of chunks, create new chunks in working memory, or perform a *special action.*

Not all chunks in working memory are used for this matching; ACT-R uses a *focus of attention,* which is a kind of pointer to a certain chunk. Initially, this focus will point to a chunk in declarative memory that represents the *goal* of the task. The first chunk in the condition part of a production rule has to match with a goal that has the focus of attention. The matching of the other chunks in the condition part is based on the *activation value* of the corresponding chunks in declarative memory. Each chunk in declarative memory has an activation value. This value is based on how often and how recent the chunk was used, and on the activation of associated chunks. Chunks that are rarely used will have a low activation value. Chunks that are highly associated with a chunk with high activation will also have a high activa-

tion. An association matrix between chunks is maintained for this. The lower the activation value of a chunk, the longer it takes to compare the chunk with the preconditions of the rules.

All production rules are matched in parallel. However, because some productions take more time than others to match, this process stops after a while. When this is, depends on a kind of *cost/benefit* evaluation of the current production rules. Each rule has a set of parameters associated with it, *a, b, q,* and *r. a* is the cost of execution of a rule, *b* is the average cost that has to be made after execution to reach the goal, *q* is the chance that the rule succeeds, and *r* the average chance that the goal is reached after execution of the rule. The probability of success of certain rules (*q* and *r*) times the value of the goal to be attained is the benefit of a rule. The cost of a rule is determined by the cost of the execution of that rule *(a*) and the estimated cost to reach the goal after that (*b*). The benefit of a rule minus its cost gives the expected yield of a rule. Each time, the best yield of all the matching rules is taken and compared to a probability distribution. If the probability that an even higher yield is found is low enough, the matching process will stop and the rule with the highest yield will be executed.

Rules can have *Special Actions.* These include writing something to the screen, but also changing the focus of attention to another chunk. This can be used, for instance, in an inference process where a rule places a new chunk in working memory as the result of a certain inference. By changing the focus of attention, the process can continue with this new chunk. Also, a goal stack can be maintained, which makes it possible to specify subgoals, and return to the original goal if these subgoals are reached.

ACT-R can learn in two different ways. The first way is concerned with the efficiency of know-ledge storage and representation. The learning mechanism here simply adapts the various parameters (activation values, a, b, q, r, etc.). The second way is creating new production rules. This is done by analogy. Because learning is not of concern in this context, it will not be explained further. The reader should however keep in mind that learning is a essential part of ACT-R. To fully understand and make use of this architecture, this part should not be forgotten.

6.1.3 Soar

SOAR is the result of a re-evaluation of the work done in the 1950's on GPS (General Problem Solver), a program that was to be both intelligent and a model for human problem solving. One of the developers of GPS, Allan Newell, extended the GPS-ideas in the 1980's, which led to SOAR.The central hypothesis of SOAR is that all human cognition is *problem solving.* Problem solving is searching a *problem space,* which has as dimensions various *operators.* This search is heuristic, in that the search is controlled by the estimated distance to a goal. SOAR makes goals and subgoals itself. Subgoals are made when SOAR reaches an *impasse,* a situation in which it can not decide which operator to apply. The subgoal then creates a new problemspace

with new operators. This new problemspace is now searched for the right operator to solve the impasse.

SOAR consists of two memory systems: a *Long Term Memory* and a *Working Memory.* As with most architectures, information is present in working memory in the form of so-called *WME's (Working Memory Elements).* Again, these are very similar to the frames used in artificial intelligence. It can be put in working memory by so-called *I/O modules,* that interface the architecture with its environment, or by the architecture itself. The long term memory, also called *recognition memory,* contains *operators* (production rules). In SOAR, the rules which propose an operator for a specific state and the rules which apply that operator are separated. SOAR contains a *decision mechanism* that can decide which operator to use in a certain state, if there is more than one proposed operator. This mechanisms consists of *search/control rules,* that prefer some operators above others.

Learning in SOAR is called *Chunking.* This happens when a subgoal is created and reached. SOAR will then add new operators to the Recognition Memory, which is applicable to precisely the state in which the subgoal was created. These new operators form *a Chunk.* When a similar problem is encountered, it is no longer necessary to create a subgoal, but the new operator can be applied directly.

Soar has been used to model driver behavior (Aasman, 1995,[1]). Although the results where promising, some major shortcoming were also identified. One technical shortcoming was that Soar is computational very inefficient. Other, more fundamental, shortcomings were the absence of timing mechanisms, absence of perceptual and motorical modules, and the difficulties for SOAR to forget; chunks are kept in working memory too long.

6.1.4 Other Architectures

Some other architectures exist, but are not so widely used as SOAR or ACT-R, although they are comparable to those two. For example, EPIC emphasizes perceptual processes, but is not very good at handling rulebases. Also, some attempts have been made to make a cognitive architecture with neural networks or genetic algorithms. Although these last approaches seem promising, they have not reached usable architectures yet. Lastly, some architectures are set up with a specific goal in mind, like FLAC.

6.2 FLAC: a different approach

A cognitive architecture is needed to implement the cognitive processes needed for Automan. A relatively simple architecture is needed with not much add-ons that are really relevant for the implementation of Automan. For instance, from a theoretical viewpoint the Cost/Benefit evaluation of ACT-R is very interesting, but from a practical viewpoint it is an inconvenience, because there is less direct control over the matching process[1].

The architecture does, at this point, not need learning algorithms. Later on it will be very interesting to see whether Automan can be extended to model the learning of driving a vehicle, but initially the goal is to model a driver at a certain level of experience. This level of experience can vary, but not while the system is running.

What is needed, is an architecture that can handle fuzzy logic and vagueness. As said, for the task at hand fuzzy logic has many advantages. All systems of Automan are designed using the specific benefits of fuzzy logic. To the knowledge of the writers such an architecture is not currently available.

Also, the existing architectures are not suited for applications that interact with their environment in a time-dependent, dynamical way. While all architectures work with relatively fixed goals, Automan has a constantly changing hierarchy of goals, implemented by high- and lowlevel behavioral patterns. Also, in Automan, it is necessary to keep track of the information in working memory, because most of this information will degrade over time. Furthermore, the model assumes that it is possible to evaluate rules in a certain sequence; first the behavioral patterns, then the visual schemes, etc. There is no architecture that contains mechanisms for these processes.

What is needed therefore is a new cognitive architecture. In specifying this architecture a different approach has been taken than other architectures have. We have not started from a high-level theory of what cognition could be. Instead, specifications of which cognitive functions would be needed for Automan were setup and used in designing an architecture. This architecture turned out to be very generic in nature, and not only suited to modeling driving behavior. Arguably, the result is not a true cognitive architecture; no learning mechanisms are included, and there is no underlying unified theory of cognition. However, the result is general enough to be extended to such a system.

The architecture is called **FLAC (Fuzzy Logic Architecture for Cognition).** In the following chapter, a complete description of it is given. For now a short overview will do.

In FLAC, the central module is the *Working Memory.* This memory consists of objects (which are *instances of concepts* in FLAC terminology). These objects form the current representation the architecture has of its environment and its own state. Again, these objects are very similar to the frames used in artificial intelligence. Objects can be created in working memory by a perceptual process (in this case, the perceptual filter), or by the architecture itself. In the case of objects created by a perceptual process, working memory will receive the kind of object and the neces-

1. To keep within the theoretical framework of ACT-R, one can say that all chunks that have to do with driving have a fairly large activation value, because they are used all the time. A problem would be the chunks that have to do with exceptional situations. It is a bit unrealistic to assume that deciding on an emergency brake manoeuvre would take more time than deciding to cross an intersection, because emergency brake manoeuvres are less frequent.

sary information about this object from the perceptual process and create it. The properties of the objects can be of various types.

