Sapporo Annual Report Phase 2

March 1992

Intelligent Traffic Control for Urban Networks

Development of AI Concepts for Traffic Management Systems

Joint Basic Research Project
NTT Data Communications Systems
FZI Forschungszentrum Informatik
Dept. Technical Expert Systems and Robotics
Prof. Dr. U. Rembold

Authors
Bernd Wild, Bernd Schütze, Olaf Dalchow,
Axel Saffran and
Ron Zohar

Contents

1 Introduction

1.1 Synopsis

This report reflects the work done in the 2nd phase of the project which started in April 1991 and ended in March 1992.

Major emphasis was put on the construction of a functionable simulation environment capable of modeling urban traffic and of manipulating road network objects and traffic engineering entities in a flexible and powerful manner. To achieve this, various techniques and methods of the domain of AI, DAI, OOP and OOGUI were used.

Besides this report a number of internal papers are about to be produced which are almost concerned with the traffic control architecture and the signal plan design. They will be available in the next weeks and months. Therefore, parts of this report, which cover these fields, are not as extensive as the description of the work around the actual Sapporo prototype.

The Sapporo prototype is the first software product which emerges from the research project. It incorporates most of the work done so far and can be used as a platform for further research and development. Apart from the detailed description of the internals and principles of Sapporo, the subsequent documents have been produced:

- Sapporo Software Reference Manual
- Sapporo User’s Manual

Together with the software the documentation form the results of the 2nd phase.

1.2 Intelligent Traffic Control for Urban Environments

During the last 30 years private transportation increased to such an extent that in particular during rush hours our mobile society is turned into the opposite. By the year 2000 experts calculate a traffic jam amount which will have increased by 60%. The increasing immobility, sarcastically titled “the lust for a collective standstill” by psychologists, poses an increasing threat to the industrial society which is geared to maximum mobility. The shift of stock capacities to the roads (*just-in-time concept*) rendered the industrial production increasingly dependent on an optimal and seamless control of the traffic flow. A reduction of delays and travel times can help to avoid high costs for the national economy.

The situation has aggravated in the city centers of many capitals whose road networks have already reached their capacity limits concerning the management of the traffic volume. The enhance the networks´ performance and optimum control of the traffic streams by a traffic management system is required for an optimum management of the limited resource “transportational area”. Appropriate control strategies can reduce the delays for the vehicles, the number of stops and thus the pollutional emissions. The networking of the physical system “traffic”, which is characterized by a rapidly growing complexity and a highly dynamical structure, requires a computer-based traffic guidance system of an expandable and flexible structure. Moreover, it must provide a user interface which is capable of explaining system-internal interrelations and making decisions comprehensible for the user.

This is the context in which a research project aiming at the development of a computer-based traffic management system (*SAPPORO*) is situated; a project which is to examine the methods of artificial intelligence and their application on traffic modeling, forecasts and control. Similar to computer-based production systems for material flow design and control, which permit short development and throughput times, a computer-based traffic management system´s control strategy is expected to adopt swiftly to the short-term changes of traffic flow, in order to maximize the performance of the network. While computer-integrated manufacturing has already reached a prevalent level in various branches of industry, the efforts for corresponding traffic management systems for managing the limited resource “transportational area” were rather limited.

A fast modification of the strategies of the traffic guidance system´s control components, which guarantee optimal traffic flow, becomes increasingly important because of the rising costs for the ever growing immobility. The current load of a street network can be recorded by extensive recording systems, but it is also necessary to have a forecast on future traffic events to allow for fast reactions by the traffic guidance system in order to avoid possible bottlenecks caused by modifications of the control strategy. Furthermore, experiments for the assessment of the effects caused by decisions of the traffic guidance system´s control components cannot be made in reality. Therefore, a traffic management system must have an appropriate simulation model for carrying out experiments for the assessment of alternative control strategies. The experiments carried out in the model help to avoid wrong decisions and to optimize the strategy used to control traffic streams.

2 Traffic Control Systems

This chapter deals with the basic information required to design a traffic control system and a simulator. It gives a short survey on the methods and techniques for modeling road traffic, as they are employed in traffic engineering. Various approaches to and techniques of simulation, which form the basis for the design of the simulator, will be presented.

There are different ways to influence the intra-urban traffic by traffic control systems. The best-known method is control via traffic lights. Apart from this, traffic can be influenced by variable traffic signs (e.g. parking guidance systems, variable speed limits) or lane-oriented control signals (a lane can be alternately assigned to different traffic streams). The following will focus on traffic control via traffic lights. A survey on the methods and control strategies of existing traffic control systems presented in this paper can be found in [Lapierre et al. 87], [Wiedemann 91] and [Wild & Berning 91].

2.1 Traffic Lights and Signal Plans

Since the first installation of traffic lights in Westminster, London 1868 [Webster 66], the control strategies, the system complexity and the fields of application of traffic lights have been continuously developed and enhanced. In contrast to non-signaled junctions, traffic lights are not only to increase traffic safety on junctions by means of disentangling colliding traffic streams, but also to increase the performance of a junction concerning traffic flow by giving priority to individual streams while blocking colliding streams. The basic principle of traffic lights control in road traffic is the alternating priorisation of colliding traffic streams at junctions.

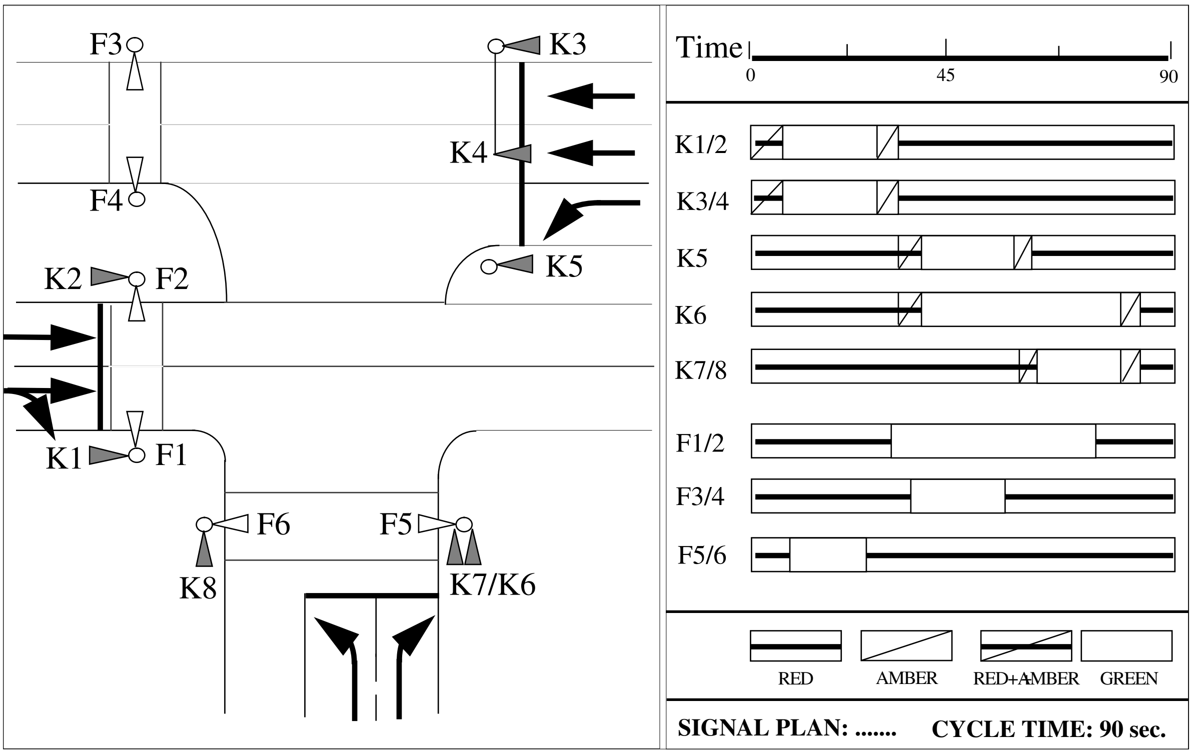


Figure 2.1 Geometry and signal plan of a junction

The signal plan forms the basis for controlling the traffic lights of a junction (Figure 2.1). It defines the duration of the stages at a junction and displays the operation of the signals of each signal group, i.e. the various green/amber/red/red-amber phases for traffic streams and green/red phases for pedestrian streams according to a time axis in seconds. The time which passes before the individual signal groups are triggered again, is referred to as cycle time of the signal plan. Obey the following restrictions/guidelines for the design of the signal plan for traffic lights at a junction.

- The guidelines for traffic lights (RiLSA) defining minimum green times, safety times which must be obeyed, duration of the amber times, conditions for transition between signal plans and other regulations which guarantee traffic safety. These guidelines are detailed in [RiLSA 91] and apply to the German traffic network. These regulations are country-specific.
- The junction´s geometry which defines the minimum offset of green intervals of two colliding traffic streams. The minimum intervals between leaving and entering streams, which are to guarantee that the two stream do not meet in the junction during phase rotation, depend on the distance between the corresponding stop line and the intersection.
- The size of the involved traffic streams, which determine the sharing out of the total green time to the involved streams. A major stream will usually be assigned a longer green phase than a minor stream.

As the traffic streams of an urban traffic network are particularly time- and weekday-dependent, e.g. morning peaks of an increased inbound traffic and evening peaks of an increased outbound traffic, the signal plan must shift accordingly and adapt to the current traffic load, if it is to guarantee an optimum traffic flow at any time.

2.1.1 Signal Control Modes

The trigger method of traffic lights differentiates between fixed-cycle signaling, time-plan- and traffic-dependent signal control modes (Figure 2.2).

When operating in fixed-cycle mode each signaling system has an own fixed signal plan, whose phase sequence and duration are time-invariant and independent of traffic load. The original signal control mode is extremely inflexible concerning large variations in traffic load. Only in the mid-1960s fixed-cycle signaling was replaced by time-plan- and traffic-dependent control modes.

The time-plan-dependent signal control mode has a fixed number of preassessed signal plans for each junction with its various characteristic traffic loads. The signal plans will be used according to the traffic situation. The switch-over intervals between every two signal plans are predefined according to a preceeding analysis of the day- and week-specific profiles of the traffic volume of this particular junction. This means that the regularities of traffic flow must have been determined by extensive measurements. This results in the assessment of when to activate which signal plan.

Traffic-dependent signal control modes require the use of detectors, e.g. induction loops integrated to the lane, which determine the current traffic volume at the junction. Usually, it suffices to measure one or few main traffic streams in the junction legs, i.e. in the approach roads of the junction. The resulting information on traffic load is used to determine the optimum signal plan.

In this context it is possible to differentiate between the three methods of signal plan selection, signal plan modification and signal plan generation.

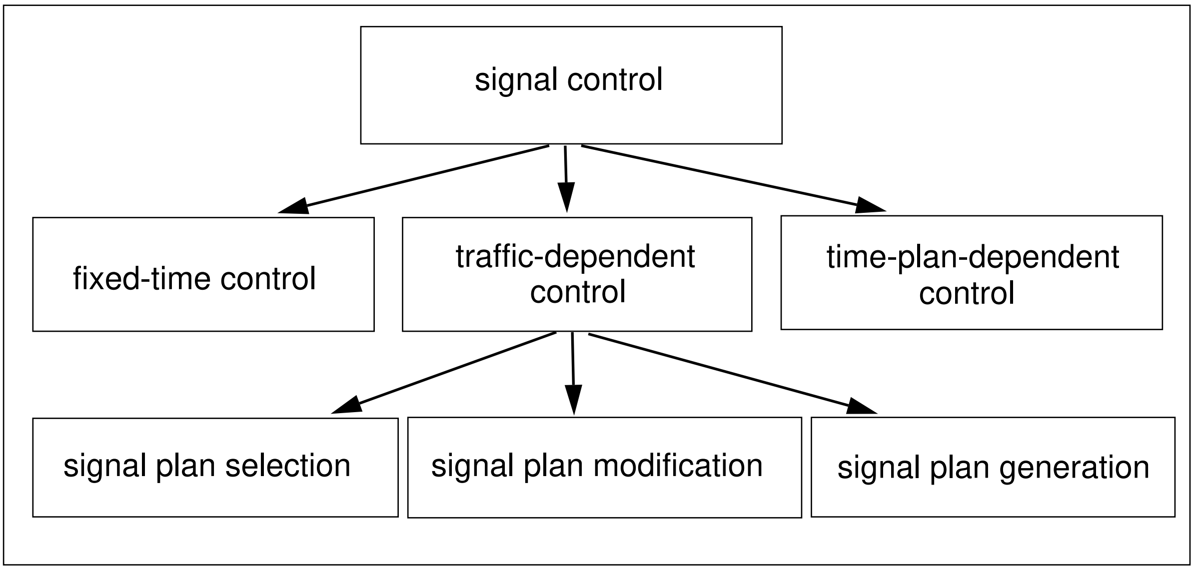


Figure 2.2 Signal control modes

2.1.2 Signal Plan Selection

There are several pre-assessed signal plans for various characteristic traffic situations. In traffic-dependent signal control mode a selection program automatically determines the plan best suited for the traffic volume assessed by the detectors.

There are special switch-over programs for switching between two signal plans, which avoid illegal or unreasonable traffic situations during the transition between two signal plans. If a signal plan was directly linked to another, this could have the consequence of too short safety times between the green phases of two colliding streams or of green times shorter than the stipulated minimum green times. The switch-over programs grow ever more complex, and the switch-over intervals ever longer the larger the signaled junction is.

2.1.3 Signal Plan Modification

Starting from a signal plan frame program, the individual elements of the signal plan can be short-term modified to the current traffic situation. The subsequent options exist:

- Extension or reduction of individual phases. Each modifiable stream requires a detector. Depending on the observed current traffic load, the entire available green time can be individually distributed to the various traffic streams, i.e. after each cycle the duration of the green time for each traffic stream is modified to the current load. The order of the phases and the cycle time remain unchanged.
- The integration or skipping of individual phases. At every intersection where not every cycle requires a green time, this green time can be attributed to other traffic streams. This phase will be only integrated on request when needed, e.g. at pedestrian crossings or traffic lights for cyclists with a push-button, priorisation of public transport vehicles, rescue services or fire brigades on separate lanes with special phases. The cycle time remains unchanged.

Signal plan modification and signal plan selection are frequently combined. In this context the frame program for the modification by a time-plan-controlled signal plan selection is determined and the individual duration of the phases of this signal plan is remotely set according to the current traffic load via the traffic-dependent signal plan modification. This is the most wide-spread combination in existing traffic guidance systems.

2.1.4 Signal Plan Generation

The method of signal plan generation does not know a true cycle time. All control elements such as start and duration of green times and the order of phases can be selected at wish and are directly influenced by the individual road user. The cycle time is only defined by the minimum green times, the maximum red time and the stipulated safety times. The signal plan generation is characterized by a high algorithmic complexity and an enormous complexity concerning the recording system (numerous measurement sites and transmission paths). Each road user approaching the junction signals his arrival via a detector. An optimization algorithm decides upon the optimum phase sequence, phase switch-over time and phase duration according to the order of detector messages and the number of messages from each direction. The signal plan generation inevitably turns into fixed-time signaling during peak traffic loads and is therefore more suitable for junctions with a low traffic volume or for low traffic volume programs. This method used by *PRODYN* [Henry et al. 83] is still in the experimental stage and largely unknown.

2.1.5 Optimizing Criteria

Apart form signal plan selection, the extension and reduction of signal phases or the integration or skipping of phases in the signal plan modification and the individual control of one junction by signal plan generation, a traffic guidance system can additionally influence the traffic via coordination of adjacent traffic lights. It is for instance possible to generate a *green wave* by synchronization of the green intervals of an aterial road via an appropriate offset. This results into an additional increase of road capacity, as deceleration and acceleration times are avoided.

The use of all these control methods is determined by the control strategy of the corresponding traffic guidance system. The control strategy aims at optimizing the traffic flow according to an optimum criterion.

Possible relevant aims for signal control strategies:

- minimization of the number of stops in the traffic network
- minimization of the average delays in front of traffic lights
- minimization of the length of traffic jams
- minimization of the average travel time through the network
- optimization of the load ratio at junctions
- minimization of ecological damages caused by fuel consumption and emissions

2.1.6 Look-ahead versus Follow-up Optimization

The above traffic-dependent methods for controlling traffic lights via signal plan selection, signal plan modification and signal plan generation are based on a follow-up optimization of the signal plan depending on the traffic loads currently observed detectors. The signal plan is modified or a switch-over program is started when a detector signals a bottleneck or a relevant change in traffic load. Concerning the signal plan selection with its long switch-over time in particular, but also the signal plan modification it is possible that the cause of the bottleneck has already vanished immediately after the switch-over has occurred, as the switch-over can only start at the end of the current signal plan at the earliest. The modification of the signal plan is therefore performed with a certain delay to the traffic requirements.

A better performance of the signalized junction could be achieved, if it was possible to recognize bottlenecks and their expected duration in advance, and if the signal plan was modified at the very moment the traffic peak hits the junction. One way to predict traffic loads is the simulation of traffic development. This requires a minimum of forecasting of the traffic development from the currently active signal plan to the point of time when an immediately induced signal plan modification could have its effect, usually an interval of five to ten minutes. According to this simulation, the simulated traffic situation will be assessed for signal plan modification in the future.

2.1.7 Signal Plan Determination of Sapporo

Figure 2.3 shows a schematic graphic of road traffic control via the traffic guidance system ***Sapporo***. The traffic guidance system controls the traffic flow in the physical traffic network in an outer control loop according to a defined optimum criterion. The measurement data are sensor data observed at various sites in the traffic network. The control data of the traffic in the control loop form the signal plan for the various traffic lights. The behavior of the drivers affects the development of the traffic flow as an interference of the physical system.

The processing of the traffic data in the traffic guidance system has been organized as an internal control loop. After evaluation of the data observed by all sensors, the current traffic state is mapped onto a model of the traffic network. Starting form the modeled current traffic situation and the currently valid signal plans of the traffic lights, the simulation of the traffic flow generates a prognosis of the expected traffic situation in the near future. Then, a signal plan determination component, which acts as an internal control element, tries to reach an optimum traffic condition according the simulation via an analysis of the forecast traffic situation and an involved variation of the valid signal plans. The iteration of the simulation and signal plan determination can be repeated several times until an optimal signal plan configuration has

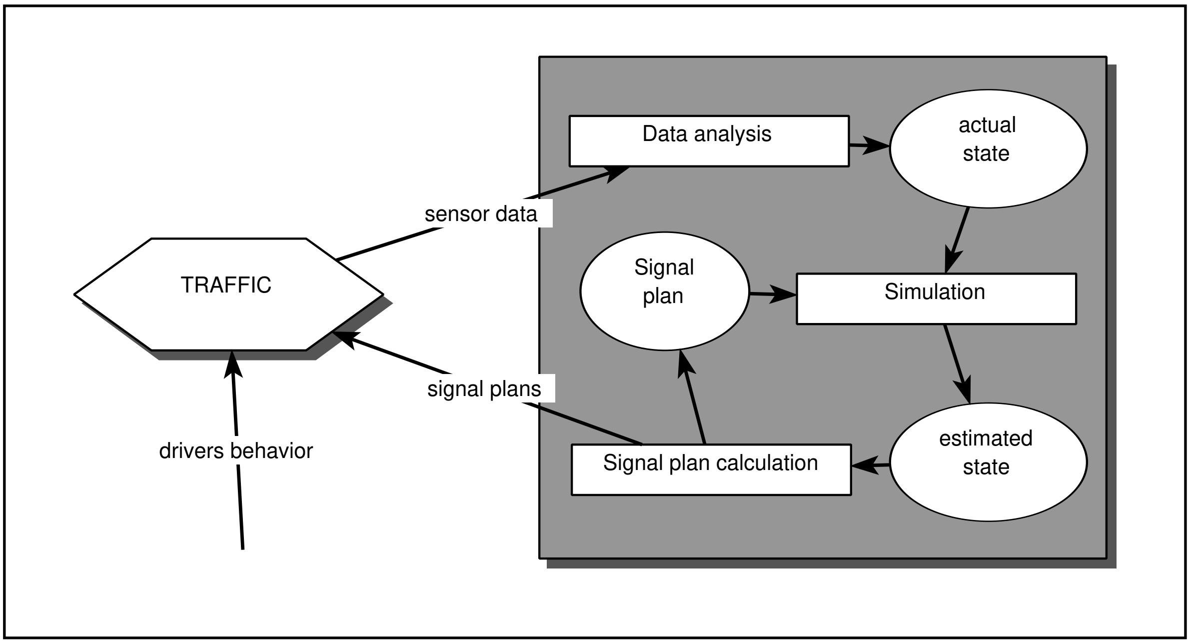


Figure 2.3 Control loop for traffic control in the traffic guidance system ***Sapporo***

been found. The result is then transmitted to the traffic lights of the physical traffic network. For signal plan determination the above presented methods of signal plan selection, signal plan modification and generation can be used. In the first prototype of the ***Sapporo*** system the method of signal plan selection is tried.