FLAC also has a *Procedural Memory,* in which all rules are stored. Rules can be of two types: *Creation rules,* that can create new objects in working memory, and *Update rules,* that can update properties of objects in Working Memory. Finally, FLAC has a *Declarative Memory,* in which facts and definitions of concepts are stored.

The objects in working memory are continuously evaluated with the evaluation rules in procedural memory. If a creation rule matches, it can create a new instance in working memory. It is possible to specify a hierarchy of evaluation of creation rules; all rules belonging to a certain type of concept can be evaluated first, than all rules belonging to another type of concept (in Automan, these types are behavioral patterns, visual schemes, emotions and actions). If a property of an object in working memory, whose value was requested by a creation rule, is not known or (in the case of a linguistic variable) too vague, the update rule corresponding to that property will be evaluated.

A full description of FLAC and the implementation of Automan in this system will be given in the next chapters.

6.2.1 The three task levels in FLAC

In chapter 4 three task levels are presented. In FLAC it is possible to separate these levels in the way described in the last paragraph. First the behavioral patterns of the highest level will be evaluated. These behavioral patterns will use the objects currently in working memory together with lower-level behaviors active at the same moment to set new goals. Now the second level is evaluated using again available objects and the (newly) activated higher level behaviors and goals.This process is followed by the third and last level. In short: first the Strategic Level, second the Tactical Level, followed by the Control Level.

Chapter 7
FLAC Subsystems

In this chapter a detailed descriptions of FLAC will be given. In figure 11 a schematic overview of FLAC is given.

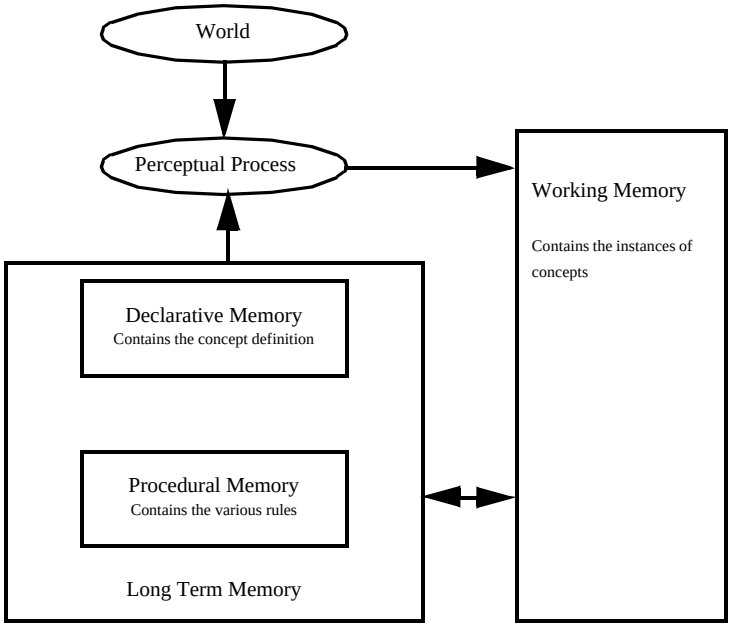

Figure 11: Schematic overview of FLAC

7.1 Working Memory

In the design of Automan a working memory is specified. This memory contains the current representation of the environment and the driver itself and is the base from which Automan functions. Because this memory plays such an important role in Automan it will also be the first part of FLAC to be designed. It is therefore the central part of the architecture (somewhat comparable to the focus-of-attention in ACT-R) from which all processing is done.

This working memory contains so called *Instances of Concepts.* A concept is *an abstract definition of a physical or none-physical object in human reasoning*. Thus, a car is a concept, but overtaking is also a concept. A concept is composed of the name of the concept and a set of properties (slots). The concepts are defined in the *Declarative Memory.* In working memory, not the concepts themselves are present, but *in-*

stances of these concepts. In an instance of a concept, the properties of the concept have specific values. For example, one can have multiple instances of the concept *car*; one with color property blue and brand property Volvo, and one with color property red and brand property Mazda. Both instances will have a property road that refers to the object representing the road they are driving on.

Instances of concepts can be placed in working memory by applying rules or by a perceptual process. Properties of concepts can have various types (see section 7.2.1). All instances and properties have timestamps specifying when they were last set or updated. If a property is a linguistic (fuzzy) variable, the property also contains a vagueness term. This makes it possible to implement the updating of old information. If a property has a timestamp that is too long ago, the property needs to be updated. Instances of concepts are removed from working memory if their timestamps are too long ago (which can be set in a structure file), or if they are no longer needed. At what time instances are no longer needed / useful, needs to be specified explicitly in the rules. For example, road instances will be removed if they are too far behind Automan.

7.2 Long Term Memory

In Automan, a lot of different rules are required. In FLAC, a system is needed that can store and retrieve these rules. Also, because FLAC works with instances of concepts, the concept definitions must be stored somewhere. While the working memory contains the instances of concepts that are currently relevant, in the *Long Term Memory* all concepts and rules are defined. Conceptually, it is convenient to differentiate between two kinds of long term memory, similar to what is done in cognitive psychology.

All rules and concept definitions are read into Long Term Memory at the initialization of FLAC. They can be conveniently specified in structure files. For efficiency reasons, the whole Long Term Memory is implemented as a huge cross-linked structure. Long term memory can not be changed at runtime yet. It is initialized at the start of FLAC with the various rules and concept definitions, and is thereafter only consulted. This might seem strange, but all processing is done in working memory. Changing long term memory would be the result of learning, which is not incorporated for two reasons: firstly, Automan is not set up as a model that can learn, and secondly, it is not a trivial matter to devise a suitable, efficient and realistic learning mechanism.

It may seem to the reader that some unrealistic choices were made here. This is not the case, however. With complex and real-time simulations speed is of the essence. Therefore if a part of the system can be made much more efficient *without* compromising the cognitive architecture, this implementation should be chosen. This is why the following ‘features’ are added to the system:

- **Use of compiler**. Computer-languages that are interpreted are much slower then compiled code. This is the case with prolog lisp and java, compared to pascal or c.
- **No on-line editing.** While the system is running, no changes can be made to for examples the rules. The program should be shutdown and restarted with the new code every time something has changes. This follows somewhat for the first choice, but it remains difficult to modify a running program.

In the future a more userfriendly system will be developed. By then, computers are much faster and will hopefully run as fast with an interpreter as contemporary computers do with compiled code. The current situation however allows rapid code-development and testing of the model. If it becomes clear that every part meets it requirement a GUI will be added as well as an interpreter.

7.2.1 Declarative Memory

In the *Declarative Memory* all definitions of Concepts are stored. Every concept has a unique *name,* and a set of *properties.* These properties can be of the following types[1]:

- **Fuzzy**. These are properties with Fuzzy Values, and are accompanied by a vagueness term and a timestamp
- **String**. The values of these properties are strings (lists of characters)
- **Number**. The values of these properties are floating point numbers (rational numbers)
- **IS-A.** This is a fixed property, and relates to another concept. It can be used to implement a hierarchy in concepts (like subclasses in C++). For example, a police car is a car with some additional properties
- **Boolean**. These are properties that can be either ‘true’ or ‘false’
- **Reference to another Instance.** For instance, a road instance will contain references to the traffic instances on it

Some properties also contain references to a rule. This is necessary because properties have to be updated from time to time. For elaboration on this process see section 7.3.

There is no limit to the number of properties a concept can have. In declarative memory only the distinctive and necessary properties are defined. Mostly, these are the properties that the perceptual process sets. FLAC has the ability to add new properties to an instance of a concept in working memory if needed. For example, if a traffic sign is observed that says the current road has right-of-way, FLAC can add the property ‘right-of-way’ to the road instance.

1. In the future, it may be useful to add others.

In figure 12 is sketched what happens when a perceptual process is outputting a new object. Declarative memory is searched for the right concept definition, and the relevant instance is created.

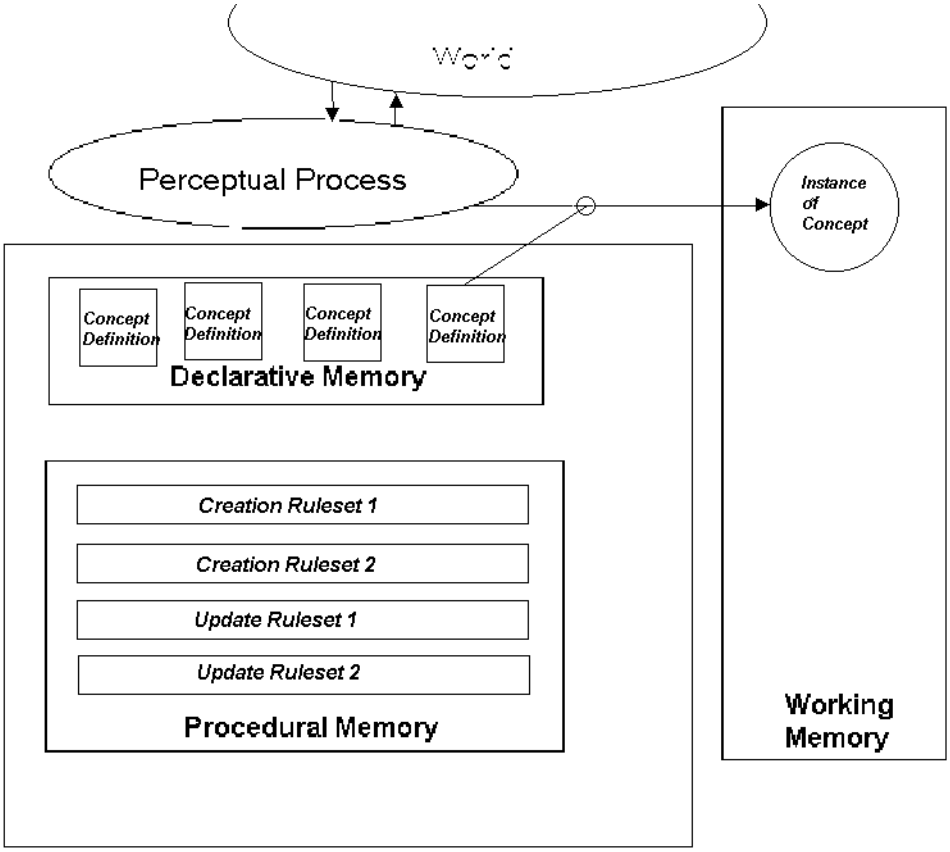

Figure 12: The creation of a new instance of a concept in wm. through a perceptual process

7.2.2 Procedural Memory

In the *Procedural Memory* the rules are stored. A rule has the general form IF condition THEN action. The *condition* part of the rule has to match with properties of instances or sometimes just the availability of an instance in Working Memory for the *action* part to be executed. Rules come in two types:

- **Creation rules.** These rules can create new instances of concepts in Working Memory
- **Update rules.** These rules can update or infer property values of concepts in working memory

Creation rules can be used to create for example a new instance of a behavioral pattern. When the preconditions of such a rule apply to the current (internal) situation it results in the creation of such a particular concept. For example: Automan is driv-

ing on a straight road. A slow car is in front of him and the road ahead is clear of any traffic. These conditions could then trigger the activation of the overtaking-behavioral pattern.

When a rule is evaluated, a property of an instance is requested. It could be that some time has past since the last request and the property is no longer up-to-date. This value needs to be re-evaluated. If it is not a property that can be perceived (because it is an abstract concept or the object is no longer in sight), the update rule is invoked to calculate the new value. For example: when approaching an intersection the property *is-crossing-intersection* of another car is requested by a rule. This car came around the corner and is seen for the first time. Therefore, the property is not known, because it takes time to perceive some if not all relevant properties (like speed). So the rule to evaluate the requested property is invoked: *if its speed is steady or increasing then the car is about to cross the intersection* and perhaps: *if the car is on the intersection then it is crossing the intersection*.[1]

In the condition as well as the action part of the rules, (fuzzy) logic operations as well as mathematical functions can be used.

7.3 Evaluation of Concepts and Rules

FLAC functions mainly by creating new instances of concepts in working memory or setting properties of these instances, governed by the rules in the procedural memory. FLAC consults working memory and tries to match the instances present with the creation rules in procedural memory. This is performed in a specific sequence.

7.3.1 Creating Instances

In every cycle, FLAC tries to match the instances in working memory with the *Creation Rules.* This matching can be done hierarchically. All rules that can create certain types of instances are processed at once. An exact sequence can be specified, consisting of which Creation Rules have to be processed at what time. In Automan, this is neccessary because the behavioral patterns must be activated before the visual schemes, after which the emotions and the actions must be updated.

1. These are only indications of how such rules may look like. Real rules can be much more complex.

In figure 13 the process of creating a new instance through a creation rule is sketched.

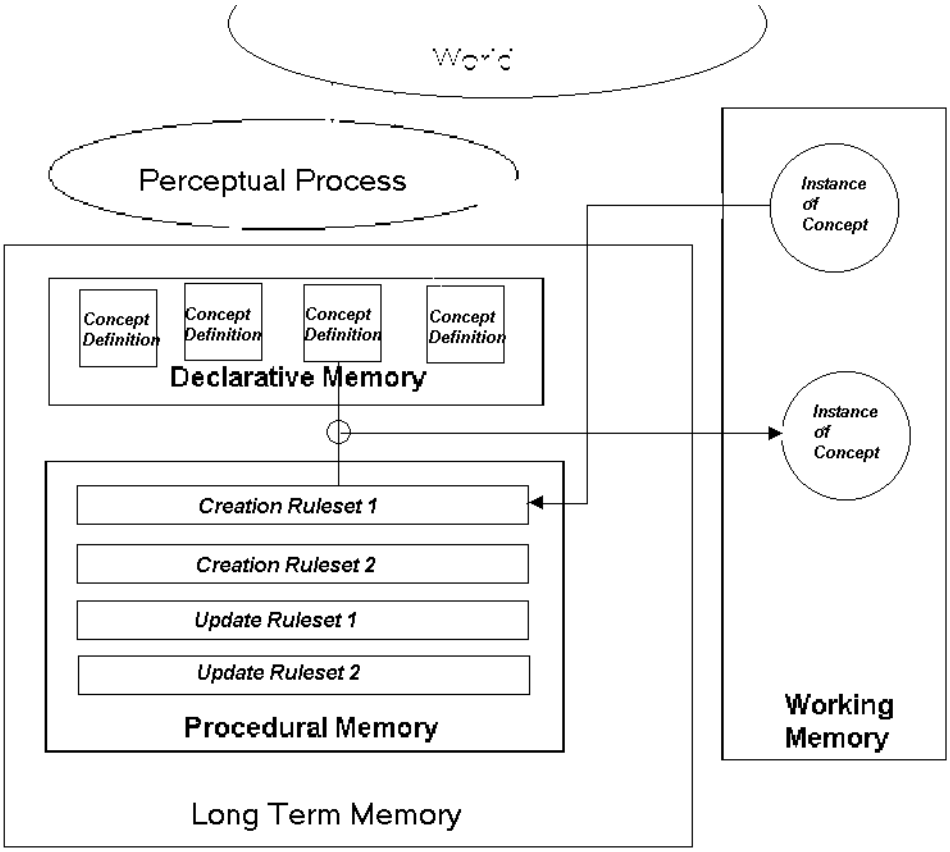

Figure 13: The first creation ruleset is consulted, resulting in the creation of an new instance

If the condition part of a creation rule matches with the properties of the relevant instances in working memory, the action part will be executed. This action part will create a new instance, and set the relevant properties. However, it could be possible that the relevant instances for a condition part are already present in working memory, but that their property values are unknown, not present or have a timestamp that is too long ago. In that case, the *Update Rules* are invoked to determine the value.