3 Traffic Models

3.1 Shock Waves

The macroscopic model by Lighthill and Witham [Lighthill & Witham 55] is based on the ***continuum theorem*** known from hydrodynamics. To put it simply, this theorem says that the total of all state changes equals zero in a continuously flowing medium in an infinitesimal small time-distance interval (in short, mass can neither disappear or appear in this trajectory). Applied to traffic flow, this theorem regards the movements of the traffic stream as a continuous flow event in analogy to the dynamic performance of fluid media. Therefore, the assumption of a sufficient density of the corresponding traffic flow is of importance. Another basic assumption states that the interrelation between traffic volume q and density k, which is depicted in the fundamental diagram, is valid both for the stationary traffic flow (independent of time and place) and the instationary traffic flow.

If $q = q(x, t)$ and $k = k(x, t)$ are continuous and continuously differentiable functions of place x and time t, $q = q(x, t) = q[k(x, t)]$ applies, where $q = q(k)$ is empirically determined on the basis of the stationary interrelation between traffic volume and density.

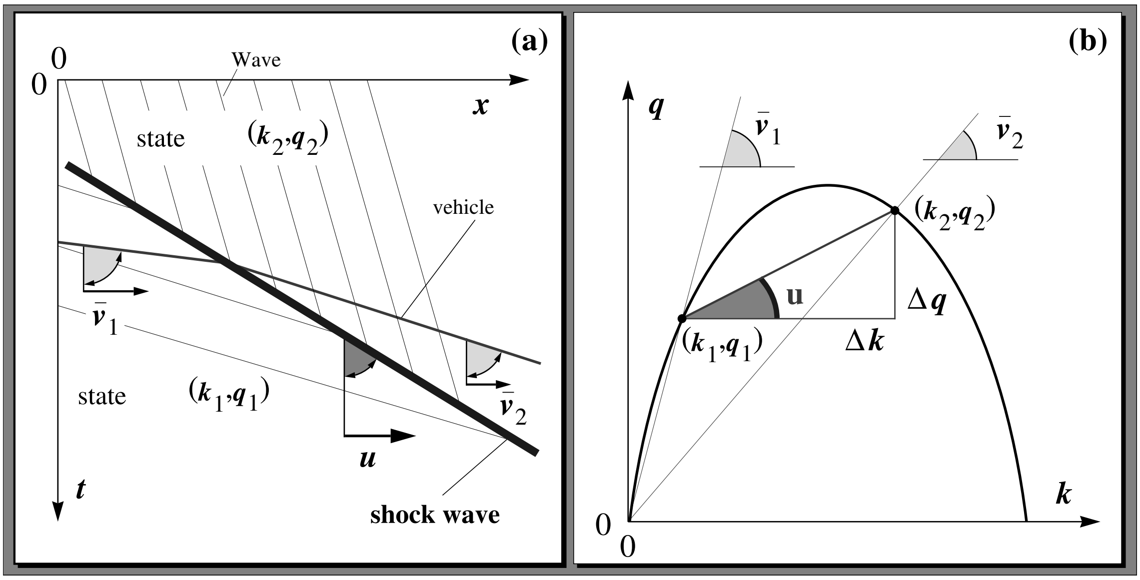


Figure 3.1 Movement of the shock wave: (a) Depicted as time-distance diagram, (b) Determination of the propagation speed in the fundamental diagram

Regarding a tiny street segment Δx in place x, which contains some vehicles, the resulting net rate of cars using this street segment at time t is

$$q(x, t) - q(x + \Delta x, t) \sim (-\Delta x) \frac{\partial q(x, t)}{\partial x}$$

and equals the modification rate of the number of vehicles in the street segment Δx:

$$(-\Delta x) \frac{\partial k(x, t)}{\partial t}$$

The vehicles remain in this segment as no traffic is added or disappears.

The ***continuum equation*** of traffic describes this.

$$\frac{\partial k(x, t)}{\partial t} + \frac{\partial q(x, t)}{\partial x} = 0$$

By means of the above equation and $q = q(k)$ a model for q and k can be deduced which analyzes the modifications of the parameters q and k, regarded as average values, in the x-t dimension(Figure 3.1(a)).

The assumption that the traffic volume q depends exclusively on the density k allows to deduce that [Leutzbach 88], changes in traffic stream propagate as kinematic waves of the same density at the propagation speed c.

$$c = \frac{dq}{dk} = \bar{v}_m + k \cdot \frac{d\bar{v}_m}{dk}$$

If a traffic stream characterized by a high density follows one of a low density, the waves of high density will catch up with the low-density waves, generating discontinuities at the interfaces. According to the continuity theorem these state modifications move at the speed u along a ***shock wave*** with the traffic stream. The shock wave, which corresponds to the propagation speed of the state modifications, separates two adjacent traffic situations: the situation in front of the shock wave of the traffic volume q_1 and the density k_1 from the situation behind the shock wave of the volume q_2 and the density k_2 (Figure 3.1 (a)). In this case $q_1 - uk_1$ is the arrival rate of the vehicles at the shock wave. The rate of vehicles departing from the shock wave is $q_2 - uk_2$. Thus, on the basis of the continuity equation (cf. above) the ***shock wave speed u*** results.

.

$$u = \frac{\Delta q}{\Delta k} = \frac{q_1 - q_2}{k_1 - k_2}$$

As the traffic states (k_1, q_1) and (k_2, q_2) comply with values in the fundamental diagram, the shock wave speed u can be deduced from the gradient of the secant intersecting the two points (Figure 3.1 (b)). If u is positive, the shock wave propagates along the traffic stream. If it is negative, the shock wave moves contrary to it. If $u = 0$, the wave is of a stationary type.

3.2 Qualitative Modeling

The already presented quantitative traffic models require a profound analysis of the applied parameters. These values must be precisely specified and mathematically represented, i.e. no distinction is made between certain and uncertain knowledge. As the knowledge is presented in mathematical equations describing the relations between these values, it is not possible to differentiate between cause and effect in such a relation. It is not possible to state causalities. Moreover, the use of a quantitative traffic model for simulation requires the interpretation of the results by traffic engineers or by means of programs. A model which provides a qualitative description of the relations between the values and is restricted to qualitative values or states would be more useful. These kind of models are already used in traffic control engineering for the description of traffic flow.

Some concepts have been developed in various papers and are used by most approaches to qualitative modeling [Bürle & Lehmann 88]. After a short synopsis on general approaches to qualitative modeling, the transformation of the macroscopic traffic model into an adequate qualitative model will be explained.

3.2.1 General Concepts

The reduction of the structure description, which is used in a mathematical model, must generate a qualitative model whose structure meets certain demands. Modeling is based on the following principles:

a) The principle of ***locality*** [de Kleer & Bobrow 84]:
An element of the model can only affect its direct neighbors and vice versa. Spatial neighbors must be deducible from the structure from the start. The principle of locality simplifies deducing causal relations between processes and components.

b) The principle of ***hierarchical modeling*** [Bobrow 84]:
The description of a complex model is produced by a qualitative description of its components and their interrelations. On this basis it is possible to draw conclusions on the entire complex system.

c) The principle of ***context-free modeling*** [de Kleer & Bobrow 84]:
A model may not contain assumptions on the valid context. Assumptions explicitly made for all representatives of a class of objects can neutralize this principle: they reproduce the facts which are generally valid. Exceptional situations can be excluded by means of these assumptions. If the situations themselves are to be examined, the assumptions must be modified.

While the state of the modeled system is described by a number of exact values for all parameters in mathematical models and uncertain knowledge can only be presented by means of partial differential equations, qualitative modeling tries to display these values appropriately qualitative. For this reason the parameters are described by relations and reference values. The reference values are to orientate themselves by ***qualitative landmarks"***, which are particularly important for the corresponding field of application; so, for instance in the description of traffic flow "maximum speed" or "maximum traffic flow". A ***quantity space*** is defined onto the range of values of the observed physical variable [Iwasaki 89] which only contains a finite number of such landmarks. If such a value is reached, this is to correspond to a state modification of the system. The relations are restricted to simple comparisons (such as *"equal"*, *"greater"* and *"smaller"*) and a modification rate (such as *"ascending"*, *"descending"* and *"constant"*).

After an explanation of the general concepts for qualitative modeling, the various approaches to qualitative modeling are to be presented. They generally divide into component-oriented and process-oriented approaches [Puppe & Voß 87].

In the ***component-oriented structure description*** ([Kuipers 84], [de Kleer & Bobrow 84]) of a physical system, the ***component*** is the central unit, which invokes actions by itself and which is affected by actions. One or several components are responsible for each state modification. Components can be connected to each other and information can be exchanged via these links. These lines are the only means to distribute modifications. The local effect of components on each other via predefined channels strictly reduces the number of possible interactions. The system structure itself restricts the possible causal relations. One component has several qualitative states in which the component´s behavior is regarded as being constant. A *"specification"* describes the valid range of a state. A component´s behavior is reproduced by a sequence of such states. Dynamical relations are displayed by qualitative differential equations, deduced from the differential equations of a defined mathematical model. A sequence of state modifications can be regarded as simulation of the model of a device over the time. For the determination of possible state modifications rules have been developed based on the important characteristics of the qualitative calculus of [Bobrow 84].

Contrary to the component-oriented approach the active elements in the ***process-oriented structure description*** of [Forbus 84] ("Qualitative Process Theory" = QPT) are exclusively the ***processes***. Processes represent the dynamical operations within the structure. A process is directly responsible for every observed effect or modification in the system. The qualitative behavior of the components, which are true passive units in this context, is described in the QPT by processes. The activity of a process can modify the characteristics of components, generate or destroy components. The existence of certain components or the compliance with specific conditions on the characteristics of existing components, however, can be a prerequisite for the start or the termination of processes. Apart from a process´s *"prerequisites*" and *"parameter conditions"*, which comprise the conditions under which a process is active, it is also possible to indicate in the QPT whether these conditions can be influenced by the system itself or not. *"Relations*" and *"influences"*, which represent direct and indirect influences on the physical variable, are used for the description of the dynamic performance of a process. The QPT can also define a variable as dependent or independent. Moreover, causal interrelations can be displayed in a process-oriented approach, as the qualitative relations can either be directional or indirectional in contrast to the component-oriented approach.

It must be stated that the process-oriented approach seems to be better suited for modeling causal interrelations, dynamic events and partial knowledge than the component-oriented approach [Puppe & Voß 87]. In the component-oriented structure description the structure is predefined and irreversible. However, if mathematical equations and quantitative relations have already been analyzed and a statical system structure exists, the component-oriented approach is to be preferred.[1]

3.2.2 Qualitative Relationships

The development of a qualitative traffic model started with a macroscopic traffic model, which reproduces the traffic state in the time-distance dimension by means of the two continuous functions traffic volume $q(x, t)$ and traffic density $k(x, t)$ [Moreno et al. 90]. The state equation[2] describes the parabolic relation between the two traffic variables, which is assumed to be valid independent of time and place. As already mentioned above, the relation depicted in the fundamental diagram is not accurate anyway, as it depends on such factors as road characteristics and driver behavior. To deduce the qualitative relation, we additionally assume that this relation is accurate and that the vehicles in the traffic streams drive at the average speed, determined on the basis of the observed traffic density of the fundamental diagram. Furthermore, we assume that the vehicles do not move on according to the speed distribution[Leutzbach 88].

To obtain ***qualitative density values***, we look at the relation between density and speed. In contrast to the relation between density and traffic volume, this relation is strictly monotonous, i.e. each speed value is assigned exactly one symbolic density value. According to the qualitative modeling approach a partial order is defined in the value range of traffic density $(0 \le k \le k_{max})$ by means of significant reference values, e.g. k_{opt}. The partial order only contains a finite number of qualitative density values, which describe the assigned density intervals (Figure 3.2). If the boundary of an interval is reached, a state transition to the next density value occurs. The ordinal relation defined on the basis of this finite set of values is the greater-than relation between the density intervals, which have been assigned to the qualitative density values.

A qualitative density value, thus, is characterized by a traffic density interval and the intervals of the corresponding average speed and traffic volume. Make sure that each volume parameter is assigned two qualitative density values (Figure 3.3). For representation and processing of the qualitative density value a symbolic value such as *"D-1"* is used for the interval of the minimum density value 0 of *"STOP"* for the interval of the maximum density value k_{max}. These qualitative density values can in turn be assigned to traffic states which provide

1. For an examination of the approaches refer to [Struß 89] and a relevant overview is provided by [Iwasaki 89].

2. Cf. section 2.3

Qualitative description		Quantitative description		
verbal	**symbolic**	**intervals**		
state of traffic	qualitative density value	density $\left[\frac{cars}{h}\right]$	speed $\left[\frac{km}{h}\right]$	volume $\left[\frac{cars}{s}\right]$
free flow	D-1	[0.000, 0.015]	[70, 65]	[0.0, 0.3]
partial free flow	D-2	[0.015, 0.045]	[65, 54]	[0.3, 0.7]
	D-3	[0.045, 0.075]	[54, 44]	[0.7, 0.9]
max. flow	D-4	[0.075, 0.100]	[44, 35]	[0.9, 1.0]
dense traffic	D-5	[0.100, 0.125]	[35, 26]	[1.0, 0.9]
	D-6	[0.125, 0.155]	[26, 16]	[0.9, 0.7]
	D-7	[0.155, 0.185]	[16, 5]	[0.7, 0.3]
traffic jam	STOP	[0.185, 0.200]	[5, 0]	[0.3, 0.0]

Figure 3.2 Assignment of traffic density intervals, average speed and traffic volume to the qualitative density values [Moreno et al. 90]

a verbal description of the state produced by this value in a certain part of a section. The macroscopic model describes the traffic state in a certain section by a continuous density function. Here, the traffic state is represented by a list of dynamically changing zones (***density zones***), in which a given traffic state is described by a corresponding qualitative density value (Figure 3.4)

.

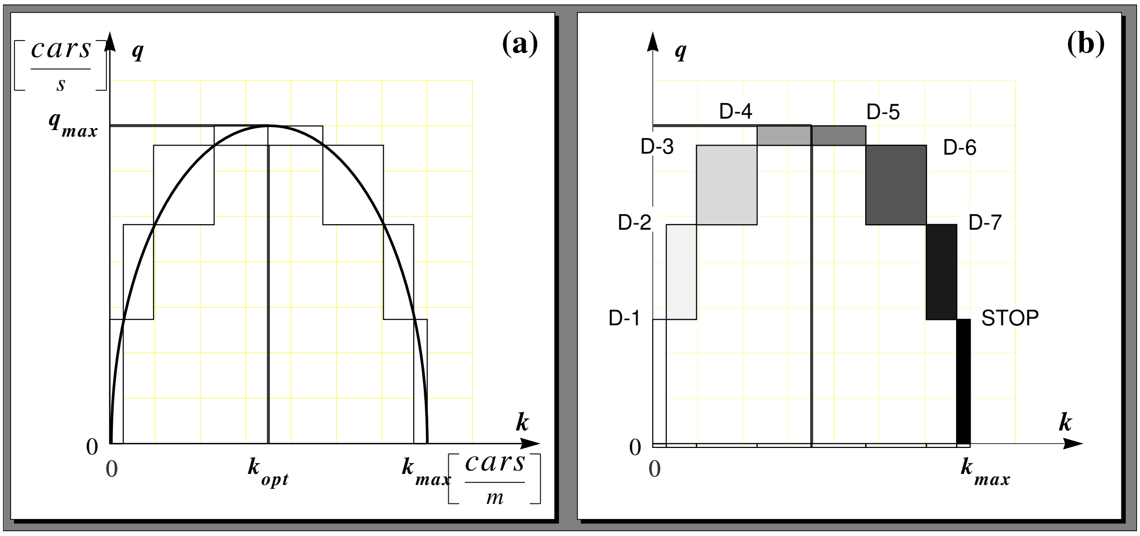


Figure 3.3 Transformation from the quantitative description (a) in the fundamental diagram to the qualitative description (b) by symbolic density values

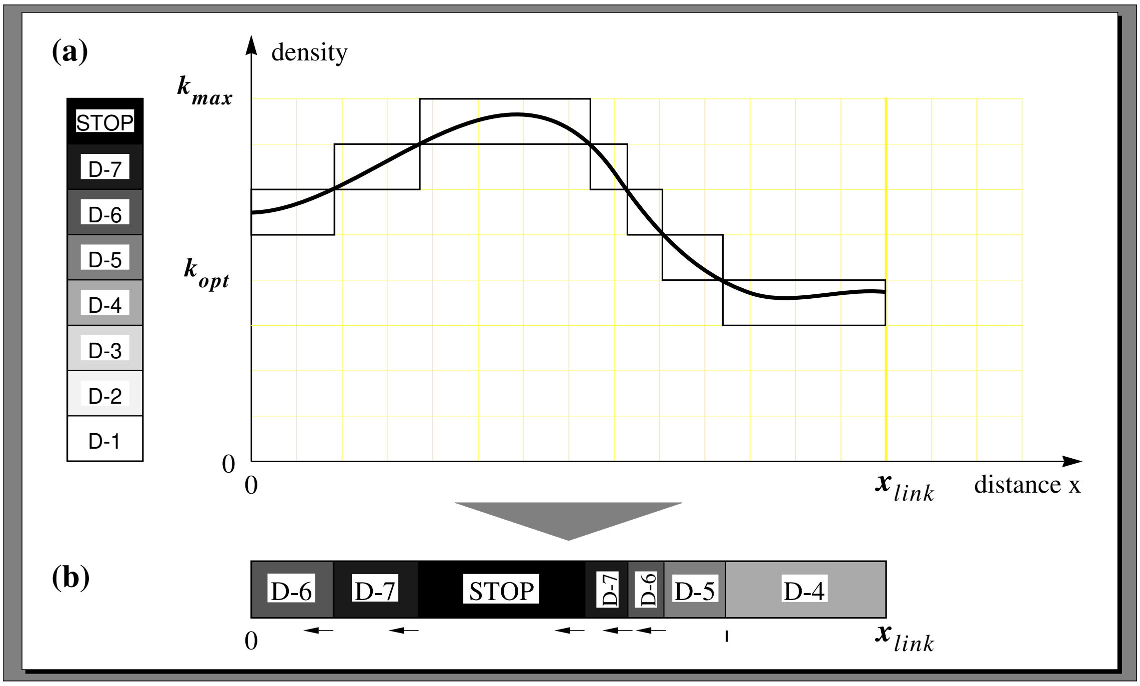


Figure 3.4 Description of the traffic state in a certain section at a fixed time:
(a) quantitative description via the continuous density function,
(b) qualitative description via a list of density zones

3.2.3 Qualitative Traffic Flow

If the description of the traffic states is transferred from the macroscopic model to the qualitative model, it becomes obvious that the density zones move in the sections if regarded over the time. Moreover, density zones disappear and new zones appear on the section.

To deduce the state transitions on a section in our qualitative model, the speed of the boundary between two density zones must be determined first (Figure 3.5 (a)). For this purpose the thoughts which lead to the determination of the shock wave speed in the description of the traffic flow via shock waves, are transferred to the situation at the boundary between two density zones. If the qualitative density value *"D-1"* (assuming that the average values of the assigned intervals are k_1, $\bar{v}_1$ and q_1) characterizes the situation in zone *DZ-1* in front of the zone boundary (in the direction of the traffic) and *"D-2"* (with k_2, $\bar{v}_2$ and q_2) the situation in *DZ-2* behind the boundary, $q_2 - uk_2$ is the rate of vehicles departing from the zone boundary to *DZ-2* and the arrival rate at the zone boundary equals $q_1 - uk_1$. According to the continuity equation the subsequent results for the ***border zone speed u***:

.

$$u = \frac{\Delta q}{\Delta k} = \frac{q_1 - q_2}{k_1 - k_2} = \frac{k_1\bar{v}_1 - k_2\bar{v}_2}{k_1 - k_2}$$

Similar to the shock wave approach the speed of the zone boundary can take on positive and negative values. If *u* is positive, the boundary moves along the direction of the traffic. If *u*

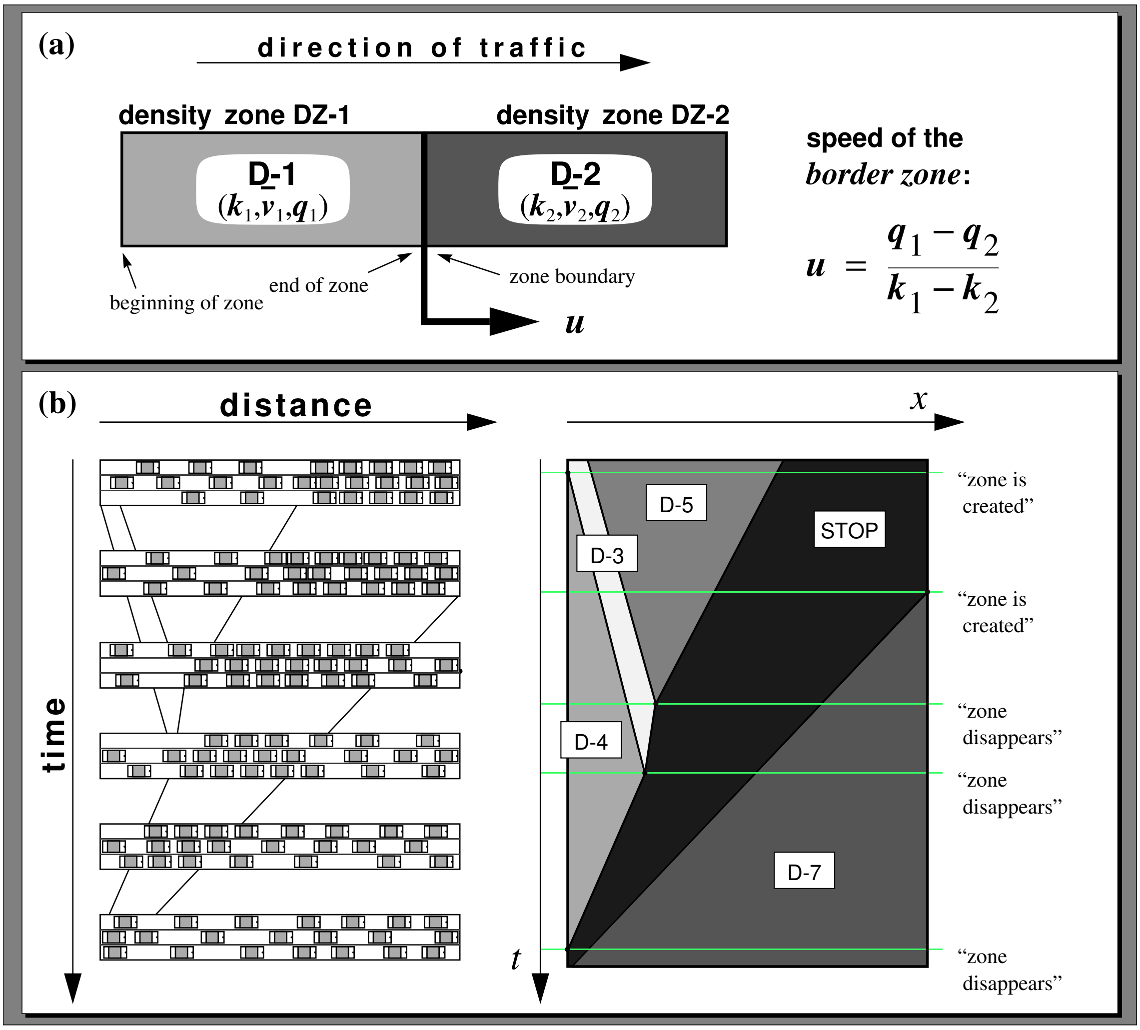


Figure 3.5 Movement of the density zones: (a) Determination of the speed at the boundary between two density zones, (b) Display of the development of the traffic density zones in a section in the time-distance diagram

is negative, the boundary moves in the opposite direction. If the boundary speed at the beginning of a density zone is smaller than the boundary speed at the end of the zone, i.e. if the two boundaries move towards each other, the length of the density zone will be reduced until the zone finally disappears (Figure 3.5 (b)).