7.3.2 Updating Properties

In declarative memory only the necessary and distinctive properties of a concept are stated. In most system that use frame-structured data, a problem arises: it is unrealistic and even impossible to know all the properties in every situation in advance. And if it is possible, the structure is usually highly complex and consumes a lot of computer memory. In Flac the unique and distinctive features of a concept are al-

ways known (it should be). Other properties have to be inferred and are only inferred *when they are needed*. Also, some properties have to be updated. Whenever such a property is requested by a rule, FLAC will consult the procedural memory for an *Update Rule.* This update rule states which value the property has to have in certain situations. When the rule is invoked the timestamp of the property is reset to the current time. Update rules can also be included in the evaluation sequence. This is necessary, because some properties need to be continuously updated (emotions, activations of behavioral patterns, etc.)

In figure 14 this process is sketched. The second creation ruleset is invoked. However, a property of the first instance needs to be inferred or updated before this ruleset can create a new instance. Therefore, the update ruleset is called. This ruleset looks at the instances present to update the relevant property in the first instance. After this is done, the new instance can be created, just as in figure 13.

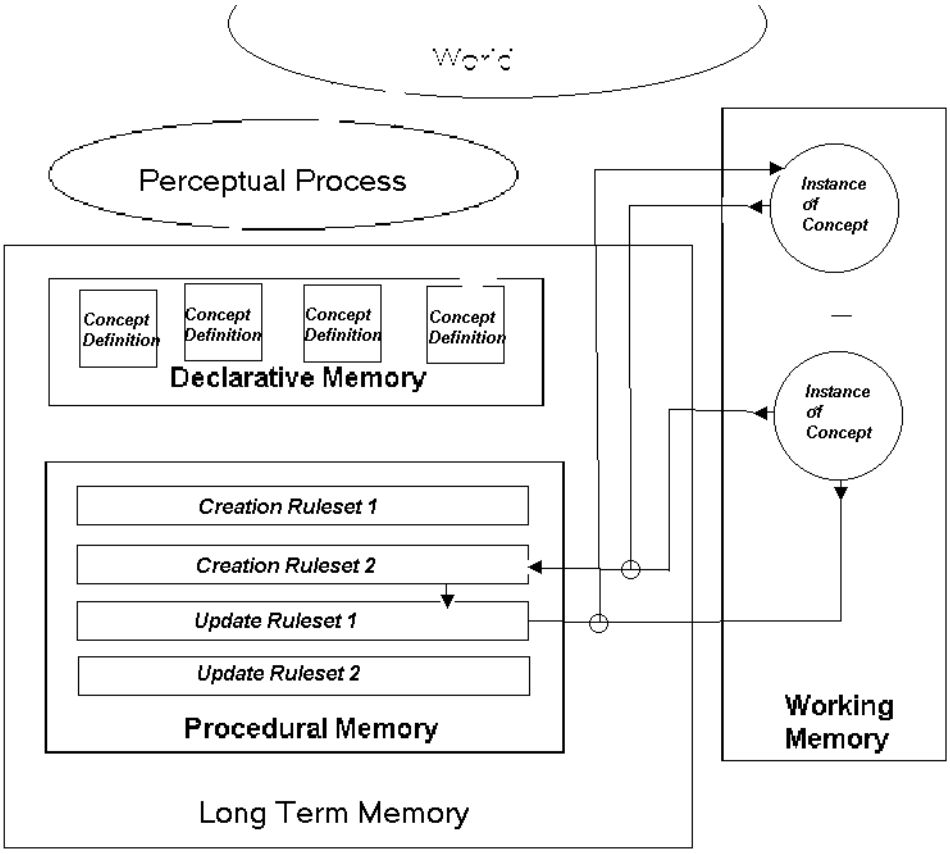

Figure 14: The second creation ruleset invokes the first update ruleset

Apart from being necessary for Automan to function, the use of update rules has another, more general, advantage. As said before, one of the problems for any logic-

based system that operates in a dynamic environment is the *frame problem:* determining which parts of the environment or its representation of it change when performing an action (see [11], [12]). With update rules, it is possible to have all property values represent the current perceived (internal) state[1]. If a property should be changed, either because the environment has changed or because new actions have been performed, it will be changed through its update rule. However, this may not be necessary. If the value of the requested property is fresh (has recently been updated) and not too vague, it is not efficient to recalculate the value. If, for example, the action of a car behind Automan driving on a highway has been evaluated and set to 'driving straight ahead' not even a second ago, it is not really necessary to re-evaluate its action again. Only if Automan is about to overtake somebody, its action becomes highly relevant and will probably be re-evaluated more often. The level and speed of this decay can be set by specifying the amounts in the concept definition.

1. The combined information given by every concept and its properties describe the *current* situation. Memorized objects have the properties set which are relevant at this point in time. With this it is possible to have a general description of a situation; objects that are relevant only at a certain point need only be known at that time.

Chapter 8
Automan in FLAC

With FLAC, the model of Automan can be implemented. Because the system is not implemented yet, it is only a rough description of how it should be done. However, if FLAC is build and Automan designed the way they are discribed, this would be the result.The reader should however bare in mind this situation. Some of the aspects in this chapter are fixed principles, others are more educated guesses. When the implementation of the model is complete every hole will be plugged.

8.1 Perception

FLAC itself does not contain a perceptual system. Vision but most of all perception of objects is a complex system. Not every necessary process in the biological mind has been identified and those which are identified are mathematical very hard to understand. At the time this document is written no computersystem exists that can match *any* higher order biological entity like humans or cats. In the first section it is argued that it is not a necessary part of a cognitive model like Automan.Therefore all information comes directly from the simulator and a separate perceptual system is designed according to the specifications laid out in section 4.4. Of course, still a world projection module has to be used as outlined in section 4.3. This perceptual system has the ability to create and update instances in FLAC's working memory. The concept definitions used for this are known in advance, so it is known what concepts can be expected from perceptual input.

The implementation of the visual schemes will be explained in section 8.3.

8.2 Memory

FLAC's memory systems correspond very well to the memory systems used by Automan (this is of course not very surprising). Therefore, Automan's memories are

directly implemented as FLAC's working memory, Automan. For instance, the declarative memory could contain the concept definition of a car:

```
Concept-Def Car {
 is-a Vehicle;
speed : Fuzzy;
color : Fuzzy;
direction : Fuzzy;
relative-direction : Fuzzy;
};
```

Figure 15: Concept definition of a car

8.2.1 A subjects knowledge about himself

In working memory the current internal situation is registered. This situation contains descriptions on and values of emotions, actions, speed and position. This knowledge is used in decision-rules to for example reposition the car or deduce the time of impact with another car. It is an always active instance of the concept Subject. Like with normal concepts, rules can update properties of Subject.

8.3 Behavior System

Behavioral patterns can be easily implemented as concepts. One "super concept", *behavioral pattern,* is defined. This concept has one property, namely *activation.* All behavioral patterns can now be specified, and have an IS-A property with as value *behavioral pattern.* The low level behavioral patterns have properties *Steering Angle, Acceleration*[1], and *Signal.* These properties will be used by the Action System to perform the actions. Higher level behavioral patterns have no direct access to these properties, because they do not directly steer the vehicle, but influence lower level behavioral patterns that do. FLAC's evaluation sequence will be set to first process all rules that can create instances of the behavioral pattern-concepts.

In these creation rules, a ruleset is created for instantiating the behavioral patterns. The rules in it specify instances and their properties that must be present in working memory for a behavioral pattern to become active. For example, suppose Automan is on a straight road with not much traffic, and is behind a slower car. This will of course be perceived and the corresponding concepts will be created in working memory. In the procedural memory will be a rule that creates an instance of the *Overtaking* behavioral pattern because of these instances. The activity of this behavioral pattern will be set to 1. This can be modified later on; how will be addressed at a later time. The real overtaking maneuver can of course only start if the road is clear. Therefore, there is a "sub" behavioral pattern *Start_Overtaking*. For this behavioral pattern to become active, two conditions must be satisfied: an instance of

1. Physically speaking, braking is also a form of acceleration, namely negative acceleration.

the Overtaking behavioral pattern must be present, and the other lane must be clear. If this is the case, a new instance of Start_Overtaking will be created with activity 1.0. This instance has properties *Steering Angle, Acceleration,* and *Signal.* These properties will be used by the Action System later on.

If the preconditions of a rule, whose action part specifies the creation of a behavioral pattern that is already present, matches with instances in working memory, the behavioral pattern will not be created again. After the behavioral pattern creation ruleset is evaluated, an update ruleset will be invoked to determine the activations of the behavioral patterns. This update ruleset looks at other active behavioral patterns and emotions and sets the activities of the behavioral patterns accordingly.

The visual schemes are implemented in the same way, only they use a different ruleset, which is evaluated after the behavioral pattern and activity rulesets. Furthermore, there is no hierarchy in the visual schemes, and no activations. The creation rules used for the visual schemes look at the most active behavioral pattern and other relevant instances and create the appropriate scheme. If the scheme is already present, nothing is done. The visual schemes do contain properties for the various gaze directions, with a numeric value that represent their priorities.

8.4 Emotion System

The emotion system is implemented as an update ruleset. Emotions are properties of the Self-concept, and are updated after the visual schemes. An update ruleset specifies on basis of what this is done. For example, if the Overtaking behavioral pattern has been active for a while, but the Start_Overtaking behavioral pattern is still not active, the frustration will increase.

8.5 Action System

The Action System consist of two update rulesets. These two rulesets specify which property of a behavioral pattern or visual scheme has to be changed and how, given certain instances in working memory. For example, a rule might be that the priority of the gaze direction to the right in a present Check_Intersection visual scheme instance will be increased if there is a Navigate_Intersection behavioral instance and an intersection instance with empty properties for traffic present.The Degrees-of-Freedom ruleset takes the behavioral pattern with the highest activation and properties for Steering Angle, Acceleration and Signal, and updates these properties, depending on instances in working memory. These values are used directly to drive the car. Also, the gaze direction ruleset takes the present visual scheme, and sets the values of the various gaze directions priorities.

For example, when approaching an intersection the Navigate_Intersection behavioral pattern will be active. If there is no traffic coming from the other roads, the Driving_Straight_Ahead behavioral patterns will be highly activated (its activation will be positively influenced by Navigate_Intersection, and it will only be created if no traffic is coming from other roads. However, a slower car can be in front of Au-

toman. In that case, the acceleration property of Driving_Straight_Ahead will be decreased.

In figure 16 this implementation is sketched.

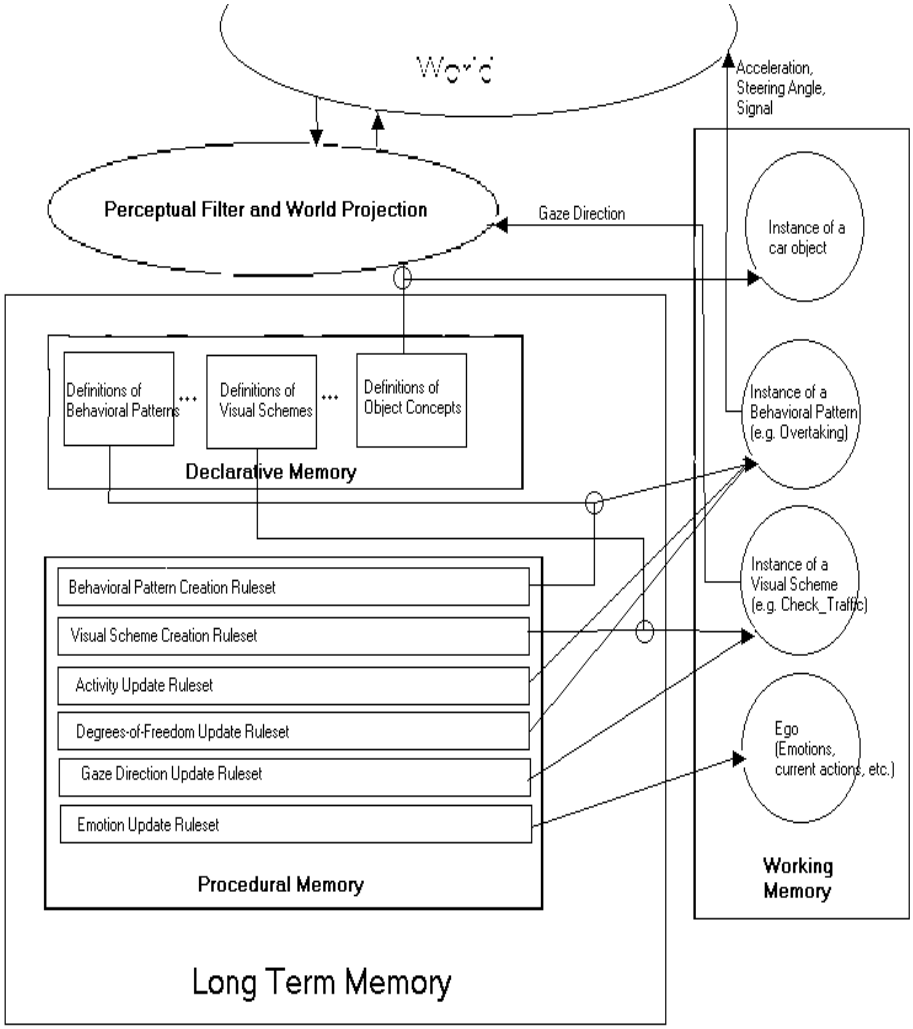

Figure 16: Implementation of Automan in FLAC. The lines represent the influences of the rulesets on relevant instances. Of course, in a real implementation, more instances would be present in working memory

8.6 The rulesets and processing cycle.

To put it all together, implementing Automan in FLAC should be done by specifying six rulesets:

- **Behavioral Pattern ruleset.** These are the creation rules that determine when a behavioral pattern becomes active. In their condition part, the rel-

evant instances and their properties are specified. Their action part calls for the creation of a behavioral pattern instance. The precise form of these instances and rules is not laid out yet. It would require some experiments to determine how to divide the driving task in behavioral patterns most effectively, and how the instances corresponding to them are made most efficiently. Conceptually, this ruleset is part of the behavioral patterns system

- **Visual Scheme ruleset.** These are the creation rules that determine when a visual scheme becomes active. Just like the behavioral pattern rules, in their condition part the relevant instances and their properties are specified and their action parts call for the creation of a visual scheme instance. Also, their precise form has to be determined by experimentation, and is also dependent on the determination of the behavioral patterns. The ruleset is conceptually part of the perceptual system
- **Activity ruleset.** These are the update rules that determine the activation of the behavioral patterns. In their condition part, the relevant other behavioral patterns and emotions are specified. When the behavioral patterns are determined, it should be fairly easy to determine which behavioral patterns influence each other in what way. They are conceptually part of the behavior system
- **Degrees-Of-Freedom ruleset.** These are the update rules that set the degrees of freedom (Steering Angle, Acceleration and Signal) based on relevant instances (Cars, their relative speeds and distances, etc.). They are called by the action system. Precise formulation of these rules should follow from the determination of the behavioral patterns; only the low level patterns, that actively drive the vehicle, should have these rules associated with them. For those behavioral patterns, it should not be too hard to define these rules. Conceptually, they can be thought of as being part either of the action or the behavior system. For the sake of the argument, they are considered to be part of the behavior system
- **Gaze Direction ruleset.** These are the update rules that set the gaze direction in the active visual scheme. They examine the relevant instances and their timestamps, and reset the priorities in the visual scheme. As with the degrees-of-freedom ruleset, their precise formulation should follow from the determination of the visual schemes. They are considered to be part of the perceptual system, although they are called by the action system
- **Emotion ruleset.** These are the update rules that update the emotions in the Self instance. Although the emotions are a fundamental part of the model, they have not received much attention yet. Precise formulation of the emotions and their influence on the behavioral patterns needs still to be done. Conceptually, these rules are of course part of the emotion system.

All concepts used by the rules have to be defined in the declarative memory. A complete processing cycle/evaluation sequence in Automan's implementation in FLAC would go as follows:

1. The **Perceptual Filter** places a new instance or updates an old instance of an object perceived in working memory, based on the objects in sight from the world projection. For example, Automan has just perceived a slower car in front of him, while driving straight ahead. A car instance with, among others, the corresponding speed property (with value 'slow') will then be created in working memory
2. The **Behavioral Pattern Ruleset** is invoked. New instances of behavioral pattern concepts are created and placed in working memory if neccessary. In the example, Automan now has a car instance in working memory, with property slow. In his procedural memory, one of the rules from the behavioral pattern ruleset will have such an instance as precondition. This rule will match, and create an Overtake behavioral pattern instance in working memory
3. The **Activity Ruleset** is invoked. The activation of all behavioral pattern instances present in working memory is determined, based on other behavioral pattern instances and emotions. One behavioral pattern instance will have the highest activation. In the example, probably also a Cruising behavioral pattern, responsible for Automan's behavior when cruising along without hindering traffic or intersections, and a Driving_Straight_Ahead[1] behavioral patterns instance, responsible for keeping the car straight on the road, are present, with a fairly high activation. For the moment, the Driving_Straight_Ahead instance is alright, and its activity will not be changed much. It will be associated with rules that brake, if the other car is too close, and while Automan hasn't looked at other traffic yet, he has to keep driving straight on his lane. However, the activity of the Cruising instance will be greatly decreased because of the presence of the Overtake instance. Automan is not cruising any more, but is preparing to overtake
4. The **Visual Scheme Ruleset** is invoked. Based on the most active behavioral pattern, a corresponding visual scheme instance is created, if not already present in working memory. In our example, because there is an Overtake instance present, a new visual scheme instance, Check_Lane_Free, is created, and the old on that was present before is removed
5. The **Emotion Ruleset** is invoked. The emotion properties of the Self instance in Working Memory are set. Well, not much emotions come into play in the example. However, one could imagine Automan's frustration increasing slightly because he is irritated by the slower car

1. It could also be a Navigate_Curve behavioral pattern instance, of course.

6. The **Degrees-of-Freedom Ruleset** is invoked. The values for steering angle, acceleration and signal are determined based on the most active behavioral pattern and perceived objects. In the example, there will not be much change yet; perhaps acceleration will be slightly decreased, depending on the distance to the other car
7. The **Gaze Direction Ruleset** is invoked. The new gaze direction is determined. In the example, because a new visual scheme is present, the direction which by default has highest priority will control the gaze direction; this will result in Automan looking to the other lane
8. The degrees of freedom are sent to the simulator, that determines what happens
9. The gaze direction is sent to the world projection, that determines the new objects in sight

This cycle repeats itself. In the example, suppose no other car is on the other lane. This will be apparent because a property of the road instance in working memory, traffic_on_other_lane, will be empty. This can be either a result of Automan's gaze direction shifting, or it might have been perceived a little time before, so its vagueness is still small enough. In any case, suppose the car to be overtaken is now nearby, a new behavioral pattern instance will be created (Start_Overtaking) by the Behavioral Pattern ruleset. Then, the activity of Driving_Straight_Ahead will be sharply decreased by the activity ruleset. The present visual scheme instance is still the right one, so no new visual scheme instance will be created by the visual scheme ruleset. Perhaps Automan's frustration will now be slightly decreased by the emotion ruleset, because he is now able to overtake[1]. Next, the degrees-of-freedom ruleset looks at the Start_Overtaking behavior; there will be rules that increase acceleration and change steering angle, so the corresponding properties will be set likewise. Then, the gaze direction ruleset will change gaze direction to looking straight ahead, to keep the car on the other lane and keep looking for traffic. And then, the cycle repeats.

As remarked, this is just a rough outline of how Automan could be implemented in FLAC. The precise form of the rules and instances have not been determined yet, just like the behavioral patterns and visual schemes to be used. Some experiments would be neccessary to determine how this could be done most effectively. Also, it would be dependent on the kind of behavior one would let Automan display, and on the situations it will encounter. However, the rulesets and processing cycle described incorporate all the characteristics of the Automan model. They form probably the most efficient and effective way to implement Automan in FLAC. Whether this is correct has to be determined by experiments.

1. Perhaps the sheer joy of the acceleration and anticipation of the overtaking manoeuvre increases Automan's happiness....

Part IV

Chapter 9
Fuzzy Logic

If the reader is familiar with computer systems like expert shells, one major disadvantage could come to mind: these systems are not really capable of human-like reasoning. This drawback comes from a fundamental property of the way the rules are implemented. The rules of systems that should be able to mimic human reasoning do that by mathematical equations and boolean logic. But this unnatural, because such systems imply perfect knowledge of the world on which the rules apply. In most cases (perhaps ever) perfection cannot be reached. Fuzzy Logic offers a great opportunity to build a new reasoning system which can make use of imperfect knowledge or vagueness. The how and why will be the subjects of this chapter.

9.1 Crisp Set and Fuzzy Sets

Classical sets as they are known from mathematical theory have crisp boundaries. This means that there exists no uncertainty on where the set ends. An element belongs to the set or not, but there is no inbetween. The following example gives a crisp definition of the set 'cylinders'.