To deduce further state transitions such as the creation of new density zones in a section or the behavior of a zone when reaching the intersection, a mere transfer of the macroscopic model will not suffice. The driver behavior and the events in the signal-controlled intersections must also be modeled. The necessary improvements of the qualitative model are explained in the design[1].

1.Siehe Abschnitte 4.4, 4.5 und 4.6

4 System Simulation

4.1 Event-Oriented Discrete Simulation Models

In contrast to the original approaches to qualitative simulation of the traffic flow the progress of time and the continuous modification of parameters, apart of landmark values and other state conditions, can invoke state modifications in the approach of [Moreno et al. 90]. Therefore, it seems reasonable to base the construction of the simulation model on ideas that have been developed for the description of event-oriented discrete models. Zeigler introduced the ***DEVS*** specification (*"**D**iscrete **EV**ent **S**ystem Specification"*, [Zeigler 76]) to show in a simple way how simulation languages can be specified for the simulation of event-oriented discrete models. The scheme is to facilitate the simulation of any computationable model of this kind [Zeigler 87]. The degree of detail, ranging from an absolutely detailed model to a very simple abstraction, can be defined by the model engineer. The parameters of the DEVS models can be of a deterministic or a stochastic nature or linked via intervals. Contrary to the qualitative simulation the abstraction process of the DEVS specification has been formalized several times and examined in various systems [Futo & Gergely 90].

The DEVS specification represents the common ground of all event-oriented discrete systems. It is not only a mere aid for the design of simulation models [Fishwick & Modjeski 91], but aims at a formal representation of event-oriented discrete systems, which allow similar mathematical manipulation options as does the description of a continuous system by means of a set of differential equations. The employed language is a mathematical formal one (set theory) and must be translated into a programming language when implemented in the computer. DEVS Scheme represents an implementation of the DEVS specification in a LISP-based and object-oriented programming environment (scheme), which allows the hierarchical and modular specification of event-oriented discrete models. With that a system-theoretic approach is pursued, which is not supported by the standard programming languages. The following presents the classes used for modeling and simulation in DEVS Scheme. They are implemented according to the set-theoretic formulation of DEVS specification. Subsequently, the basic principle of the simulation algorithm will be explained.

4.1.1 Classes in DEVS Specification

Figure 4.1 provides a first overview of the class hierarchy in the DEVS Scheme: all classes are subclasses of the universal class of *entities*, which provides the methods to manipulate the objects of the other classes. The inheritance mechanism of the object-oriented system guarantees that these methods must be defined only once. *Models* and *processors*, which are the main subclass of the entities, provide the basic constructs required for modeling and simulation. The models subdivide into the class of *atomic models* and of *coupled models*, which can be further specialized. The class of processors has three subclasses: *simulators*, *coordinators* and *root coordinators*, which are to fulfill all tasks in the context of simulation. Simulators, coordinators and root coordinators execute a simulation of the models according to the concepts of a hierarchical abstract simulator [Zeigler 84].

An ***atomic model*** *M* is an elementary component of the entire model and is represented by the tuple

$$M = (X, S, Y, \delta_{int}, \delta_{ext}, \lambda, \tau)$$

in DEVS, where X is the quantity of types of external input events, S the list of states of the model M, Y the quantity of output events produced by M, δ_{int} (and δ_{ext}) the internal (and external, respectively) state transition function, which indicates whether a state modification in model M was invoked by internal of external events, λ stands for the output function and τ for the function required to advance simulation time. As models consisting of only one component

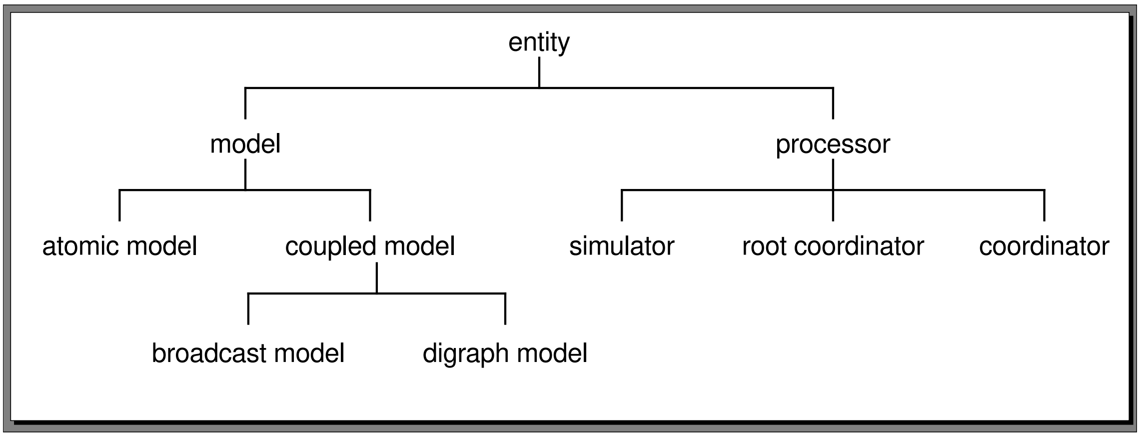


Figure 4.1 Hierarchy of classes in the DEVS scheme

hardly occur in reality, it is necessary to allow more complex models to be hierarchically and modularly constructed of the (simple) atomic models. The resulting ***coupled model*** consists of a set of atomic models. Its behavior is determined by the interactions of its components.

[Zeigler 84] describes the general functionality of the class of ***simulators*** in detail. The atomic models themselves only display the model engineer´s view on one model (or one component), looking at the simulation system from the outside. In principle, the states, the event times and the characteristics of the models are available to the system description by the atomic models during model development. The simulation model must be capable of managing the past simulation time correctly, to invoke an atomic model´s state transition functions at the right time and to co-ordinate the transmission of external events to other atomic models. All these tasks are to be solved by the concept of simulators. For this reason the code for the methods assigned to the simulators must be general enough to be applicable to the simulation of all atomic models. The ***coordinator*** has the same functionality as a simulator, except for the fact that he manages the simulation of a coupled model.

As a simulation system must show a predictable behavior to the user, the behavior of the simulator and coordinator in conflict situations must be clear. Which event will be treated first if two events occur simultaneously in a model? Which message on external events sent by adjacent models will be treated first when received at the same time? The DEVS scheme also treats these problems. DEVS reduces the conflict situations by the definition of a ***SELECT function***, which solves the conflicts on the level of the coupled models. This function guarantees that a component does not transmit messages to other objects while the external transition function is calculated and that the computation of the internal state transition function of a coupled model will only then be terminated when all components of the model finished processing of the received messages on external events.

The *root coordinator* encapsulates the whole model, i.e. it excludes input and output. It only manages the global watch and advances it to the next event time as soon as the current event has been processed. Therefore, the root coordinator has methods to start the simulation and determine the next event time.

4.1.2 Basic Principles of the DEVS Scheme

The basic problem of the simulation according to the DEVS scheme is that a set of simulation objects (atomic models, coupled models etc.) must be managed. The individual simulation objects can modify their state at discrete event times (***internal events***). They can also affect the state of other objects of this set by a state modification, i.e. they can cause state modifications in these objects (***external events***). To co-ordinate all these objects, a superordinate ***coordinator object*** is introduced to this type of simulation. The coordinator object controls the computation of the state transitions in the object and the intercommunication between the objects. For this purpose it manages two lists:

a) a list on all simulation objects which are to be coordinated (**`component-list`**),

b) a chronologically sorted list of event times, where the next event time in the simulation objects and the name of the corresponding object are entered (**`tN-list`**).

By means of these lists the coordinator controls the order in which the events in the objects are processed. Every first element in the **`tN-list`** can execute its event, i.e. proceed the transition to a new state. The new event times in the simulation objects, which are computed when processing the corresponding event, are again sorted into the **`tN-list`** with the name of the corresponding simulation object. The coordinator object also facilitates intercommunication between the simulation objects on external events. The messages sent in this context have the following components:

- name of the source (the sender of the message)
- time of transmission (as a local or global timestamp)
- name of the recipient and the values to be transmitted

To process the messages and to control the time of event handling, the coordinator does not require knowledge on the type of events in the simulation objects. It requires neither information on the form nor on the contents nor on the values transmitted in the message. This general approach allows the expansion of the simulation model by further simulation objects (atomic or coupled models) without modifications to the objects responsible for the co-ordination of the simulation.

To control intercommunication and event handling, each simulation object (atomic or coupled model) has the subsequent three functions apart from variables for storing the current simulation time and the description of the current state including the lists for recording the events that occurred in the simulation object:

1) ***External transition function***

This function transmits a message to an object on an event in an adjacent object, which affects the receiving object. This message is stored in the object (after handling). The return value results from the current simulation time. So, the receiving object applies for immediate processing of its next event.

2) ***Internal transition function***

This function orders an object to determine its state at the next event time based on its current state description (resulting from an external or internal event). The function produces this time as the return value. If it is not possible to determine a state transition, i.e. if an internal event will not occur in the future, it returns the particular value **`INFINITY`** (for the time).

3) ***Output function***:

This function instructs an object to process the next event available and to create messages to neighboring objects based on this event. The messages form the return value of the function.

Figure 4.2 displays how the functions are used in the simulation algorithm. After initialization of the coordinator object and the individual simulation objects, the determination of the next event time t_a according to the sorted **`tN-list`** of the coordinator (first entry) invokes a step during the simulation run.

As long as the current simulation time t_a has not reached the simulation horizon t_h, simulation objects will be determined whose entry on the next event time in the **`tN-list`** of the coordinator is not greater than t_a. If this applies to several objects, one object will be selected by means of the **`SELECT`** function[1]. Then, the output function for generating messages to adjacent objects is invoked for this object and the messages are transmitted to the neighbor objects by the coordinator. The coordinator, therefore, invokes the external transition functions for all adjacent objects which have been sent a message and updates the corresponding entries in the **`tN list`**. The internal transition function for the currently selected object is invoked at the end of a simulation step in order to determine the next internal event. Then, the coordinator updates the entry for the simulation object in the **`tN list`**.

In the design of the simulator prototype the DEVS scheme is reduced to a level of simulation objects, modeling the traffic flow in the network elements, and to a superordinate coordinator object as detailed above. The simulation algorithm represents a variant of the DEVS scheme reduced to the requirements of the prototype[2], whose implementation is object-oriented just like in the DEVS scheme.

1.Cf. section 3.4.1

2. For a more detailed description of the simulation algorithm including a precise explanation of the intercommunication between the objects and of the functions refer to [Zeigler 90].

4.2 Object-Oriented Simulation Models

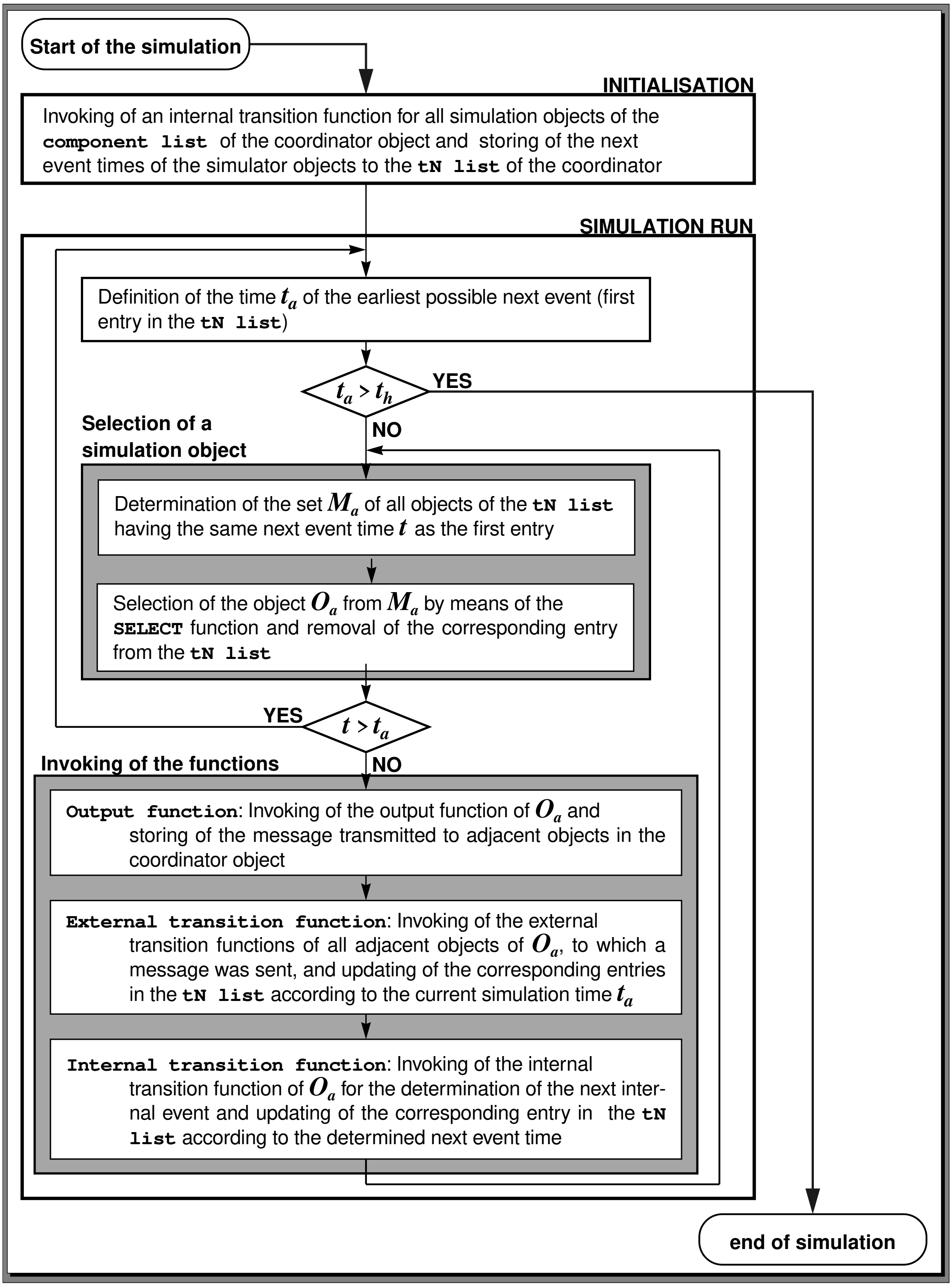


Figure 4.2 Algorithm for the execution of an event-oriented discrete simulation

The object-oriented paradigm characterizes a transfer of real structures to software structures. Real technical systems can be regarded as a set of physical objects (control unit, vehicle etc.), which intercommunicate by means of other objects (information object, message etc.). The structure of the real system is reflected by an object-oriented model structure, which reproduces the real objects and the information objects by software constructs.

The pivotal term of object-oriented programming is the ***object***. In this context an object is a self-contained entity (of an arbitrary complexity), which contains local data characterizing the state of the objects [Schönthaler & Németh 90]. On the basis of these data only locally defined operations can be executed. Execution of the operations is invoked via the object´s interface, which is the only way to communicate with the object. This concept, thus, is a consequent implementation of the principle of encapsulation (*Information Hiding* [Parnas 72]), which says that there is no need for the environment to know the internal structure of an object when using it.

The term *"object-oriented programming"* is used rather vaguely and should be replaced by the more precise term *"inheritance programming"* [Österle 88]. To be a true "object-oriented" programming language, a language must support the following concepts:

a) ***Objects***: "active" data structures, composed of a series of characteristics and behavior-defining methods; the characteristics are called *slots*, where a slot is characterized by its actual value (*slot value*) and a legal range of values.

b) ***Classes***: an abstract form of the objects (template); a class definition describes data elements (instance variables), common to all entities of a class together with the functions applicable to these data elements (***methods***).

c) ***Entities***: individual objects formed by instantiation from a class; any number of objects can be generated from a class; they inherit methods and the entity variables of their class; the assignment of the entity variables is individual.

d) ***Class hierarchy***: a class can be deduced from another class by explicit agreement; it inherits all entity variables and methods from the top class.

e) ***Dynamic link***: a method call is not resolved at the time of compilation (statical) according to the type of object reference, but at run time (dynamical) according to the class of the object reference.

The methods of an object are called by means of messages sent to the object. Another approach to method calling is the use of a ***generic function***. Figure 4.3 compares the two approaches. The central question is: where is the employed method determined? The sending of a message, which consists of an entity and the name of the method to be activated in this entity, results in an activation of the corresponding procedure. It is determined within the objects whether the method is locally known or not. Correspondingly a message will be generated and transmitted. The processing of the message produces the value resulting from the function call.

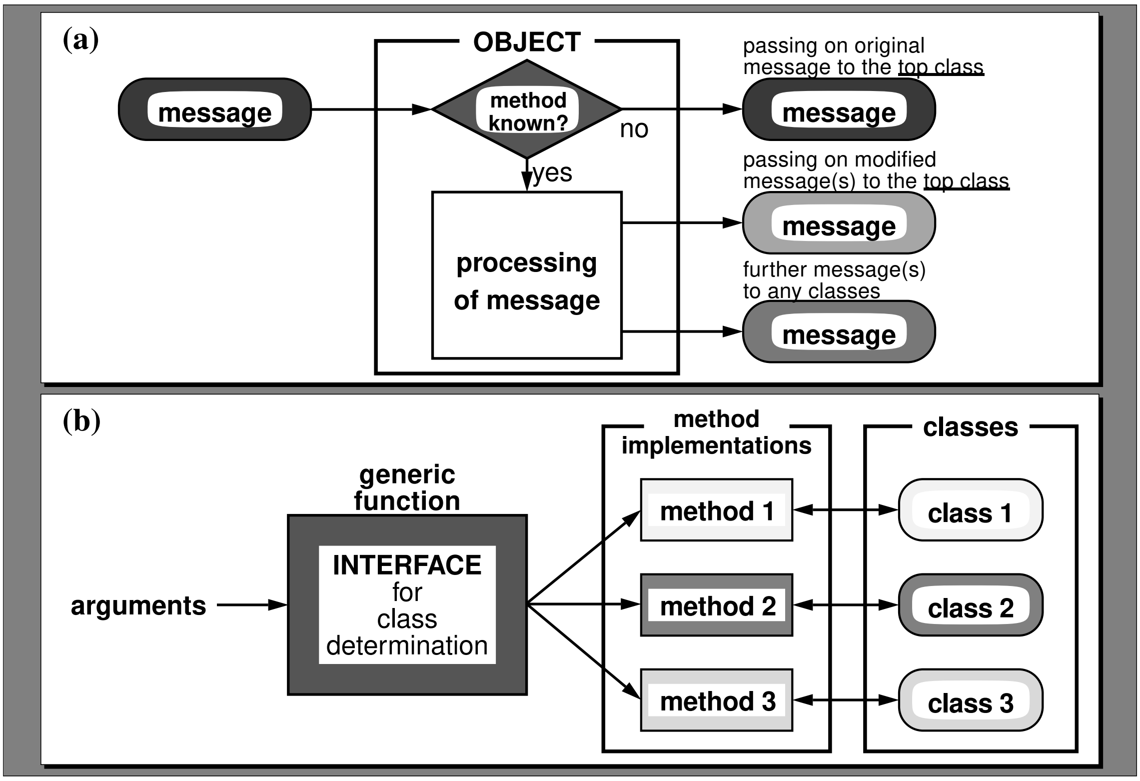


Figure 4.3 Addressing a method: (a) via transmission of messages or (b) via calling a generic function

Contrary to this, a generic function represents a procedure which is given an entity as first argument. A method will be selected from a set of implemented methods of the same name, which are assigned to a certain type of object class, on the basis of the class membership of the entity. This method will be activated.

The DEVS specification was first implemented with DEVS-Scheme with the concepts "*classes, objects and transmission of messages*". In a new implementation the DEVS specification is used as an expansion to CommonLISP [Steele 90] and the object system CLOS [Keene 89]. The DEVS-CLOS system is based on the object-oriented characteristics of CLOS. CLOS, however, operates with the concepts *"classes, objects and calling generic functions"*[1]. The implementation of the simplified variant of the specification, used for the design of the simulation model in the developed prototype, all functions, used for the intercommunication between the objects and control of event handling, must be defined as generic functions and the corresponding object-specific methods must be implemented for all types of simulation objects.

Object-oriented programming has the advantage of supporting a modular design reusing existing codes and expandability of the system. Apart from software-ergonomic advantages, it allows a natural representation of model components and their relations when creating a sim-

1. The inherent problems in an implementation of the DEVS specifications are explained in [Sevenic 90].

ulation model. Not only do object-oriented simulation models allow a detailed reproduction of the structure, but also of the function of technical systems (communication included).

5 The Design of the Sapporo System

5.1 Prototype Development

This chapter aims at designing the prototype of a simulation system, which already includes part of the relevant characteristics of the simulator. The simulator is to be used in a traffic guidance system as a tool for optimization of the traffic lights control. The prototype was designed according to a procedure called *experimental prototyping"* in software engineering (Figure 5.1). The principal aim is to complete an existing system specification, which forms the basis of a later implementation of the simulator. This kind of prototyping aims at an experimental verification of the suitability of the object specification, of the simulator model and of the solutions for individual system components. Starting from an idea of the analysis of the simulation system, a prototype is to be developed which allows to observe the relations between the system components, test the adequacy of component interfaces by means of simple examples and test the system in experiments [Schönthaler & Németh 90].

The decisive criterion for the assessment of the prototype is not the quality of the construction, but rather the prototype´s functionality, easy modification and the short-term devel-

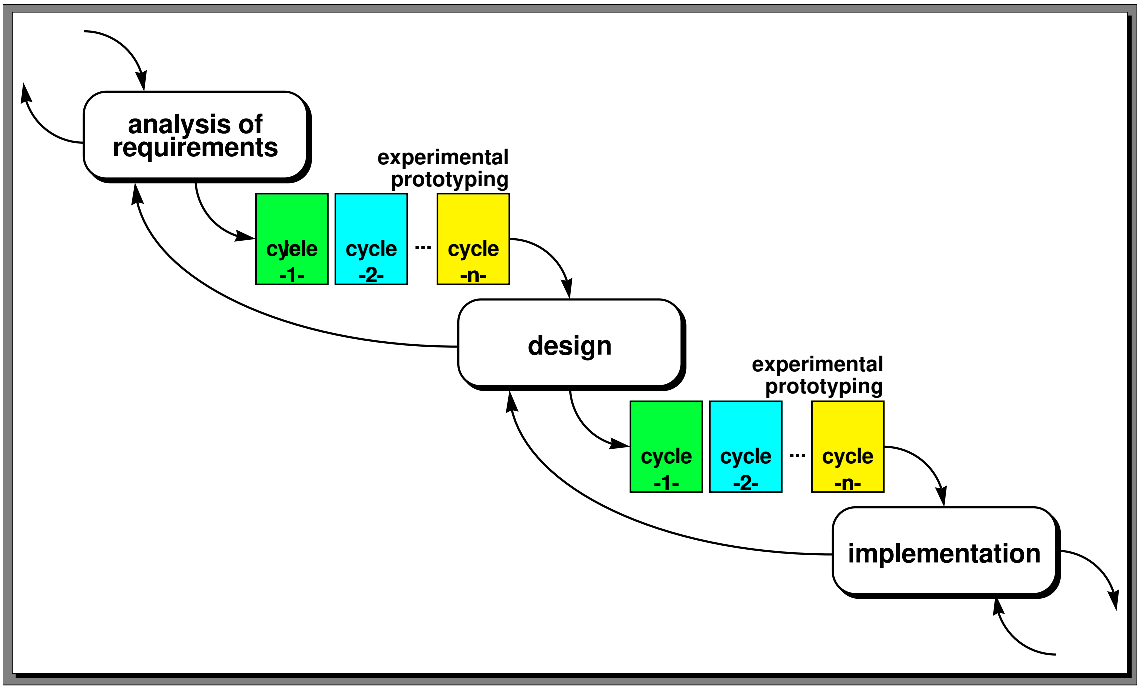


Figure 5.1 Integration of the *experimental prototyping* to the life-cycle model

opment. It allows the later user an early insight in functionality and performance of the future simulation system. An incomplete prototype will result from this work. It will already dispose of the basic functions of the designed simulation system and allows to examine the use of the model, the simulator and the user interface. It will rather be a one-way prototype, an operational model of the system. The object-oriented design of the simulator and the object-oriented implementation in CLOS allow to take over the basic components of the prototype and their expansion when implemented into the target system.

Within the framework of the object-oriented design of the prototype, object classes are specified first, which are used for the implementation of the qualitative traffic model and allow a discrete event-oriented simulation. After this, the simulation algorithm produced by a simplifying application of the DEVS specification and the event treatment in the simulation objects will be explained. Finally, the improvement of the qualitative traffic model, which is to describe the traffic flow on street sections, and the application of the qualitative approach to the simulation of the events on intersections and marginal points of a traffic network will be shown.

5.2 Object Classes

The following will present the object classes which have been specified[1] and implemented for the description of the traffic network´s topology. Then, the simulation objects assigned to the network objects will be described. In the simulation objects the traffic flow is reproduced by means of a qualitative, macroscopic traffic model. The objects used for the computation of state transitions in this model are presented in the section titled “density zone calculus”.

5.2.1 Network Objects

In the object-oriented analysis of a traffic guidance system the basic elements can be divided into ***statical*** and ***dynamical elements*** according to their temporary nature. The ***functional elements***, used for the construction of the user interface, form a third group. They allow the user to generate and manipulate all objects detailed below.

Statical elements are on the one hand all objects which describe the network structure and the marginal conditions of traffic. This includes the position of sections in a street network and their links as well as the individual characteristics of the sections. Statical elements are on the other hand all objects, which are of importance for controlling the traffic streams - from simple traffic lights to complex control components of the traffic guidance system. Entities of this object class are only once generated during the modeling process and are not modified during the simulation. Contrary to this, the dynamical elements can be generated and removed during simulation. They describe the current traffic state. The parameter values influenced by temporary variations such as the traffic volumes at the incoming lanes of the road network and the trip matrices for the determination of the sections of O/D traffic (*O/D matrices*).

Classes that have already been implemented for the topological description of the road network form the basis of the design of object classes, which will be used for the simulation of the traffic flow:

a) **`Network`**: It consists of a set of **`roads`**, combined for organizational reasons or the description of a part of a map.

b) **`Road`**: It consists of a ordered set of **`sections`**, linking the start and the end of a road.

1. An overview of all elements required for modeling the intra-urban traffic, the individual behavior of the road users, the traffic lights control and a traffic guidance or traffic information system is provided by the specification of top classes in the *SAPPORO* traffic guidance system [Wild & Berning 91].

c) **`Section`**: It is a part of the road, linking two adjacent sites (e.g. junctions), having the most important characteristics related to traffic engineering (number of lanes, width, restrictions).

d) **`Lane`**: It describes a lane as part of a section (one-way road or multilane road) as well as its links to other lanes (no-turnings, mandatory turnings and turning behavior).

e) **`Node`**: Node where two sections meet or a marginal point of the network; start or end of sections with the appropriate coordinates in a predefined coordinate system.

f) **`Source/sink`**: point-like objects as generators/consumers of individual vehicles or traffic streams.

From the simulator's point of view the sources and sinks also count among the statical elements, although they describe the dynamical traffic flow, since the position of the sources (sinks) and the values and value sequences, respectively, generated by them are regarded as constant or predefined during simulation. As only fixed signal plans are used for traffic control in intersections and constant O/D matrices are used for the description of the turning behavior when designing a prototype, further objects specified for traffic management are not of importance.

5.2.2 Simulation Objects

The simulation objects and the coordinator object, which manages the simulation objects during simulation, are generated on the basis of the description of the road topology by network objects. The traffic flow in a network object is modeled in a simulation object (Figure 5.2):

- in each incoming lane of the traffic flow or endpoint of the network (**`node`**) with an assigned **`source`** by a ***marginal object*** (**`endpoint`**),
- in each **`node`** by a ***crossing object*** (**`crossing`**),
- on each **`lane`** by a ***link object*** (**`link`**).

The data and methods common to all simulation objects are combined in the class **`dynamic-standard-mixin`**. They allow to manage the simulation objects during the simulation by a coordinator object according to the principles of the DEVS specification. For this purpose each simulation object requires the slots (Figure 5.3):

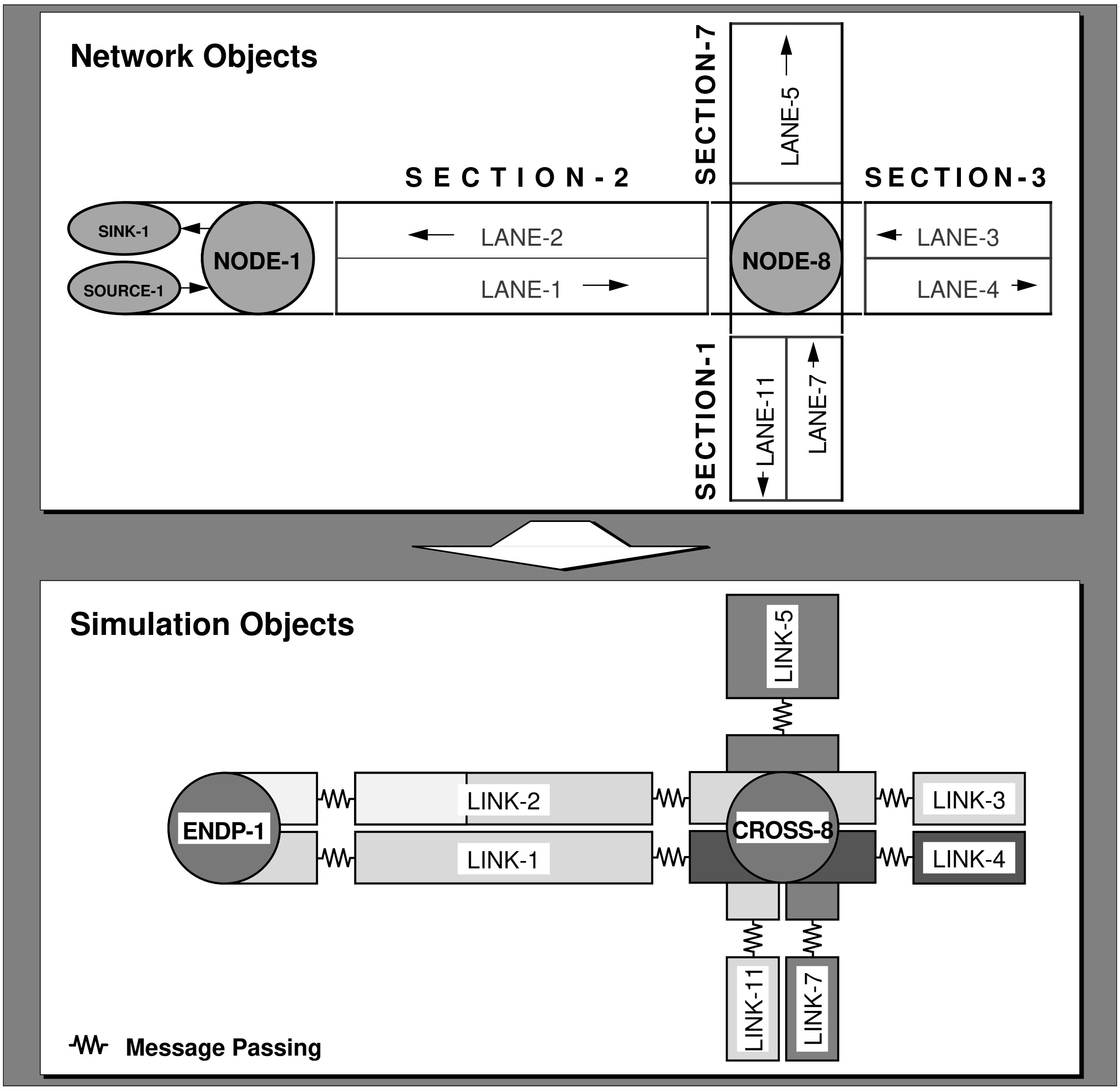


Figure 5.2 Network objects and the corresponding simulation objects

- **`time-of-next-event`** for storing the time of the next event to the simulation object,
- **`next-event`** for storing the next possible (pre-calculated but not yet effected) event,
- **`event-list`** as a list of all events which already occurred in the simulation object (in chronological order).

A simulation object is initialized with a state description at the beginning of a simulation by means of the internal function

`(set-object-state sim-object init-state).`

During simulation the current state description of the object is determined by

`(get-state-at-timepoint sim-object time).`

For intercommunication between the simulation objects and control of the event handling in the objects, the coordinator invokes the following methods[1], which are defined as generic functions and implemented as interface methods for each class of simulation objects:

1. Cf. section 3.4.2

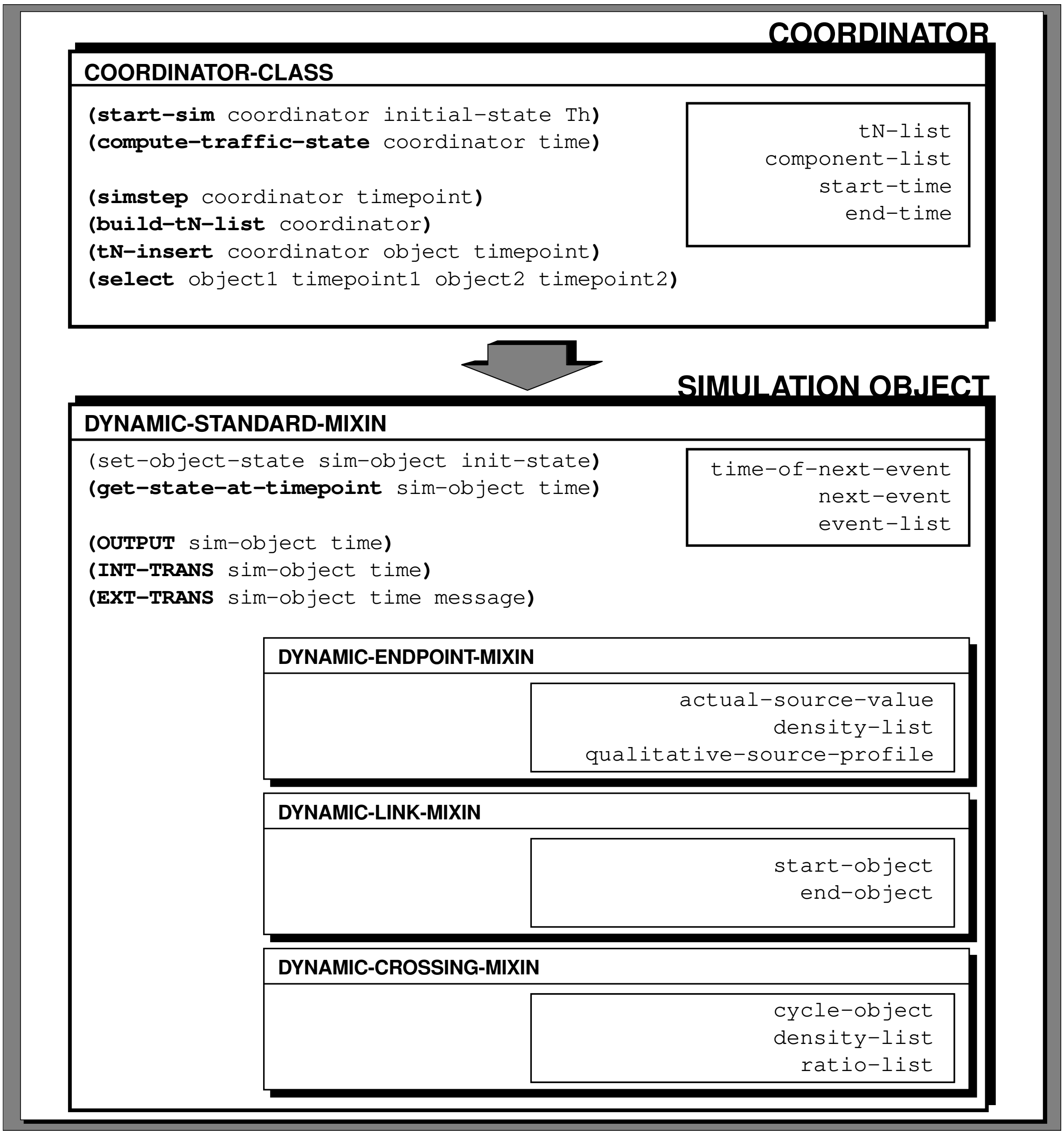


Figure 5.3 Overview on the methods and slots of the coordinator object and the simulation objects managed by it

- **`(OUTPUT sim-object time)`** as output function,
- **`(INT-TRANS sim-object time)`** as internal transition function,
- **`(EXT-TRANS sim-object time message)`** as external transition function.

The coordinator object requires the following slots for simulation:

- **`tN-list`** for storing the next (possible) events with the corresponding simulation objects

- **`component-list`** for storing all simulation objects
- **`start-time`** and **`end-time`** for starting and ending the simulation
- and the internal methods:
- **`(simstep coordinator timepoint)`** for performing a simulation step at a given event time,
- **`(build-tN-list coordinator)`** for generating the **`tN-list`** by invoking the internal transition function of all simulation objects,
- **`(tN-insert coordinator object timepoint)`** for adding a new entry to the **`tN-list`**; the entry consists of a simulation object and the corresponding next event time,
- **`(select object1 timepoint1 object2 timepoint2)`** to chose from two simulation objects, where the object with the lower event time is preferred or in case of the same event time, the object is chosen according to a predefined preference order.

The interface methods of the coordinator object allow to start the simulation and define the traffic state at a certain simulation time:

- **`(start-sim coordinator initial-state Th)`** starts a simulation of a given initial state, used to initialize the simulation objects and executes the simulation step by step (via **`simstep`**) until the simulation horizon is reached,
- **`(compute-traffic-state coordinator time)`** extracts the traffic state from all simulation objects managed by the coordinator.

To describe the current traffic state and execute the qualitative simulation in the individual simulation objects, the individual classes of simulation objects have further slots:

- a list of qualitative density values (**`density-list`**), which apply to the end and the start, respectively, of the lane adjacent to the endpoint, and the currently generated density value (**`actual-source-value`**), which is read from the qualitative description of the profile belonging to the source (**`qualitative-source-pro-file`**), describe the traffic state in an marginal object,
- an crossing object requires the signal plan (**`cycle-object`**), which controls the traffic flow on the intersection by means of a traffic lights system, a list containing the turning

ratio (**`ratio-list`**) and a list of the qualitative density values (**`density-list`**), which apply to the end and the start of the lanes adjacent to the intersection, to describe the traffic state,

- the state of a lane simulation object is described by the current distribution of density zones to the entire length of the lane, which does not need to be stored separately, as it is already included in the **`event-list`** of the object; only the simulation objects at the start and endpoint of the lane (**`start-object/end-object`**) must be entered to determine the direction of traffic and to send the messages.

Apart from the simulation objects and the coordinator object, the objects needed for simulation also include situation objects (**`situation-class`**), which are used to store the simulated traffic state to all simulation objects. A new simulation can be started on the basis of a stored situation. For this purpose the following values are stored to the slots of a situation object:

- the time of the traffic situation (**`situation-time`**),
- the qualitative description of the traffic state of the simulation objects managed by the coordinator object (**`situation-list`**).

5.2.3 Density Calculus

The term ***"density calculus"*** combines all objects which facilitate the qualitative description of the traffic state in the elements of a road network according to the macroscopic qualitative model and can be used to simulate the traffic flow by the movement of density zones. The term "calculus" can be traced back to the implementation of the simulation by means of the qualitative model in a PROLOG-based system [Moreno et al. 90]. While the PROLOG-based system deduces new states in the simulation objects via rules, the simulator prototype designed in this paper determines state modifications via LISP functions.