An object A belongs to the set Cylinders if and only if:

$$\left(\frac{2}{10} < \frac{(A[surface])}{(A[height])} < \frac{4}{10}\right) \wedge A[surfaceType] = circle$$

Formula 1

According to this definition the following discrimination can be made:

Figure 17: Objects on the left are no cylinders, the objects on the right are cylinders according to formula 1 on page 63

Now we would like to build a robot that can pick-up every cylinder in a room. The first few objects in figure 17 will of course be left alone, but what about the objects more to the right? The question is: when becomes an object a cylinder? In most practical applications these crips definitions are too rigid. Fuzzy sets have no crisp boundaries. These sets are discribed by vague or ambigous properties. In this way an element has not the value 1 or 0 (1 = belongs to the set; 0 = no member of the set) but is mapped to a value ranging from 1 to 0. This value is called the membership value and will be determined by a function.

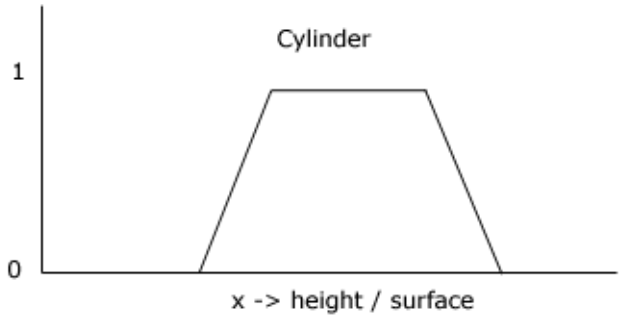

Figure 18: Membership function for cylinders

With this function, every object from figure 17 will be reevaluated. This gives the following set with for each object its membership value.

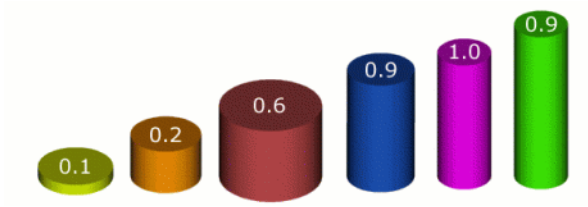

Figure 19: Objects with their appropriate membership value

At this point it is possible to program the robot with a rule like: pickup any object that looks very much like a cylinder. 'Very much' will then be defined as any object with membership value over 0.8. The major avantage for the robot is that it does not need a perfect visual system. It can make an error in detecting an object but still be able to classify it as a cylinder.

If a cylinder is flattened it becomes a disc and if it's streched it becomes a bar. But these distinctions are also fuzzy. Adding membership functions for the two definitions will produce the next figure:

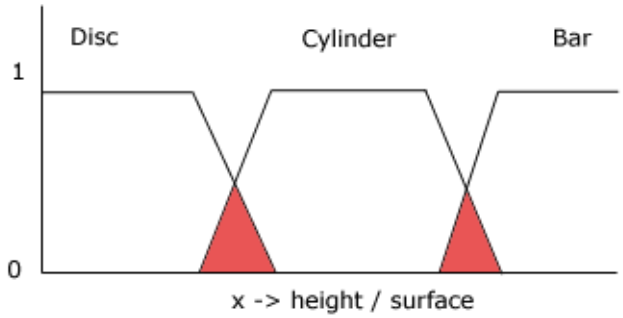

Figure 20: Membership functions for discs, cylinders and bars

9.2 Ambiguity

In figure 20 on page 65 the membership functions cross at two points. How should this be read? The red surface defines the area where an object belongs two both sets. In fuzzy logic, sets that are defined over exactly the same domain need not te be mutual exclusive. In classical logic an object is either a disc or cylinder, in fuzzy logic this need not to be the case. The orange object from figure 19 on page 65 has a membership value of 0.2 in the fuzzy set 'Cylinders' and the membership function for

'Discs' (figure 20 on page 65) will give a value of 0.8. The orange object has the following value: (, Disc 0.8, Cylinder 0.2, Bar 0.0). This type of information is called ambiguious, because one object can have more than one meaning.

9.3 Vagueness or Fuzziness

Fuzzy logic can be used to build better system, because it is more natural. But sometimes one would like to know how vague the fuzzy set really is. If it is too vague you might not want to use the set. A car has for example a direction. But if that direction is too vague one could say 'I can not say which direction it is heading.' The vagueness or fuzziness of a set can be found by a simple function. This function calculates the surface of the memberfunction and takes the ^{2}log of that value. The outcome defines the fuzziness of the set.

9.4 Statistics and Fuzzy Logic

The distiction between fuzzy logic and statistics is not always clear to some. Even books on Artificial Intelligence get it wrong[33]. Some properties of fuzzy logic may give this false impression:

- Both systems handle uncertain information
- The shape of the membership function usually looks like a statistical distribution
- Like chance, a membership function is defined on the interval [0..1]

Figure 21 shows the information world. This information can be split into two dif-

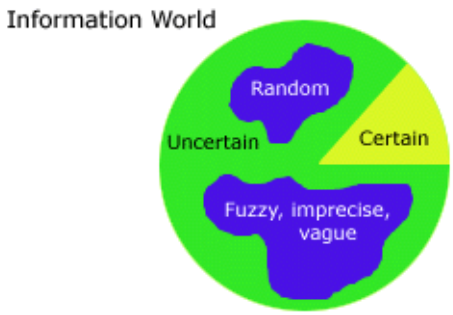

Figure 21: Information world[35]

ferent types: certain and uncertain. Most of the uncertain information is not ran-

dom, but vague. The next example should make clear what the major difference between these two is.

An experiment goes as follows: a person stands in front of you with in one hand the orange disc ( ) with membership value 0.8. Now you are asked to guess in which hand he is holding the disc. So now we have:
- chance: 0.5 of chosing correctly, but the disc has fuzzy value 0.8
You've guessed and the otherone opens his hand. The uncertainty disappears: you know for a fact if this hand holds the disc (1) or not (0). But still the disc has a fuzzy value of 0.8, it does not become 'more a disc' now you can look at it or it does not become 'less a disc' if you can't look at it.

Figure 22: An example that shows the difference between fuzzy and statistics

9.5 Natural Language

Speed is a common property associated with objects that move. But when humans reason with the speed of some car for instance, we don't say 'Wow, that car is driving 201,2 kph' but we usually say: 'Wow, that car is driving really fast'. It is obvious that this distinctions is fuzzy: when is a car driving at normal speed and when is it driving fast? Figure 23 shows a possible membership function for Speed.

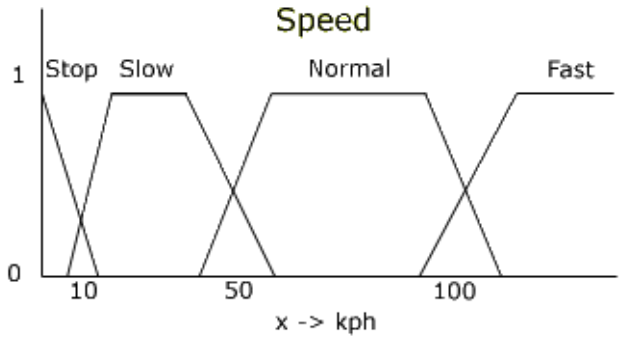

Figure 23: Membership function for speed

When it is possible to assign a distinctive name to such a set of fuzzysets this name is called a linguistic variable. Because fuzzy rules are almost ever defined by the reasoning proces of a person (usually an expert in some field) it is always possible to assign such a name.

With the use of linguisic variables it is now possible to build a system that can reason more in a way humans do. For example:

IF (Speed = high) ^ (Distance = close) THEN Brake := hard

It can be seen from this example that the outcome of a rule with linguistic variables can give an answer in the form of a linguistic variable.