The "density calculus" is based on the quantitative relation between the macroscopic traffic parameters average speed $\bar{v}_m$, traffic volume q and density k, as depicted in the fundamental diagram. Either the $\bar{v}_m$-k relation or the q-k relation (i.e. the traditional fundamental diagram) in form of a list of value tupels suffice for a description of this relation. The equation $q = \bar{v}_m \cdot k$ can extend the tuple to a triple:

- $k \xrightarrow{\bar{v}_m\text{-}k \text{ relation}} \bar{v}_m \longrightarrow q = \bar{v}_m \cdot k$,
- $k \xrightarrow[\text{diagram}]{\text{fundamental}} q \longrightarrow \bar{v}_m = q/k$, $\bar{v}_m = 0$ for $q = 0$.

If the relation between the three parameters complies with the marginal conditions formulated by [Leutzbach 88], the fundamental diagram will be used to generate the density calculus. When generating an object of the class of fundamental diagrams (**fundamental-diagram-class**), only conditions 1 - 4 will be checked. The following slots describe the fundamental diagram:

- **value-list** for storing the list of $(k,q,\bar{v}_m)$ triples,

- **value-descriptors** to describe the type of triples (**:density-flow-calculatedspeed** or **:density-calculatedflow-speed**),

- **max-flow** for storing the max. flow value q_{max} $(0 \le q \le q_{max})$,
  max-density for the max. density k_{max} $(0 \le k \le k_{max})$ and
  max-speed for the maximum (average) speed $\bar{v}_f$ $(0 \le \bar{v}_m \le \bar{v}_f)$.

As we assume in this model that the relation, which is described by the relation in the fundamental diagram, is valid independent of time and place, the traffic state at a given time and in any place is clearly defined by a $(k,q,\bar{v}_m)$ triple. The value range of the traffic density and thus the value range of the traffic parameters volume and average speed are divided into intervals, which are assigned a qualitative density value when the quantitative relation is converted into a qualitative relation. Instead of a value triple, a symbolic density value describes now the traffic state at any place of the road network. An object of the ***density value class*** (**density-value-class**) has the slots or subclasses:

- **key** as a symbol for the density value (e.g. **D-1**, **D-2**, … **STOP**),

- **interval-range-mixin** for storing the interval boundaries of the traffic density, volume and average speed intervals assigned to the qualitative density value,

- **mean-density**, **mean-flow** and **mean-speed** for storing the average values of the appropriate intervals.

The qualitative density values can be assigned to objects of the ***traffic state class*** (**traffic-state-class**) to describe the traffic state also verbally by means of a technical term related to traffic engineering. These objects dispose of the following slots and top classes:

- **density-values** with a list of the assigned density values,

speed of the zone border $\left[\frac{m}{s}\right]$		density value of zone **after** border							
		D-1	D-2	D-3	D-4	D-5	D-6	D-7	STOP
dens. value **in front of** border	D-1	-	15.556	12.381	10.000	7.619	4.906	2.154	0
	D-2	15.556	-	10.000	7.826	5.455	2.727	0	-2.154
	D-3	12.381	10.000	-	5.455	2.857	0	-2.727	-4.906
	D-4	10.000	7.826	5.455	-	0	-2.857	-5.455	-7.619
	D-5	7.619	5.455	2.857	0	-	-5.455	-7.826	-10.000
	D-6	4.906	2.727	0	-2.857	-5.455	-	-10.000	-12.381
	D-7	2.154	0	-2.727	-5.455	-7.826	-10.000	-	-15.556
	STOP	0	-2.154	-5.455	-7.619	-10.000	-12.381	-15.556	-

Figure 5.4 Matrix displaying the speed of the zone border between two density zone

- **`identifier`** with an identification of the traffic state,

- **`interval-range-mixin`** to store the interval boundaries of the density intervals of density, volume and average speed assigned to the traffic state.

To store references to the objects of the classes **`fundamental-diagram-class`**, **`density-value-class`** and **`traffic-state-class`** as well as pre-calculated results for the determination of the movements of the density zones on the sections, the class **`density-calculus-class`** is defined:

- **`fundamental-diagram`** reference to the fundamental diagram,

- **`density-value-keys`** list of the symbols of the density values,

- **`sorted-density-value-list`** list of the qualitative density values (according to the assigned average density in ascending order),

- **`sorted-traffic-state-list`** list of the traffic states,

- **`border-speed-table`** to store the matrix displaying the speeds of the boundaries (Figure 5.4) between two density zones, where the speed is computed with the average values of the assigned intervals of density and volume according to the shock wave approach,

- **`new-density-zone-table`** to store the matrix describing the possibility to create density zones of a certain density between two zones on a section[1].

1. Section 4.5 explains this improvement of the qualitative model.

The speed of the border between two zones of a different density, which is required to describe the movement of the density zones, can be determined by

```
(get-border-speed density-value-1 density-value-2)
```

from the matrix containing the pre-calculated values (from **border-speed-table**)

The call

```
(new-intermedium-zone-p
    density-value-new density-value-1 density-value-2)
```

determines whether a new density zone of the density value **density-value-new** can be generated between two existing zones by means of the pre-calculated values from the **new-density-zone-table**.

5.3 Prototype Model

An event-oriented discrete simulation is performed by means of the coordinator object and the simulation objects on the basis of the DEVS specification. According to the basic idea of this approach[1], the superordinate coordinator object takes care of the coordination of the simulation in the simulation objects, in which the traffic states are displayed and the state changes for the description of the traffic flow are determined in a qualitative model. Furthermore, the coordinator object allows to transmit messages on events (state modifications), which affect the state of adjacent simulation objects, between objects.

For intercommunication and invoking the computation of the next state in a simulation object, the coordinator calls generic functions, which are defined depending on the class of the simulation object. After a description of the simulation algorithm using these functions, the following will explain the event treatment of endpoint, land and intersection simulation objects within these functions.

5.3.1 Object-Based and Event-Oriented Simulation

1. Cf. section 3.4.2

Techniques

After having loaded the network objects and after the topological description of the traffic network and the generation of the objects of the "density calculus", all objects required for simulation are generated and assigned references by means of the

```
(simulation-install network)
```

call. So, a corresponding link object is generated for each lane, a endpoint object for each marginal point object of the network and a crossing object for each intersection. All simulation objects are entered into the **`component-list`** of the coordinator which is also generated. Moreover, a qualitative representation of the sequence, describing the traffic flow into the road network, is generated for each marginal point object by the qualitative density values according to the division of the ranges of values of the macroscopic traffic parameters. For each crossing object the corresponding signal plan object is assigned a reference (interface to the control component of the traffic guidance system).

Simulation by the coordinator object is started by the method

```
(start-sim coordinator initial-state Th).
```

Starting with the traffic state recorded in **`initial-state`** for a certain time in all simulation objects, the simulation is performed until the simulation horizon **`Th`** is reached. If no start situation is given, the lane simulation objects are initialized with the lowest qualitative density value, i.e. an empty traffic network, where free-traffic is possible on all lanes, is described.

During simulation the time jumps from event time to event time, at which one event occurs in at least one of the simulation objects. If the coordinator object gets the instruction

```
(sim-step coordinator timepoint),
```

it takes the element of the highest priority (**`select`** function) from the **`tN-list`**, if the event time assigned to the entry (event time and simulation object) corresponds to the current simulation time. Corresponding to the basic operation of the simulation algorithm (Figure 4.2) the state transition functions of the selected simulation object and the adjacent objects will be called.

The state transition functions are implemented as generic functions within the simulator, which are called by the coordinator object to control the simulation. The methods, which are then called dependent on the type of object class of the simulation object, have independent of the object type the same functionality from the coordinator object´s point of view:

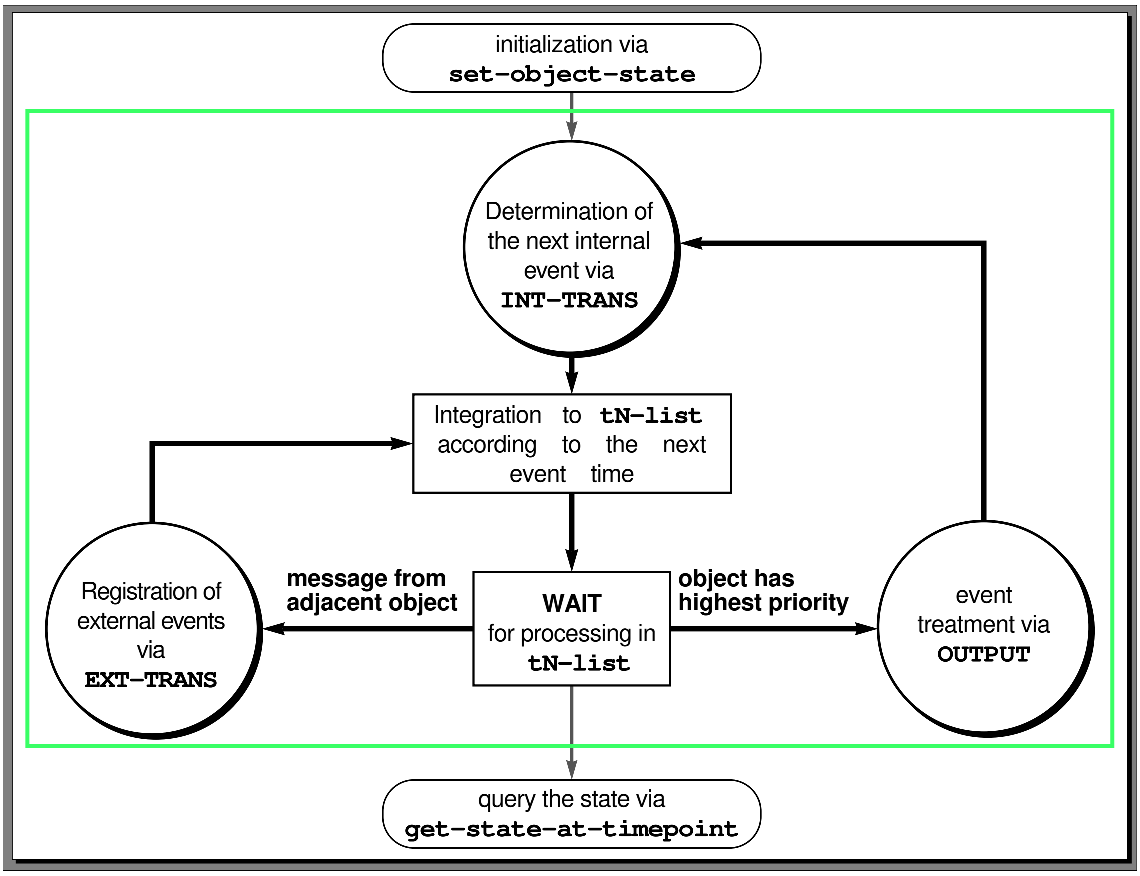


Figure 5.5 Operation of the simulation from the simulation object´s point of view (by means of the interface methods of the simulation object)

a) The addressed simulation object is instructed to process its next event and generate a message to adjacent objects affected by the event if needed by **`(OUTPUT sim-object time)`**. As only modifications of the qualitative density values are signaled to other simulation objects in the prototype, only new density values must be communicated between the objects. Thus, the message format [Zeigler 90] known from the DEVS specification can be reduced as follows:

```
<message>    ::=    (<receiver-object> (<sender-object>
                     <density-value>))
```

- The coordinator object needs the name of the recipient to pass on the messages. Apart from the new density value the recipient needs the name of the sender of the density value for correct internal processing.

b) Messages to adjacent objects of the currently selected simulation object are transmitted via **`(EXT-TRANS sim-object time message)`**. The recipient objects are instructed to process the new external events. The return value of each function call is the current simulation time, as the next events in the addressed simulation objects may have been affected by the external event (by a message on a new density value). All entries concerning the simulation objects are updated in the **tN-list** of the coordinator according to the new event time.

c) The addressed simulation object is instructed by **(INT-TRANS sim-object time)** to compute the next event in it on the basis of the current state description at a given time. The return value is the time of the next event. If it is not possible to determine a state modification, the return value is **INFINITY** (for the time).

Figure 5.5 depicts the simulation from the simulation object´s point of view as it is performed by means of the object-specific implemented methods.

5.3.2 Event Handling

Event handling in endpoint, link and crossing objects is transparent to the coordinator object within the generic functions **OUTPUT**, **INT-TRANS** and **EXT-TRANS**. The following specifies the functions for all three classes of simulation objects with regard to event handling.

Each simulation object contains the next event in **next-event** (new state description or new density value) and the time of the event in **time-of-next-event**. Events that have already been effected are entered into the simulation object´s list of events (**event-list**). The basic format of such an event reads as follows:

<event>	::=	**(<*event-time*><state><*event-type*>)**
<state>	::=	**<marginal-point-state> | <lane-state>**
		|<crossing-state>
<event-type>	::=	**<marginal-point-marginal-point>**
		|<lane-event-type>
		|<crossing-event-type>

In addition to the event time and a object-specific state description the type of the event is also stored. These data allow to determine the state of a simulation object at any time between two event times by means of the LISP functions generated for event handling.

5.3.2.1 Marginal Object

To describe the current traffic state in a marginal object, the qualitative density values of the density zones at the end and at the beginning, respectively, of the adjacent lanes are stored as a list:

<marginal-point-state> ::= ({(*<lane><density-value>*)})

The subsequent event types can occur as cause of events in this simulation object:

- new density value of an adjacent link object,
- new density value on the assigned affluent profile which is to be introduced,
- initialization by a density value,

<marginal-point-marginal-point>::=(*event: <lane><density-value>*)
(*event: PROFILE-EVENT <density-value>*)
|(*event: INITIAL <density-value>*)

During event handling the simulation object first determines the new assignment at the connections to adjacent link objects and notes this state in its event list. Only in case of a change of the density value which is to be introduced a message is transmitted to the adjacent leaving lanes. Event handling divides into three methods used by the coordinator:

a) **OUTPUT**: In case of a new density value which is to be introduced and which is capable of spreading into the road network, a message is transmitted to the leaving link object from the message on the new density value stored in **next-event**; a change of density values at the start or end of adjacent lanes in only noted in the event list.

b) **EXT-TRANS**: The marginal object receives a message on a modified density value at the end or start of an adjacent link object. The current simulation time is returned as the time of the next event in the marginal simulation object. The simulation object applies for immediate event handling at the coordinator in order to modify its own state description according to the modified density value. The density value modifications signaled by adjacent link objects are only recorded/noted and do not affect the introduced density values (and their alternation time).

c) **INT-TRANS**: The alternation time (current density value to next density value) is determined from the qualitative description of the affluent profile. The new value which is to be introduced is stored in **next-event** and the alternation time of the value is returned as the next event time to the coordinator object.

5.3.2.2 Link Object

The description of the current traffic state in a link object requires the distribution of the density zones on the lane.

```
<lane-state>            ::=    ( {<density-zone>} )
<density-zone>          ::=    (<density-value> <start> <end>)
```

In this simulation objects events can be caused by:

- a new density value from an adjacent marginal or crossing object,
- the disappearance or generation of a density zone within a lane,
- an initialization via/including density zone distribution

```
<lane-event-type>       ::=    (event: <marginal-point> <density-value>)
                               |(event: <crossing> <density-value>)
                               |(event: INTERNAL)
                               |(event: INITIAL)
```

The new density zone distribution valid until the next event time on the link object is calculated in the event handling by means of the “density calculus” and stored to the event list. If a density zone disappears at the start of end of a lane, a message on the density value of the zone that has arrived at the border is transmitted to the adjacent simulation object. The event handling is distributed to the methods called by the coordinator.

a) **`OUTPUT`**: The link object executes its message stored in **`next-event`** as an event, i.e. the anew calculated density zone distribution is entered into the event list. Events that occur within a link object such as the generation of disappearance of a density zone (not at the border of the section) do not generate messages to adjacent objects. Events, where density zones disappear or appear at the start or end of the lane, transmit messages signaling the new density values at the borders of the adjacent objects.

b) **`EXT-TRANS`**: Thus, the link object is instructed to take over the density value of the adjacent simulation object. The message on the new density value is stored to **`next-event`** and **`time-of-next-event`** is set to the current simulation time and is returned. The simulation object, thus, applies for immediate event handling, which determines/states whether the transmitted density value will cause a density zone at the start or end of the lane or disappears immediately.

c) **`INT-TRANS`**: Based on the last density zone distribution the next internal event (generation or disappearance of a density zone) is calculated according to the shock wave approach[1] that has been applied to the movement of density zones. The anew calculated distribution of density zones is stored to **`next-event`** and the time of the calculated event is returned.

1. Cf. section 2.4

5.3.2.3 Crossing Object

The qualitative density values of the density zones at the start of the leaving and the end of the entering lanes, respectively, are of interest for the description of the current traffic state in a crossing object.

<crossing-state> **::=** **({(<*lane*> <*density-value*>)})**.

The events in this simulation object can be caused by:

- phase switching of the corresponding signal plan or modification of the turning behavior on one of the entering lanes,
- a modification of the density value of the zone at the border/margin of an adjacent lane,
- initialization including density value assignment

<crossing-event-type> **::=** **(*event: CYCLE-EVENT*)**
|(*event:* <*lane*> <*density-event*>)
|(*event: INITIAL*)

In the event handling, the simulation object first determines the current turning ratio for the currently valid phase of the assigned signal plan. Then, it calculates a new assignment of? density values by means of an extensive algorithm[1] which is stored to the event list as the new state. Finally, it generates a list of messages to adjacent link objects, at whose borders new qualitative density values will generate new density zones. Event handling is distributed to three methods used by the coordinator:

a) **OUTPUT**: The crossing object executes its internal event, i.e. the new assignment of density values is determined on the basis of the message in **next-event**, the current signal plan phase and the current turning ratio. A message is generated for all adjacent link objects where a new density value is located at the border. The list of messages will be returned.

b) **EXT-TRANS**: The crossing object receives a message on a new density value. The message is stored to **next-event**. The current simulation time is returned, so that the simulation object can apply for parallel event handling at the coordinator object.

c) **INT-TRANS**: The crossing object determines the next time of a signal plan modification (switch-over time), stores it as **CYCLE-EVENT** in **next-event** and returns the switch-over time.

1. Section 4.6 contains a description of the algorithm.

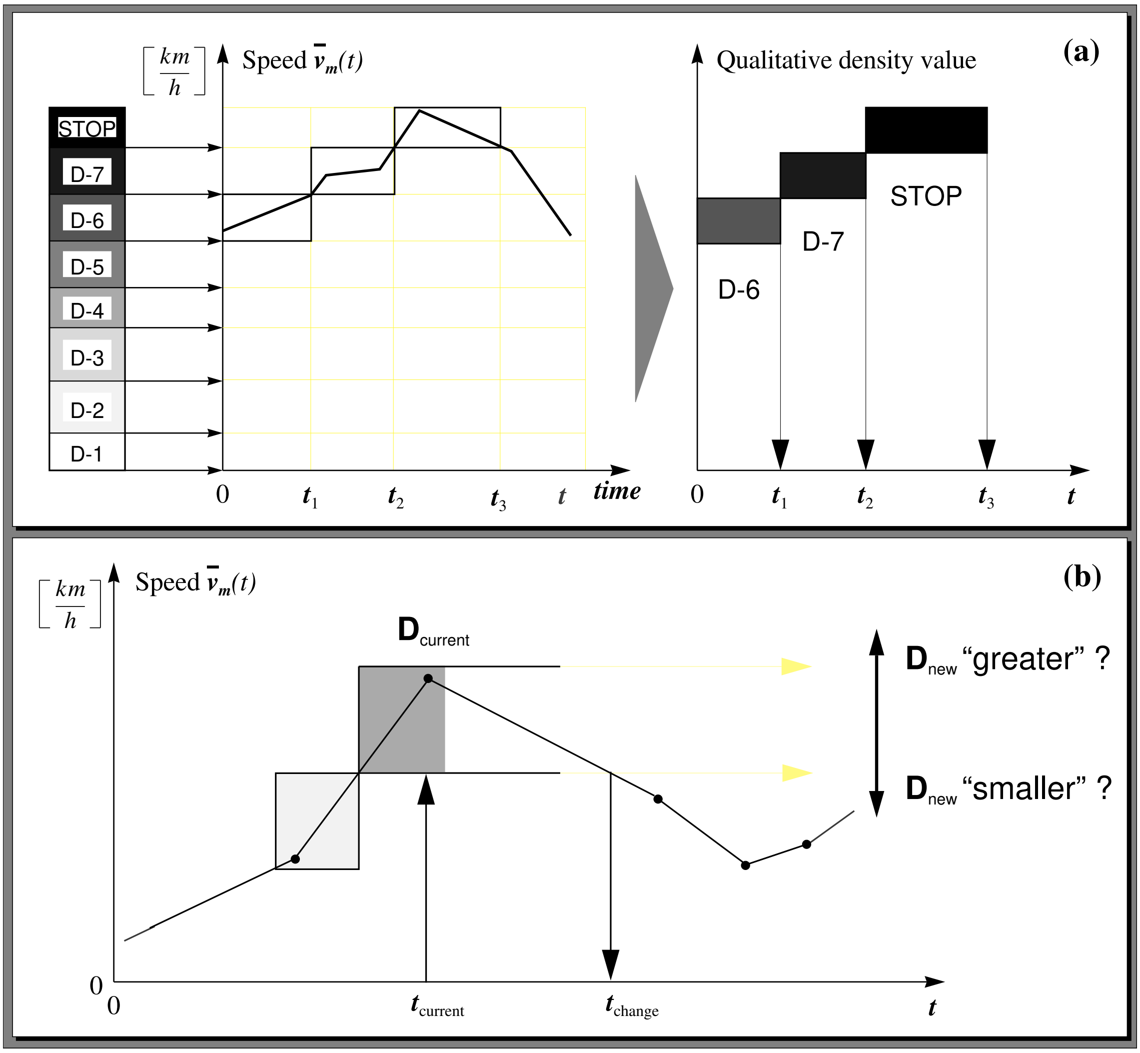


Figure 5.6 Determination of density values: (a) Assignment of quantitative profile values to qualitative density values and (b) determination of the time of the transition to a new density value in a profile

5.3.3 Simulation of Traffic at Marginal Points

The traffic streams which are introduced to the road network at marginal points describe ***profiles*** which display the temporal development of the observed traffic parameters in these points. Available measurement values are traffic volume, average speed and occupation ratio. Later, in the simulator the profiles will be replaces by values observed by detectors for the representation of the current load situation. The transformation of numerical values into qualitative density values, detailed below, is independent of the type of measurement data.

The division of the value range of density, average speed and traffic volume into intervals and the assignment of these intervals to a qualitative density value facilitates mapping of the quantitative measurement data onto qualitative density values (Figure 5.6 (a)). The assignment is performed according to the observed traffic parameter:

a) Measurement of traffic volume:

Here, the assignment of the current volume value to a certain traffic volume interval of a qualitative density value will be examined. As the fundamental diagram is of a parabolic nature, one volume value can be assigned two qualitative density values. Therefore, a definite assignment must be provided by choosing one of the density values on the basis of a certain criterion. The realistic model is based on the assumption that density values of a "close vicinity"[1] in their qualitative description are selected on the basis of the pre-calculated density value. this approach, however, involves that only qualitative density values, which cover one of the fundamental diagram´s parabola branches. This reflects the problem that it is not possible to determine by a mere measurement of the traffic volume whether the traffic gets congested at the moment or whether only few vehicles pass a detector at free flow.

b) Measurement of the traffic volume and the average speed:
As any average speed is definitely assigned to a traffic density value, the qualitative density value, whose interval of average speed includes the current speed, can be determined unambiguously. When the current qualitative density value is determined, only the current speed is taken into account (no comparison with the preceeding density value, no consideration of the volume value).

c) Measurement of the traffic volume or the occupation ratio:
Since it is difficult to determine the traffic density directly [Lapierre & Steierwald 87], this value is assessed indirectly by means of measurement of the load ratio *B* and the formula
$k = B/\bar{l}$,
where $\bar{l}$ is the average vehicle length. $\bar{l}$ is calculated on the basis of the maximum density value k_{max}:
$\bar{l} = 1/k_{max}$.
The resulting numerical density value facilitates a definite assignment of the measurement value to a density value as in b).

Apart from the simple assignment of quantitative measurement values to qualitative density values the simulation in marginal objects includes the calculation of the transition time from the current (qualitative) density value to the next density value. Starting from the current density value and the current simulation time, the course of the profile after this time is regarded and the point of time will be determined at which the numerical values fall below or exceed the borders of the interval assigned to the current density value (Figure 5.6 (b)).

The current qualitative density value is calculated by the method

```
(get-qualitative-density-value
     endpoint time old-density-value-key)
```

1. Cf. order relation in section 2.5.2.

by assigning the corresponding qualitative density value to the quantitative profile value or by reading the qualitative value from the pre-calculated qualitative description of the affluent profile on the basis of the point of time.

```
(get-next-profile-change-time
     endpoint time old-density-value-key)
```

calculates the point of time (in seconds), at which the currently valid density value (**`old-density-value-key`**) changes (in?)to a new density value. In order to avoid the calculation efforts of a transformation of an entire profile during simulation,

```
(create-qualitative-profile endpoint)
```

can generate a qualitative description of a profile before the simulation, which reads as follows:

```
<profile>          ::=    ( {(<time> <density-value>)} ).
```

In the later use of the simulator only few transformation of quantitative measurement data to qualitative density values will be performed during simulation. Only the data observed by the detectors and possibly some forecast values calculated by means of smoothing procedures [Ask 90] will be transformed into qualitative density values at the beginning of the simulation.

5.3.4 Simulation of Traffic on Links

A list of density zones describes the traffic state in a link object in the qualitative model. The dynamical development of the density zone distribution can only partially explained by means of the shock wave approach that was transferred to border speed between the density zones. This approach can explain the movements of the density zones and the appearance and disappearance of density zones at the borders of a link/lane, but it is not capable of modeling the generation of new zones in the middle of a section.

The generation of a density zone at the beginning of a lane/link is caused by a change of the traffic stream entering the lane and thus by its density value. If the border speed between the density zone of the new density value and the density zone at the beginning of the lane is positive, a new density zone (starting with length 0) will spread at the beginning of the section. A new density zone can spread at the end of a lane by a new density value of the leaving traffic stream, if the speed of the new zone border is negative, i.e. if the new zone spread contrary to the direction of the traffic.

To explain the generation of a new density zone on a lane, the present model needs to be expanded by a modeling of the driving behavior of the drivers. If the present model is applied to a situation where, for instance, a free lane precedes a queue on a section, the use of the formula for the speed of the zone border[1] produces a speed of 0, i.e. the queue remains congested. This

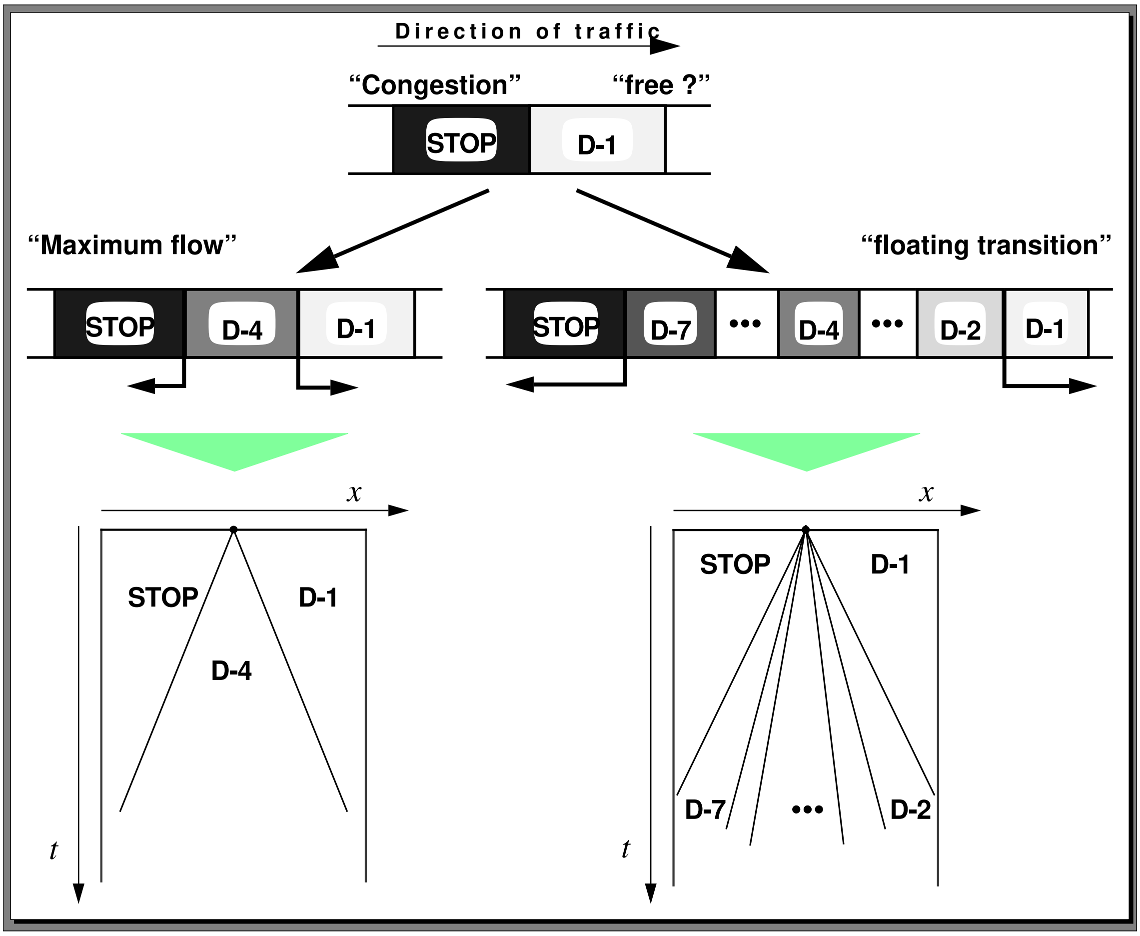


Figure 5.7 Approaches to explain the generation of new density zones on lanes

result complies with the continuum theorem, but not with the driver behavior. To include the driver behavior to the qualitative simulation model, two different assumptions were introduced to the prototype(Figure 5.8):

1) ***“Maximum flow”*** [Moreno et al. 89]:
The drivers in a traffic stream try to reach a state of a greatest possible traffic volume without deceleration at the zone borders. The fundamental diagram describes this state by the area regarded as the optimum state of a traffic flow control: the volume falls slightly under q_{max}, the speed exceeds $\bar{v}_{opt}$ and the density falls under k_{opt}.

2) ***“Floating Transition”***:
The drivers in a traffic stream try to adapt to the traffic situation prevailing in front of them. If there is a free section in front of a queue, for instance, zones of partially restricted traffic, of maximum traffic flow and restricted traffic form in the above order at the transition. The average speed of the vehicles in the new density zones

1. Cf. section 2.5.3

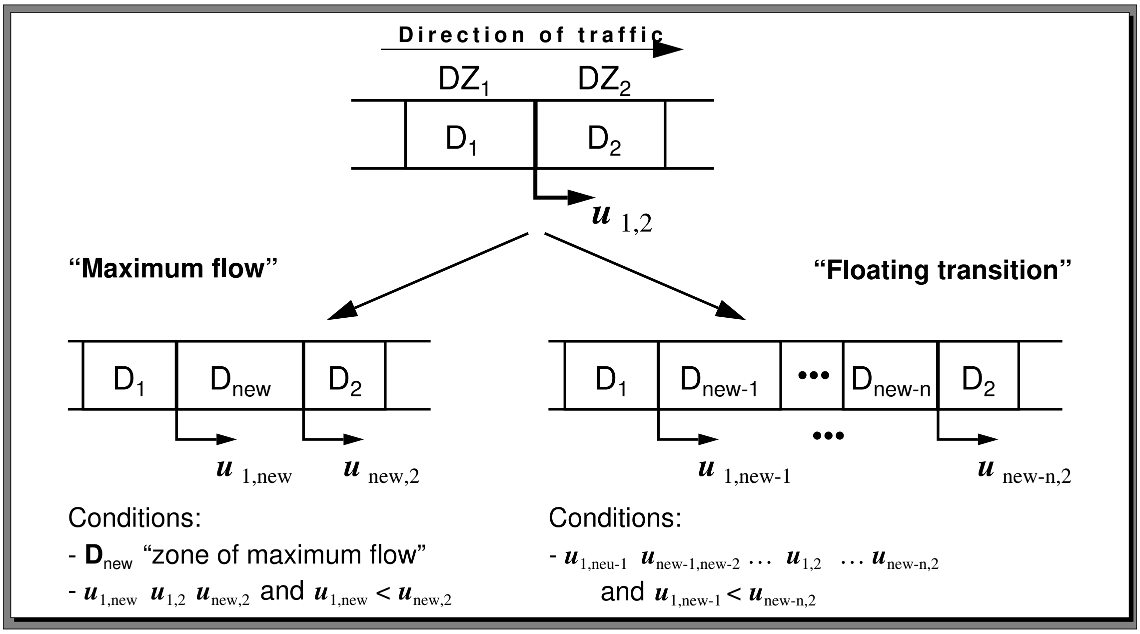


Figure 5.8 Translation of approaches trying to explain the generation of density zones into conditions applying to density values and border speeds

is reduced on the way from the zone in front of the creation point to the succeeding zone.

Both approaches to the generation of new density zones between existing zones must be described by the conditions applying to the border speed of the zones and the density values of the involved zones(Figure 5.8):

ad a) “*Maximum Flow”*:

To insert a zone, the following conditions must be fulfilled:

(i) the density value of the zone must describe the area of maximum traffic flow (inc. the maximum traffic volume value?),

(ii) the length of the new intermediate zone must increase, i.e. the border speed at the beginning of the new zone must be smaller than the border speed at the end of the zone:

$$u_{1,\,neu} < u_{neu,\,2},$$

(iii) the border speed between the two already existing density zones must be between the speeds of the new zone borders:

$$u_{1,\,neu} \le u_{1,\,2} \le u_{neu,\,2}.$$

ad b) *“Floating Transition”*:

To insert new density zone the conditions (ii) and (iii) on the speeds of the zone borders being generated must be fulfilled just like in a). The following applies to the density values of the inserted density zones:

(i´) the density values of the new density zones are “between” the density values of the two already existing zones[1].

1. Cf. order relation in section 2.5.2.

“maximum flow”		Density value of zone <u>after</u> border							
		D-1	D-2	D-3	D-4	D-5	D-6	D-7	STOP
dens.value of z. <u>in front of b.</u>	D-1	-	-	-	-	-	-	-	-
	D-2	-	-	-	-	-	-	-	-
	D-3	-	-	-	-	-	-	-	-
	D-4	-	-	-	-	-	-	-	-
	D-5	(D-4)	(D-4)	(D-4)	-	-	-	-	-
	D-6	(D-4)	(D-4)	(D-4)	-	-	-	-	-
	D-7	(D-4)	(D-4)	(D-4)	-	-	-	-	-
	STOP	(D-4)	(D-4)	(D-4)	-	-	-	-	-

“Floating transition”		Density value of zone <u>after</u> border							
		D-1	D-2	D-3	D-4	D-5	D-6	D-7	STOP
dens. value of z. <u>in front of b.</u>	D-1	-	-	-	-	-	-	-	-
	D-2	-	-	-	-	-	-	-	-
	D-3	(D-2)	-	-	-	-	-	-	-
	D-4	(D-2 D-3)	(D-3)	-	-	-	-	-	-
	D-5	(D-2…D-4)	(D-3 D-4)	(D-4)	-	-	-	-	-
	D-6	(D-2…D-5)	(D-3…D-5)	(D-4 D-5)	(D-5)	-	-	-	-
	D-7	(D-2…D-6)	(D-3…D-6)	(D-4…D-6)	(D-5 D-6)	(D-6)	-	-	-
	STOP	(D-2…D-7)	(D-3…D-7)	(D-4…D-7)	(D-5…D-7)	(D-6 D-7)	(D-7)	-	-

Figure 5.9 Matrices describing the feasibility of the generation of a density zone between two zones corresponding to the two approaches.

The rule-based implementation (PROLOG) of a similar simulation model [Moreno et al. 90] describes the conditions on the insertion of density zones into lanes on the basis of rules which allow to deduce state transitions. In contrast to this, an implementation in the functional programming language LISP the lists of the qualitative density values, which are legal for the zones and can be inserted between two zones, are already calculated before the simulation starts and stored as matrix (**`new-density-zone-table`**, Figure 5.9) to the objects of the “density calculus”. The function

```
(new-intermedium-zone-p
     density-value-new density-value-1 density-value-2)
```

can query during simulation whether a new zone can be inserted between two density zones or not.

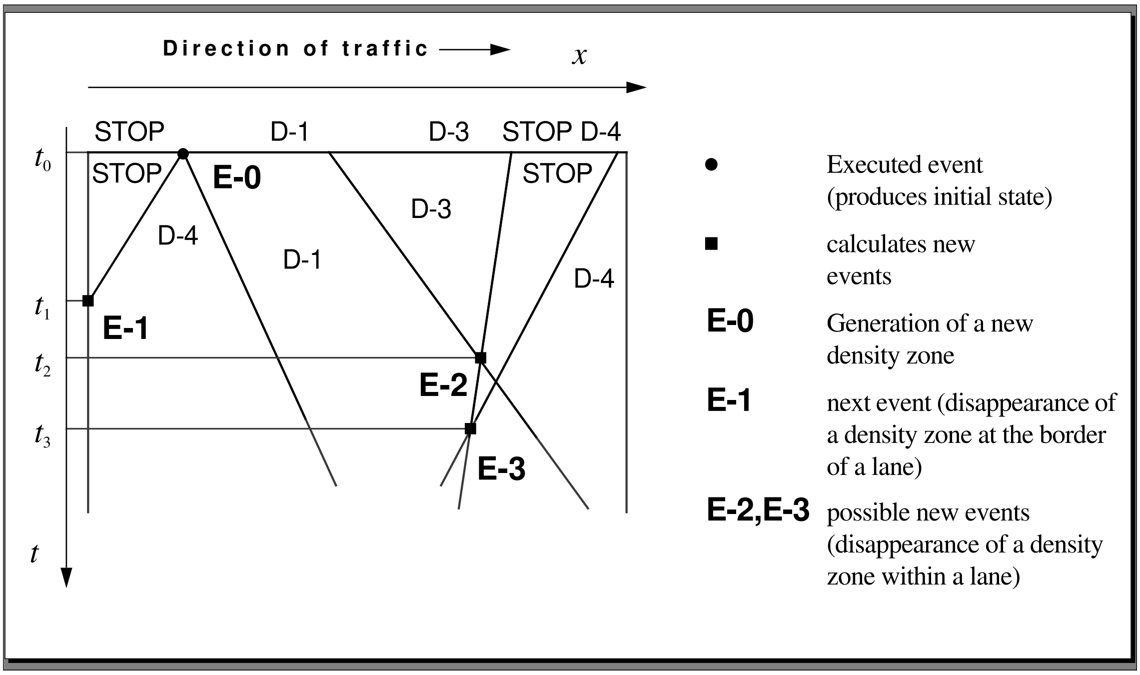


Figure 5.10 Determination of a new density zone distribution on a lane

State transitions are also calculated by means of LISP functions in this simulator prototype. Starting from a given density zone distribution, it is determined whether new density zones can be inserted. If possible, zones will be inserted. then, it will be examined whether new density zone were generated at the borders of the lane. If this is true, messages on the new density value will be transmitted to an adjacent simulation object; if not, it will be examined, whether zones have already shrunk to length 0. These zones will be removed from the density zone distribution.

If a density zone does not immediately disappear, the movements of the zone borders in the time-distance diagram will be analyzed(Figure 5.10). First, the set of possible points of intersection of the line representing the movement of the zone borders are determined themselves. The intersections in the time-distance diagram describe the new events. The event with the earliest point of time is chosen from the set of possible new events. The density zone distribution is determined for the time of the next “internal event”, which describes the then valid traffic state. This event will be stored while the other calculated events are discarded.

The

```
(compute-next-link-event old-event)
```

function implements the procedure for calculating the state transition on a lane. Based on the description of the old event `old-event` (point of time and old density distribution), it determines the new event in the link object. If a new density zone is to generated at the beginning or end of a lane, the

```
(add-new-density-at-link-end
                    link old-event next-event time)
```

function is called. Based on the old density zone distribution (in **old-event**), it calculates the distribution prevailing at the time (**time**) when the new zone is inserted (sender of the new density value and the new density value itself in **next-event**). Then, the zone of the new density value (length 0) will be inserted at the end or beginning of the lane.

This simple simulation model of traffic flow on a section does not comply with the continuity theorem of traffic. Nevertheless, significant state transitions can be deduced. This was proved in a concrete situation/setting on the one hand [Toledo et al. 91], on the other hand it was stated in the validation of the simulation model implemented here by comparison with the results of the microscopic model in the simulator MISSION.

5.3.5 Simulation of Traffic on Intersections

The following sections provide detailed information on the event handling of intersection objects. An event on an intersection is always the result of a change in the qualitative density zone determination at the end of the adjacent lanes. They result either as an external event from the disappearance of a density zone at the end of the lane or as an internal event from the stage change in the intersection´s signal plan. A temporal change in the turning ratios of a lane will also result in an internal event of the intersection object. The ***Sapporo*** system, however, does not support the latter one up to now. Each lane of the current prototype has only one constant turning distribution. During processing of an event a new viscious state must be generated in the density zone distribution of all adjacent lanes. The change of a density zone value at the intersection end of a lane can effect changes in the density zones of other lanes in this intersection.

5.3.5.1 Criteria for Density Zone Determination

A list of criteria valid for the traffic flow and its qualitative description by density zones is provided by the Spanish engineering group in [Moreno 90]. These criteria must be taken into account when computing the density zones anew after an event.

1) The total of the traffic flows from all incoming lanes equals the total of all flows that lead on to the departing lanes.
2) The turning distribution of the flow of each lane ending at the intersection must be taken into account.