9.6 Conclusion

Fuzzy logic gives developers the tool to modulate human-like reasoning with much ease. It makes it possible to write down rules in the way humans use them and not in some strict mathematical sentence.

This chapter is just an introduction in fuzzy logic. For more information on how to use it see for example (Ross, [35]). It covers the basics of fuzzy logic with some engineering applications as examples of how it can be just in practise.

Chapter 10
Bibliography

[1] J. Aasman. *Modelling Driver Behavior in Soar.* Phd Thesis, Univerity of Groningen, 1995.

[2] J.R. Anderson. *Learning and Memory*. John Wiley & Sons, Inc. 1995. ISBN: 0-471-11596-7

[3] J.R. Anderson, C. Lebiere, *The Atomic Components of Thought*. LEA publishers, ISBN: 0-8058-2817-6

[4] P.J. Bakker, S. Burry, W.B. Verwey, M.J. Kuiken, W van Winsum, P.C.van Wolffelaar, E. Webster. *GIDS implementations: the interactive traffic and driving simulator and the ICACAD.* University of Groningen. May 1992.

[5] C. Berthelon, D. Mestre, P. Peruch. *Perception of a moving Vehicle when approachin an Intersection.* In: *[15]*.

[6] B.M. Blumberg, T.A. Galyean. *Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environment*. MIT Media Lab: www.media.mit.edu

[7] V. Braitenberg. *Vehicles, experiments in synthetic psychology.* MIT Press, Cambridge, MA, 1984.

[8] R.A. Brooks. *A robust layered control system for a mobile robot.* IEEE Journal of Robotics and Automation, RA-2 (1), 1986.

[9] R.A.Brooks. *How to built Complete Creatures rather then Isolated Cognitive Simulations.* In: [44].

[10] V. Cavallo, D. Mestre, C. Berthelon. *Time-to-collision judgements: Visual and spatio-temporal factors.* In: *[31]*.

[11] A.S. Cohen, R. Hirsig. *The role of foveal vision in the process of information input.* In: [16].

[12] D.C.Dennet. *Cognitive Wheels: The frame problem of AI.* In: Z.W. Pylyshyn, The Robot's Dilemma: The frame problem in Artificial Intelligence. Norwood, NJ: Ablex publishing corporation, 1987.

[13] P. van Elslande, L. Faucher-Alverton. *When Expectancies become Certainties: A potential Adverse Effect of Experience.* In: [15].

[14] D.J. McFarland. *Animals as cost based robots.* In: M.A. Boden, The philosophy of artificial life, behaviors Oxford University Press 1996.

[15] A.G. Gale, *Vision in Vehicles III.* Elsevier Science, Amsterdam, The Netherlands, 1991.

[16] A.G. Gale. *Vision in Vehicles V.* Elsevier Science, Amterdam, The Netherlands, 1996.

[17] J.A. Groeger, V. Cavallo. *Judgements of Time-to-Collision and Time-to-Coincidence.* In: *[15]*.

[18] S. Haykin. *Neural Networks, A Comprehensive Foundation*. Prentice Hall. 1994. ISBN: 0-13-895863-7

[19] J.H.Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992.

[20] R. van der Horst. *Time-to-collision as a Cue for Decision-making in Braking.* In: *[15]*.
[21] T. Jurgensohn, M. Neculau, H.P. Willumeit. *visual Scanning Pattern in Curve Negotiation.* In: *[15]*.
[22] C.W.F van Knippenberg, J.A. Rothengatter, J.A. Michon. *Handboek Sociale Verkeerskunde*. Van Gorcum & Comp. bv. 1989. ISBN: 90-232-2269-5
[23] J. Kim. *Philosophy of Mind*. Westview Press, Inc. 1996. ISBN: 0-8133-0775-9
[24] B. Kosko. *Neural Networks and Fuzzy Systems. A Dynamical Approach to Machine Intelligence.* Prentice Hall, Englewood Cliffs, NJ, 1992.
[25] S.M. Kosslyn, O.Koenig. *Wet Mind. The New Cognitive Neuroscience.* The Free Press, New York, 1992.
[26] M.Land, J. Horwood. *The Relations between Head and Eye Movement during driving.* In: *[16]*.
[27] K. Landwehr. *Optical guidance revisited.* In: *[15]*.
[28] M.W. Matlin. *Cognition*. Harcourt Brace Publishers. 1983. ISBN: 0-15-500571-5
[29] A.J. McKnight, B.B. Adams. *Driver education task analysis*. Volume I: Task Descriptions. Final Report, Human Resources Research Organization
[30] R.F. Port (Editor), T. van Gelder (Editor). *Mind as motion, Explorations in the Dynamics of Cognition*. MIT Press. 1995. ISBN: 0-262-16150-8
[31] T. Rothengatter, E.C. Vaya (ed.). *Traffic & Transport Psychology.* Elsevier Science Ltd., 1997.
[32] A.H. Reinhardt-rutland. *Relative Visual Movement Perception applied to vehicle guidance.* In: *[15]*.
[33] E. Rich, K. Knight. *Artificial Intelligence*. McGraw-Hill, Inc. 1983. ISBN: 0-07-100894-2
[34] J.B.J. Riemersma. *Perception of Curve Characteristics.* In: [15].
[35] T.J. Ross. *Fuzzy Logic With Engineering Applications*. McGraw-Hill, Inc. 1995. ISBN: 0-07-053917-0
[36] J.A. Santos. *Detection times of a leading's vehicle motion: Effects of driving speed and road layout.* In: [31].
[37] D. Stewart. *Driver Perceptual errors and child pedestrian accidents.* In: [15].
[38] Stillings et. al. *Cognitive Science, An Introduction*. MIT Press. 1987. ISBN: 0-262-19257-8
[39] A.Steinhage, T. Bergener. *Dynamical Systems for the behavior organisation of an Anthropomorphic Mobile Robot.* In Proceedings of the IEEE International Symposium on Industrial Electronics IEEE 1997, pages 117 - 112. IEEE publications, 1997.
[40] D.L. Stewart, J. Rowland Lishman, C.J. Cudworth. *An Alternative source of Time-to-Collision.* In: [16].
[41] N.A. Taatgen. *Architecturen voor Cognitie.* 1995.
[42] J. Theeuwes. *Visual Selection: exogenous and endogenous control.* In: *[15]*.
[43] J.Theeuwes. *Visual Search at intersections: an eye movement analysis.* In: *[16]*
[44] P.C. VanLehn, K. (Ed.) *Architectures for intelligence: The twenty-second Carnegie Mellon Symposium on Cognition.* Hillsdale, NJ: Lawrence Erlbaum (1991).

[45] J.J. De Velde Harsenhorst, P.F. Lourens. *Classification of driving errors and analysis of driving performance parameters.* Tech. Rep. VK 87-17, Haren, The Netherlands: University of Groningen, Traffic Research Center, 1987

[46] J.J. De Velde Harsenhorst, P.F. Lourens. *Classification of driving errors and analysis of driving performance parameters.* Tech. Rep. VK 87-17, Haren, The Netherlands: University of Groningen, Traffic Research Center, 1987

[47] J.J. de Velde Harsenhorst, P.F. Lourens. *Aspects of Driving behavior in Learner and Inexperienced Drivers.* In: A.G. Gale (ed.): Vision in Vehicles III, Elsevier Publishing corporation, North Holland, 1991.

[48] Wolffelaar, W. van Winsum. *Driving simulation and traffic scenario control in the TRC driving simulator*. Paper ICTTP 1996: Symposium on the Design and Validation of Driving Simulators. Valencia may 1996.