3) If a change in traffic flow causes a new density zone in al lane, this zone must propagate on the adjacent stretch of road, i.e. the resulting limit speed between the old and the new zone points out of the intersection. This is referred to as a stable situation. If this was not taken into account, the generated density zone would immediately disappear at an immediate event handling of the corresponding stretch of road and would result into a new density zone modification as a parallel external event on the intersection. Consequently, the simulation algorithm would run into a deadlock.

4) According to the above regulation a maximum flow is produced on the intersection. It corresponds to the natural driver behavior when crossing an intersection.

5) If two qualitative density zones, which have an identical maximum flow, are possible on the basis of the quadratic flow-density relation, a value is produced which differs fewest from the original density zone value. Thus, the transition from one density zone value to another one excludes a jump from the ascending to the descending branch in the density-flow graph of the fundamental diagram and vice versa.

6) A density zone with a maximum density is produced at the end of all incoming lanes of the intersection which are in the red phase. The density zone will always propagate on the stretch of road. A new zone of a minimum density is produced in the beginning of all departing lanes of the intersection which do not receive a traffic flow. This zone will also always propagate.

The most obvious approach to determine the new density zone state, used by Moreno, Toledo, Rosich and Martin in [Moreno et al. 90], suggests to check all possible density zone distributions for their compliance with the above criteria. This approach proved to be successful for a minor simple traffic network in the Spanish town of Valencia. Looking at the asymptotic effort of the density zone assignments, which are to be checked, an asymptotic effort of $O(k^n)$ can be assessed for an intersection with n adjacent lanes and k qualitatively different density zones. In principle each lane can adopt any new density zone value. This effort is out of the question for larger and more complex intersection as they are to be modeled in the ***Sapporo*** system.

The criteria 1 and 2 are based on the comparison of a quantitative flow value. As the description of the traffic load at the entries and exits of the intersection is based on traffic flow intervals, described by qualitative density zones, in our model, a computational model calculus is required for computation with this traffic flow intervals. It aims at an examination of whether the current density zones at the entries and exits describe a quantitative flow assignment complying with the criteria 1 and 2. The density zone assignment would be valid in this case. The necessary computational model uses interval arithmetic.

Intervals are added via the addition of the higher and the lower interval boundaries of the summand interval. The width of the summation interval always equals the total of the widths of the summand intervals.

For multiplication of an interval by a scalar, its lower and higher interval boundary is multiplied by the scalar. This is required, for instance, when checking the second criterion for the assessment of the turning streams.

For the comparison of the intervals the intersection of two intervals is regarded. If it is not empty, values result which are contained in both intervals. When checking criterion 1, there is, for instance, a valid quantitative flow value assignment for the entries and exits, if the totals of the intervals of entries and exists overlap.

Thus, criteria 1 and 2 can be checked when using qualitative density zones.

5.3.5.2 Extension of Criteria

One important criterion for realistic modeling and simulation of the traffic flow on an intersection is not taken into account in the computational algorithm by [Moreno et al. 90]. The criterion 4, as detailed above, requires a maximum flow value on the intersection. This will produce a true description of driver behavior in the case of a lane that starts in the intersection and receives only one traffic stream coming from one incoming lane. This does not correspond to reality in most cases. As soon as one lane receives two or more traffic streams, where the maximum flow is limited, the question on distribution of the flow capacity to the incoming traffic flows arises, if the total exceeds the maximum flow. An example is to explain the problem.

Given the turning distribution depicted in Figure 5.11, lane1 splits into two lanes, lane3 receives the flows of lane1 and lane2 at the same time. The maximum flow value of each lane is standardized to 1.0. At the point of time t_0 the maximum flow in lane1 and lane2 is 1.0. What is the behavior of the traffic streams in lane3 like? According to the calculation rules with unchanged criteria the density zone providing the maximum flow on the intersection is selected from all possible density zone distributions. In this example the maximum flow is produced by the subsequent distribution:

Lane1: 1.0, Lane2: 0.1, Lane3: 1.0, Lane4: 0.1

Thus, the total flow on the intersection equals the flow total of all incoming streams, i.e. it equals 1.1.

With a distribution of 1.0 of the maximum flow at the exit of lane3 to the incoming lane1 and lane2, the entry of lane1 is attached greater importance than the entry of lane2, as the increase of lane1 to lane4 depends on the increase of the flow of lane1 to lane3, if the turning

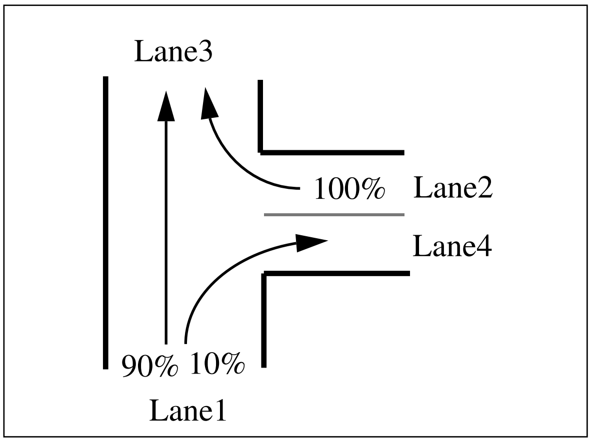


Figure 5.11 An example for an intersection with a turning distribution

distribution is to be taken into account according to criterion 1. An increased flow from lane1 thus produces a stronger increase in total flow than an equally increased flow from lane2.

In reality, the exit flow from lane3 could be distributed 0.5 to the traffic flow of lane1 and 0.5 to the traffic flow of lane2, which corresponds to the equal turning of the streams of lane1 and lane2 into lane3 known as the *merging traffic principle*. Such a distribution would produce the subsequent flow in the intersection:

Lane1: 0.555, Lane2: 0.5, Lane3: 1.0, Lane4: 0.055

In this case the total flow of the intersection would be 1.055.

Taking the merging traffic principle into account when entering into a lane, the maximum flow in the intersection will be smaller than without this criterion. The extension of criterion 4 by the traffic merging principle reads as follows:

- A maximum flow is produced in all lanes, where all n streams entering an exit may share a minimum of $1/n$ of the maximum flow of this exit.

This extension will only come into effect, when the total of the possible incoming traffic flows exceeds the maximum exit flow. If one of the incoming streams does not claim this share of $1/n$ of the exit stream, the available flow difference can be distributed to the other incoming traffic streams.

In the case of more complex intersections and “unfavorable” turning distributions the computation of a new density zone distribution without the merging traffic principle would result in non-negligible errors in the density zone calculation, as was shown for the flow in lane1, which produced the value 1.0 without merging traffic principle and 0.555 when the extended criterion was taken into account.

The merging traffic principle describes a method to integrate two or more traffic streams, where each traffic stream has similar entry rights to the resulting traffic stream as it is the case with merging lanes. In reality, however, there are hardly situations on intersections where the merging traffic principle is made use of. Here, the entry of traffic streams to other lanes is governed by the “priority to the right” rule or the “law of the jungle”. Therefore, geometric relations between the lanes of an intersection must be taken into account for realistic modeling of intersections without merging lanes. In the abstracting model presented here we assume that merging lanes exist everywhere and that the merging traffic principle is made use of.

5.3.5.3 Density Zone Determination in Sapporo

An integration of the extended criterion 4 to the original computational algorithm of [Moreno et al. 90] increases the time required for computation by the checking of the extension. The efforts, which are enormous anyhow, are still increased.

Even if the suggested determination of a new density zone assignment was clearly extendable by a control system based on the presented criteria and by any number of further criteria, the time required for computation will soon exceed the real time demanded for the simulation of road traffic.

In the ***Sapporo*** traffic guidance system the determination of density zone assignments is therefore replaced by a less effort-intensive newly developed iterative algorithm which is presented as a flow chart in Figure 5.12.

The starting parameters for the calculation of a new density zone distribution are a list of all lanes adjacent to the intersection including their current density zone values, the current phase situation on the intersection and finally a list of the turning distribution of the incoming lanes, which are represented as matrix

$$S = (s_{i,j}) \qquad i = 1,2,...,E,\ j = 1,2,...,A$$

for our algorithm where E denotes the number of incoming lanes, A the number of exit lanes and $s_{i,j}$ the proportion of the traffic flow from entry i to exit j. The following restrictions apply to the matrix elements.

$$0 \le s_{i,j} \le 1 \qquad \text{and} \qquad \sum_{j=1}^{A} s_{i,j} = 1$$

A1: Determine all lanes where traffic may flow.

A2: Determine the density zones of the lanes not participating in the traffic flow (greatest and smallest density zone, respectively).

A3: Determine all density zones for each lane participating in the traffic flow, which produce a stable situation.

A4: Determine the maximum flow for each of the above lanes:

$e_{i,\,max}$ $(i = 1,2,...,E)$ for entry lanes,

$a_{j,\,max}$ $(j = 1,2,...,A)$ for exit lanes.

A5: initialization of the entry flows:

$$e_{i,\,0} = \min\left(e_{i,\,max}\quad, \left(\forall j: 1 \le j \le A \wedge s_{i,\,j} \neq 0: \left(\frac{a_{j,\,max}}{n_j} \times \frac{1}{s_{i,\,j}}\right)\right)\right)$$

A6: Initialization of the exit flows:

$$a_{j,\,0} = \sum_{i=1}^{E} s_{i,\,j} \times e_{i,\,0}$$

A7: Iterate new flow values of the entries

$$e_{i,\,k} = \min\left(e_{i,\,max}\quad, \left(\forall j: 1 \le j \le A \wedge s_{i,\,j} \neq 0: \left(e_{i,\,k-1} \times \frac{a_{j,\,max}}{a_{j,\,k-1}}\right)\right)\right)$$

A8: Iterate new flow values of the exits

$$a_{j,\,k} = \sum_{i=1}^{E} s_{i,\,j} \times e_{i,\,k}$$

A9: Determine the total of all flow differences of this iteration step

$$D_k = \sum_{j=1}^{A} (a_{j,\,k} - a_{j,\,k-1})$$

$D_k > \varepsilon\,?$

YES (back to A7)

NO

A10: Determine the qualitative density zones corresponding to the flow values found out.

Figure 5.12 New iterative computational algorithm for density zone determination

Step A1 of the algorithm defines the lanes where really traffic flows, i.e. entries i, where $s_{i,j} \neq 0$ applies to at least one j and exits j, where $s_{i,j} \neq 0$ applies to at least one i. Only these will be considered in the following. Entry lanes which are blocked by the current phase situation will be assigned the highest density zone value (`STOP` zone) in step A2. There traffic will be congested in front of the traffic lights. Exit lanes in which no traffic flows, i.e. the lanes j, where $s_{i,j} = 0$ applies to all i, will be assigned the lowest density zone value (`D-1`) in step A2. This is an "empty" zone of the traffic density 0, i.e. no traffic follows.

In step A3 all possible density zones, leading to a stable situation at the end of the intersection, are determined, i.e. A3 determines the density zones whose border speeds at the limit of the current density zone value on the intersection is diametrically opposed to the intersection as required by criterion 3, cf. above.

In A4 the maximum possible quantitative flow value of each lane is computed from the maximum of all flow interval mean values of the possible density zones which have been determined in A3. The variables $e_{i,\,max}$ $(i = 1,...,E)$ result for the E entry lanes and the variables $a_{j,\,max}$ $(j = 1,...,A)$ for the A exit lanes. The computed maximum value is the maximum of each possible flow in one particular lane so that another stable situation is created. Note that a minimum of one corresponding qualitative density zone, which generates a stable situation, must exist for any flow value smaller than the maximum value for the subsequent iteration of the lanes´ flow values.

The existence of such a density zone can be explained by the fundamental diagram in Figure 5.13. The parabolic form of the diagram produces for each flow value below the maximum flow for entry lanes a secant of a negative gradient through a point of the curve, which corresponds to the flow value, and the point P, which in turn corresponds to the current density zone. Likewise, each flow value below the maximum flow of the exit lanes has a secant of a positive gradient in the diagram between a point of the curve of this flow value and the point P, which represents the current density zone.

In the graphic to the left in Figure 5.13 all points of the curve, whose connecting line through point P of the current density zone value has a positive gradient, are marked black. This section of the curve covers the entire value range of 0 to the maximum flow for exits on the flow axis of the diagram. Contrary, all points of the curve, whose connecting line through point P of the current density zone value has a negative gradient, are marked by a shaded line. This section of the curve also covers the entire value range of 0 to the maximum flow for entries, which corresponds to the current density zone, on the flow axis. The graphic to the right in Figure 5.13 depicts the same scenario, where the current density zone marks a point P on the right branch of the flow density chart.

After determination of the maximum flow for all lanes, A5 initializes the entry flows for the iteration. In this context each entry may send a maximum of $1/n$ of the exit´s maximum

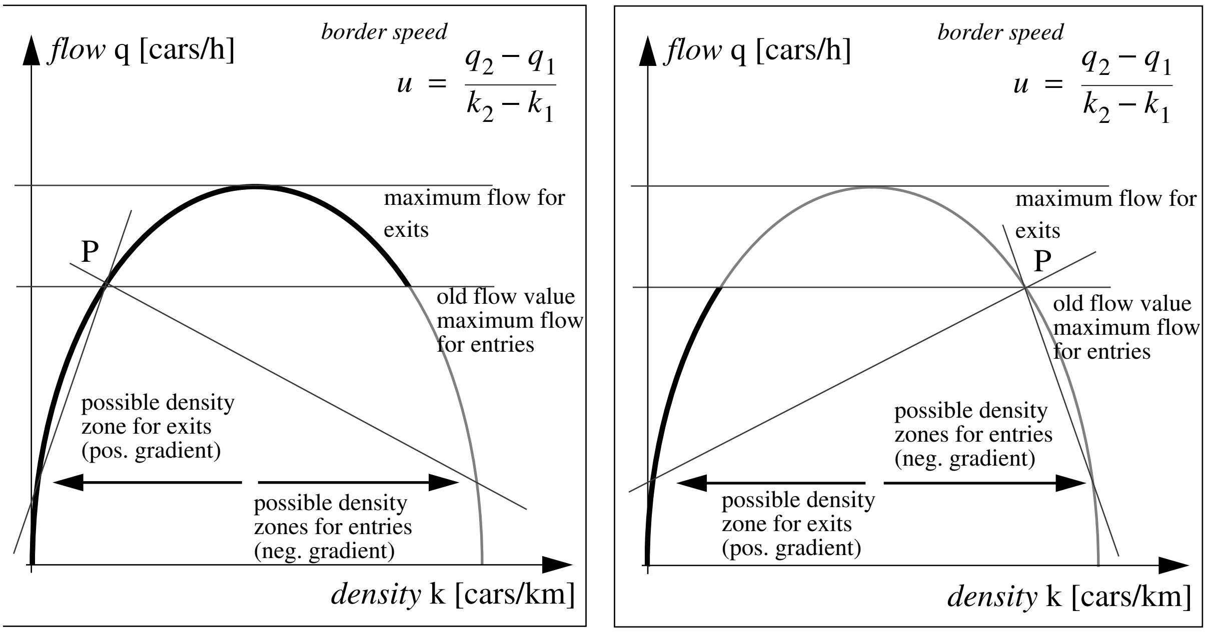


Figure 5.13 Sections of the fundamental diagram producing a stable situation where the old density zone value P is placed on the left and on the right branch of the diagram, respectively.

flow to each exist it is linked to via the turning distribution, i.e. $(a_{j,\,max}/n_j)$. n_j stands for the number of all streams received by exit j. The entry flow which exactly produces the allowed exit flow exceeds the latter one by the factor of $1/s_{i,j}$ at a turning ratio of $s_{i,j}$.

Thus, the initial value of the entry flows is the minimum of all flow values “allowed” by the exits.

$$e_{i,0} = \min\left(e_{i,\,max} \quad , \left(\forall j : 1 \le j \le A \wedge s_{i,j} \ne 0 : \left(\frac{a_{j,\,max}}{n_j} \times \frac{1}{s_{i,j}}\right)\right)\right)$$

For the initialization of the exit flows, all entry flows, which have been computed in A5, are added in A6 with regard to their turning ratio:

$$a_{j,0} = \sum_{i=1}^{E} s_{i,j} \times e_{i,0}$$

Since none of the entries requires a flow larger than $1/n$ of the maximum flow of the exits receiving the entries, the computed flow total definitely does not exceed the maximum value at the exits; it might possibly be lower than the allowed flow maximum. Therefore, $(a_{j,0} \le a_{j,\,max})$ applies.

The regulations A7 to A9 represent the iteration loop via the k parameter, $(k = 1,2,3,...)$, which tries to reach a parity distribution of the flow difference between the maximum flow and the flow finally required by the entries to all incoming streams in each iteration loop. If an entry cannot fully exhaust its nth share in an exit, because it is limited in its flow by other exits receiving the entry, for instance, this share can be equally distributed to the other entries which compete for the maximum exit flow. So, more than $1/n$ of the maximum exit flow can be introduced into the exit by an entry, therefore $(s_{i,j} \times e_i) > (a_{j,max}/n_j)$ applies.

Each iteration loop increases the flow claimed in the last loop and granted by the exits to the connected entries by the factor $a_{j,max}/a_{j,k-1}$. Thus, the entry flows of the kth iteration loop result in

$$e_{i,k} = \min\left(e_{i,max} \quad , \left(\forall j: 1 \le j \le A \wedge s_{i,j} \neq 0: \left(e_{i,k-1} \times \frac{a_{j,max}}{a_{j,k-1}}\right)\right)\right)$$

In each iteration step the exit flows are computed from the turning distribution with regard to the weighted sum of the iterated entry flows, which are received by it:

$$a_{j,k} = \sum_{i=1}^{E} s_{i,j} \times e_{i,k}$$

In A9 D_k, which is the total of the flow increases at the exits computed in the kth iteration loop is regarded as a truncation criterion:

$$D_k = \sum_{j=1}^{A} (a_{j,k} - a_{j,k-1})$$

If it is smaller than a predefined value ε, the iteration is truncated and the iterated flow values $e_{i,k}$ and $a_{j,k}$ are re-calculated to the corresponding qualitative density zones in A10.

5.3.5.4 Proof of Correctness

The newly-developed algorithm for the determination of a new density zone situation on intersection complies with all criteria detailed above. The entry flow into the intersection equals the exit flow in each iteration step, which is proved by the subsequent equation. It includes the relations between exit flow and entry flow (A8) and the precondition for the turning distribution used in the algorithm.

$$\sum_{j=1}^{A} a_{j,k} = \sum_{j=1}^{A} \sum_{i=1}^{E} (s_{i,j} \times e_{i,k}) = \sum_{i=1}^{E} e_{i,k} \times \left(\sum_{j=1}^{A} s_{i,j} \right) = \sum_{i=1}^{E} e_{i,k}$$

The turning distributions of the entry lanes, as was demanded by criterion 2, remain unchanged by the assignments in A8. The generation of a stable situation (criterion 3) is guaranteed by instruction A3. Starting from the flow situation on the intersection, which complies with the criteria 1 to 3, the flows of the corresponding lanes approach their optimum via an increase in each iteration loop. If a lane reaches its maximum flow value during iteration, a maximum traffic flow prevails on the intersection, where the merging traffic principle according to the extended criterion 4 is taken into account. If none of the lanes reaches its maximum value, the control parameter ε can determine the extent to which the computed flow values are to approach their optimum. Criterion 5 complies with instruction A10, while criterion 6 corresponds to instruction A2. The instruction A10 complies with the selection of the density zone, which comes closest to the old density zone in the fundamental diagram, demanded by criterion 5. Instruction A2 is responsible for the assignment of the lanes not involved in the traffic flow (criterion 6).

To prove the correctness of the algorithm, it remains to be shown that the existing iteration loop terminates after a finite number of steps. For this purpose we must prove that the difference D_k between old and new exit flow total becomes smaller than the predefined value $\varepsilon > 0$ after a finite number of iteration steps. The subsequent theorem states this.

Theorem 1

The sequence $D_k = \sum_{j=1}^{A} (a_{j,k} - a_{j,k-1})$ of the total of the iterated exit flow difference is a zero sequence.

$$\lim_{k \to \infty} \sum_{j=1}^{A} (a_{j,k} - a_{j,k-1}) = 0$$ applies.

According to the summation theorem for converging sequences, the total of two zero sequences results again into a zero sequence ([Heuser 84] Bd.1, S.153). Therefore, it suffices to show that the summands, i.e. the flow difference of each exit, represent a zero sequence in the sequence D_k. Consequently, only the subsequent toned down theorem 2 remains to be proved.

Theorem 2

The iterated flow difference at each exit of the intersection forms a zero sequence.

$\lim_{k \to \infty} (a_{j,k} - a_{j,k-1}) = 0$ applies to all exits $j = 1,2,...,A$

To prove this theorem we need some important relations between the iterated exit flow values which are stated by the following auxiliary theorem and are to be derived first.

Theorem 3

The following applies to the sequence of the flow values of each exit on the intersection in the new algorithm:

a) $0 \le a_{j,k} \le a_{j,max}$ applies to all $k = 1,2,3,...$

b) The sequence $(a_{j,k})$ increases monotonously. $a_{j,k} \ge a_{j,l}$ applies to all $k > l$.

To prove theorem 3.a, the upper limit of the flow of each entry, which is received by the examined exit, can be estimated. It will be definitely smaller than or will equal the flow allowed in the examined exit.

$$
\begin{aligned}
a_{j,k} &= \sum_{i=1}^{E} s_{i,j} \times e_{i,k} \\
&= \sum_{i=1}^{E} s_{i,j} \times \min\left(e_{i,max}, \left(\forall m: 1 \le m \le A \wedge s_{i,m} \ne 0: \left(e_{i,k-1} \times \frac{a_{m,max}}{a_{m,k-1}}\right)\right)\right) \\
&\le \sum_{i=1}^{E} s_{i,j} \times \min\left(e_{i,max}, \left(e_{i,k-1} \times \frac{a_{j,max}}{a_{j,k-1}}\right)\right) \\
&\le \sum_{i=1}^{E} s_{i,j} \times \left(e_{i,k-1} \times \frac{a_{j,max}}{a_{j,k-1}}\right) \\
&= \frac{a_{j,max}}{a_{j,k-1}} \times \left(\sum_{i=1}^{E} s_{i,j} \times e_{i,k-1}\right) \\
&= a_{j,max}
\end{aligned}
$$

The first conversion replaces the entry flow value $e_{i,k}$ by the formula defined in A7 in the kth iteration step. In the following two conversions in this formula the upper limit of the minimum of all legal entry flows is estimated by one entry flow currently allowed by exit j. The last factor $(a_{j,max}/a_{j,k-1})$ independent of i can be put in front of the summation. According to A8 the remaining total is the exit flow $a_{j,k-1}$, which has been determined in the last iteration step. It can be canceled in the last step.

To show the monotony of the sequence $(a_{j,k})$, we replace the entry flow value $e_{i,k}$ in the total by the formula of A7 in the first conversion. Then, the relation of theorem 3.a, stating that $a_{j,k-1} \le a_{j,max}$ and thus $(a_{j,max}/a_{j,k-1}) \ge 1$ applies, will be used to estimate the lower limit of the legal entry flows increased by the factor $(a_{j,max}/a_{j,k-1})$ in the minimum term. The third conversion uses the relation $(e_{i,k} < e_{i,max})$, as the specification of $e_{i,k}$ does never exceed the entry flow maximum in A7. The last conversion is based on the definition of the exit flow as it is determined in the kth iteration step in A8.

$$
\begin{aligned}
a_{j,k} &= \sum_{i=1}^{E} s_{i,j} \times e_{i,k} \\
&= \sum_{i=1}^{E} s_{i,j} \times \min\left(e_{i,max}, \left(\forall m\colon 1 \le m \le A \wedge s_{i,m} \ne 0\colon \left(e_{i,k-1} \times \frac{a_{m,max}}{a_{m,k-1}}\right)\right)\right) \\
&\ge \sum_{i=1}^{E} s_{i,j} \times \min(e_{i,max}, e_{i,k-1}) \\
&= \sum_{i=1}^{E} s_{i,j} \times e_{i,k-1} \\
&= a_{j,k-1}
\end{aligned}
$$

As long as $a_{m,k-1} < a_{m,max}$ applies to all exit flows $a_{m,k-1}$, i.e. as long as none of the exits which receive the ith entry has reached its maximum, the “>” relation and thus a strict monotony applies.

By means of the auxiliary theorem 3 we can now prove theorem 2.

After initialization of the entry and exit flows $e_{i,0}$ and $a_{j,0}$ the flows increase strictly monotonously until an arbitrary entry or exit has reached its maximum flow value. Inevitably the flow will not be changed in the next iteration step and the flow difference turns out to be zero.

Thus, it sets off an avalanche, as the corresponding exits and entries, respectively, can no longer increase their flow value in the next iteration step etc.

Therefore, the following can be deduced: If an entry´s or exits´s flow difference $(e_{i,k} - e_{i,k-1})$ or $(a_{j,k} - a_{j,k-1})$ has once become zero, it will remain zero in the following iteration steps.

If none of the entries or exits reaches its flow maximum during initialization or the subsequent iteration, the sequences $(e_{i,k})$ and $(a_{j,k})$ increase strictly monotonously and the following assessment can be made for the flow difference at each exit.

$$
\begin{aligned}
(a_{j,k} - a_{j,k-1}) < (a_{j,max} - a_{j,k-1}) & \\
&= (a_{j,max} - a_{j,k-2}) - (a_{j,k-1} - a_{j,k-2}) \\
&= (a_{j,max} - a_{j,k-2}) - \left(\frac{a_{j,k-1} - a_{j,k-2}}{a_{j,max} - a_{j,k-2}}\right) \times (a_{j,max} - a_{j,k-2}) \\
&= (a_{j,max} - a_{j,k-2}) \times \left(1 - \frac{a_{j,k-1} - a_{j,k-2}}{a_{j,max} - a_{j,k-2}}\right) \\
&\dots \\
&= (a_{j,max} - a_{j,0}) \times \prod_{p=1}^{k-1} \left(1 - \frac{a_{j,p} - a_{j,p-1}}{a_{j,max} - a_{j,p-1}}\right) \\
&= \alpha
\end{aligned}
$$

In the last term $(\prod (1 - \delta))$ the flow values $a_{j,p}$ are always smaller than $a_{j,max}$ according to the above preconditions. Thus, $(1 - \delta) < 1$ applies and the product is therefore a zero sequence.

$\lim_{k \to \infty} \alpha = 0$ applies.

The upper limit of this zero sequence assessed, the flow difference $(a_{j,k} - a_{j,k-1})$ is also a zero sequence (“comparison theorem for converging sequences” in [Heuser 84] vol.1 p.152) QED for theorem 2. Thus, the termination of the iteration loop of the algorithm has also been proved.

The asymptotic efforts required by the iterative algorithm presented in this paper amounts to O(k) for the conversion of the k qualitative density zones into a quantitative flow value and vice versa, O($p_\varepsilon \times n$) for flow determination in the iteration, where p_ε stands for a constant maximum value depending of the setting of the abort criterion ε. The total efforts, thus, amount to O($k + n$).

This new method for density zone determination performs a quantitative computation of traffic flow states prevailing on the intersection contrary to the original approach. Thus, a rule-based description and usage of the six criteria stated above is not possible. An extension or improvement of the criteria will always affect the computational algorithm. The main advantage of the compact numerical computation by means of the presented iteration algorithm is the linear order of computation, so that a real-time simulation can be also guaranteed for relatively large traffic networks, such as the network of the Japanese town of Sapporo.

6 Implementation

6.1 Environment

The simulator prototype was developed in a CommonLISP environment, where LISP constructs of the quasi standard from [Steele 90] were used. As LISP has been designed especially for symbolic programming [Winston & Horn 87], the algorithms for symbolic calculation can be easier and faster implemented into other programming languages. Symbolic calculation is of prime concern in the qualitative simulation in contrast to the numerical simulation. The functional language LISP features some characteristics which make it seem suitable for fast prototype implementation:

- the LISP interpreter facilitates an incremental interactive program development without the annoying *EDIT-COMPILE-LINK* cycle (interpretation instead of compilation), i.e. modifications can be tested at an earlier stage,

- the genealogy mechanisms of the object system CLOS ([Symbolics 90a], [Lawless & Miller 91]) facilitate the adaptation of existing object classes to modified requirements and the expansion by additional object classes (e.g. new classes for further types of simulation objects),

- the integration of CommonLISP to the extensive software engineering environment Genera [Walker et al. 87].

The interface development system CLIM (CommonLISP Interface Manager [McKay 91]) based on CommonLISP provides a useful tool for the design of the user interface. With regard to the already implemented user interface we decided for the Dynamic Windows System [Symbolics 90b] which had been used until now. The transformation of the extended user interface into the CLIM standard can readily be performed, but was omitted in this paper for reasons of a fast development of a prototype. Macintosh FILEmaker II used for data input on network objects in forms will now be also used for entering the fundamental diagram or the k-$\bar{v}_m$ relation.

The programming environment (hardware/software) included:

1) Symbolics Ivory system on a Sun SPARCserver 4/370 (Genera 8.1),
2) Symbolics MacIvory system on a Macintosh II (Genera 8.1),
3) Symbolics 3620 workstations,
4) Sun SPARCstation 1, Sun SPARCstation 2 (SunOS 4.1, X-Windows 11R4),
5) NeXTstation.

The simulator prototype and in particular the graphics interface were conceived to operate independent of the underlying hardware.

6.2 Program Structure

In the implementation of the object-oriented draft the simulation objects and the objects of the “density calculus” are defined first. It uses the existing subclasses which identify and categorize the entities similar to the network objects. Figure 6.1 and Figure 6.2 display the class hierarchy of the new defined object classes. These objects are to be graphical-interactively generated, modified, inspected and removed by means of an editor just as the network objects. *SAPPORO* has already data base functions to manage (network) objects. For this reason the methods defined for the network objects are implemented class-specific for the new objects or

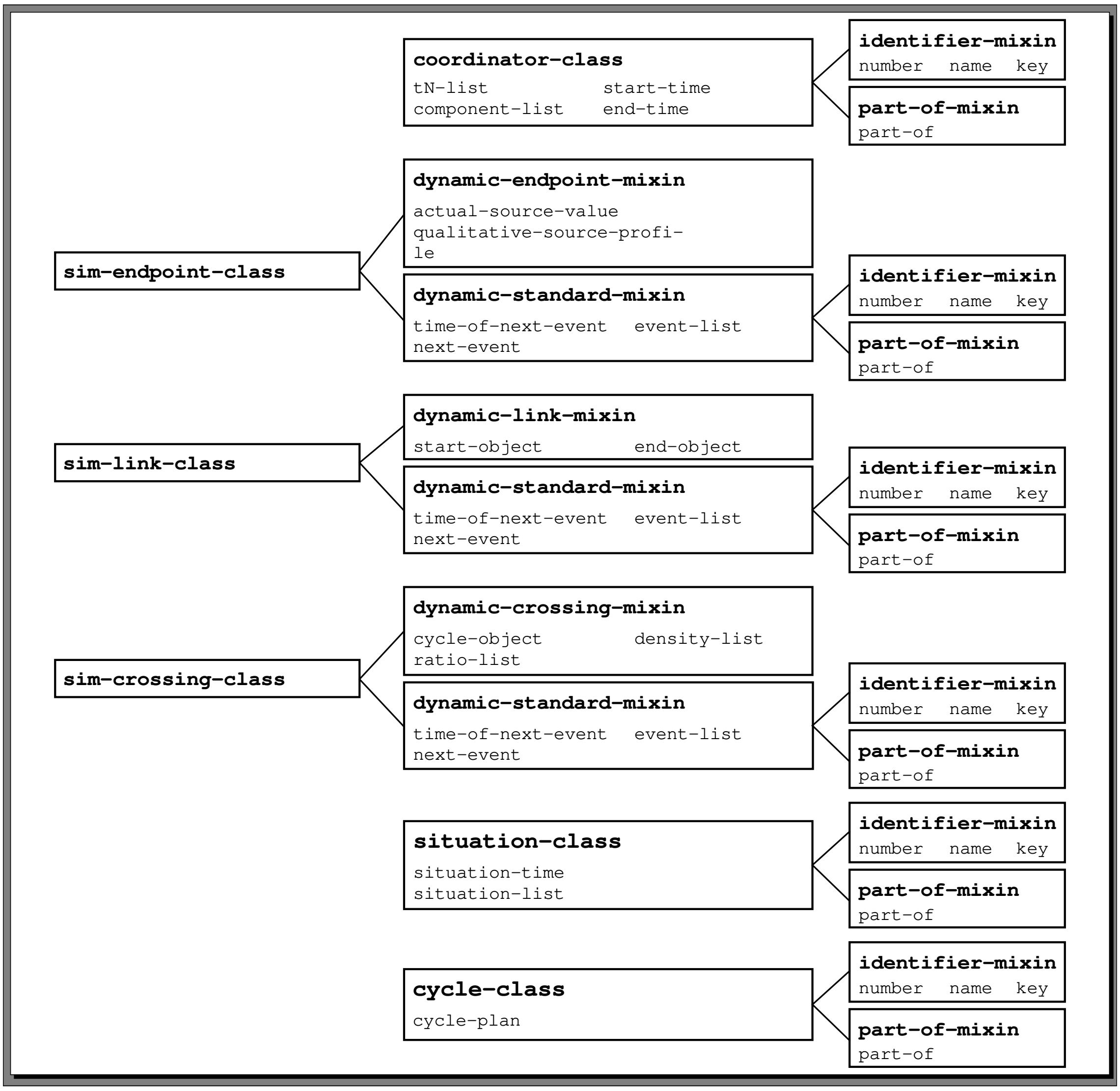


Figure 6.1 Overview of the objects required for simulation

existing functions extended so that simulation objects and objects of the "density calculus" are capable of

- generating new objects and removing existing objects,
- modifying the slot values of the objects,
- loading and storing a generated "density calculus",
- reading in the textual description of a fundamental diagram (in form of a FileMaker II document),
- storing and loading a simulated traffic state as situation in order to use it as start situation for new simulation runs.

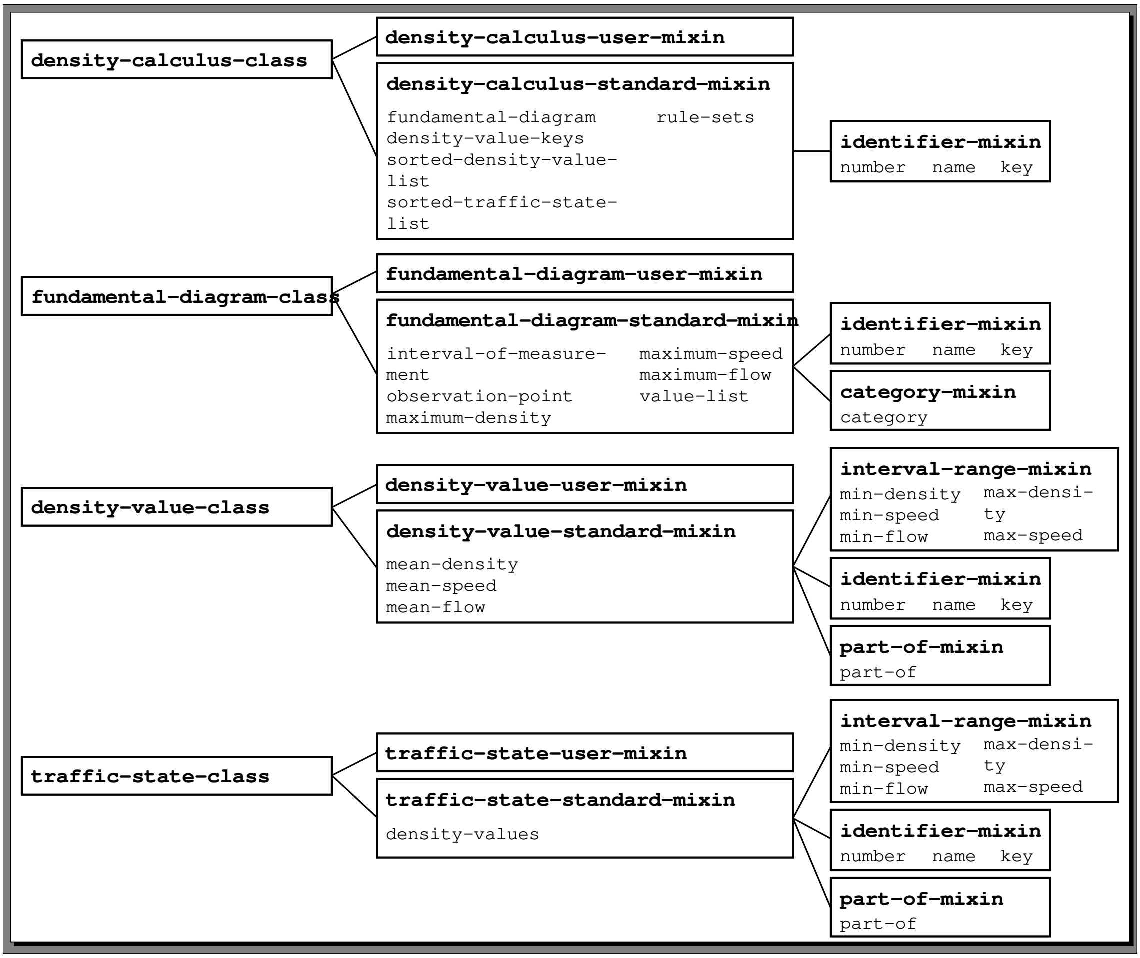


Figure 6.2 Overview of the objects of the density calculus.

Moreover, the anew defined objects are to be integrated to the existing *object browser* which allows to look at the entities of an object class, the list of its slots and the slot values in tables. The variety of methods required for this must be extended by methods for the new object classes. Within the programming structure of the *SAPPORO* system the methods for object management are effected within the modules **BROWSER** and **DATABASE** (Figure 6.3). To call the new methods, the menu items of the user interface defined in the **SAPPORO-SYM** must be extended. Furthermore, the functions to generate pop-up menus must be implemented in this module. They allow to process user input when generating a "density calculus", starting the simulation and in the output of results.

The modules, which had to be generated anew for the implementation of the simulation prototype, will be introduced in the following overview of the programming structure. then, the basic functions and methods[1] within the modules will be explained.

1. A comprehensive overview of all functions is provided by the "SAPPORO **Software Documentation**" [Wild et al. 92a].

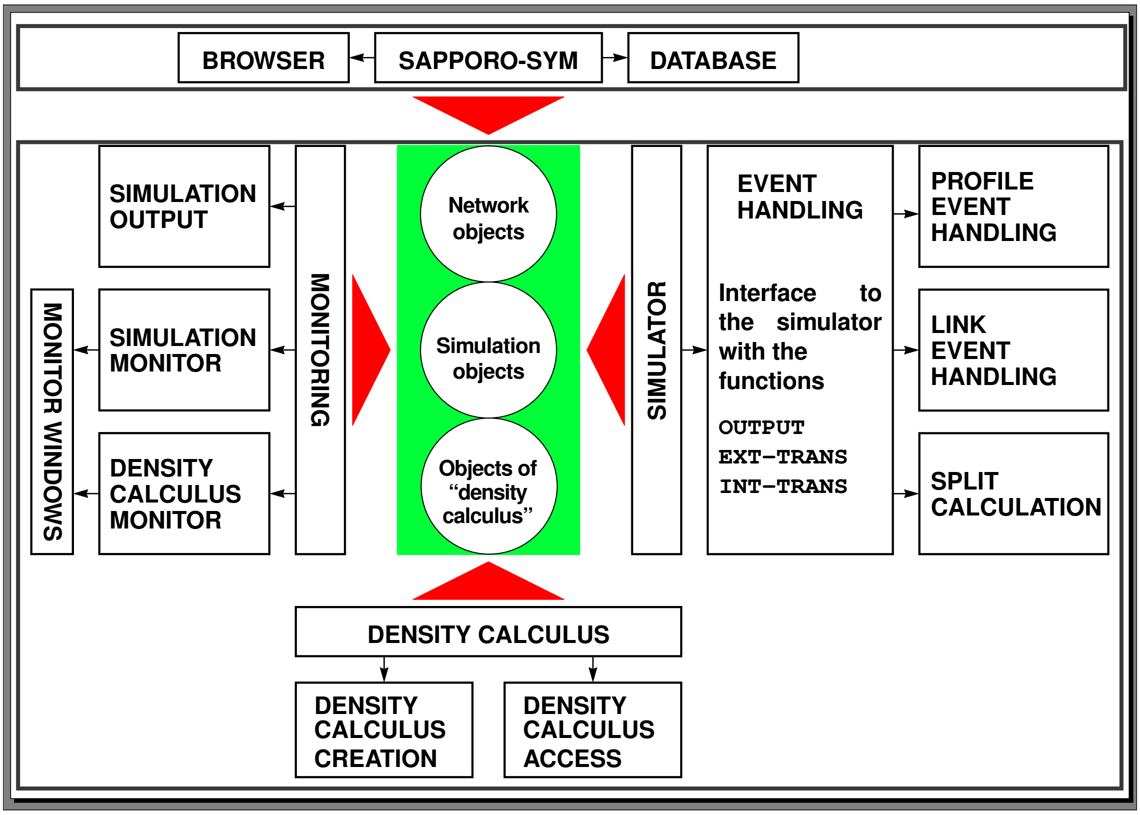


Figure 6.3 Programming structure of the simulation component of *SAPPORO*

6.2.1 Modularization

While objects are regarded as closed units in the object-oriented design, for which a set of operations and utilities are defined, the implementation is based on functions which are improved step by step. The functions that have been assigned to certain objects in the draft and determine the object´s behavior, are combined into modules [Parnas 72]. So, contrary to the functional programming paradigm a non-hierarchical programming structure (Figure 6.3) results, which is characteristic for object-oriented systems. The assignment of functions and methods to the four main modules reflects partial tasks, which are solved within the simulation system, on the one hand:

1) The **SIMULATOR** module contains the methods of the coordinator object, which manages the simulation object during simulation. The methods are implemented in a way so that the simulation can be performed independent of the concrete classes of the simulation objects.
2) The **EVENT-HANDLING** module contains the generic functions based on the DEVS specification (**OUTPUT**, **EXT–TRANS** and **INT–TRANS**), which the coordinator object needs to perform a discrete event-oriented simulation. The methods are defined class-specific for the three classes of simulation objects.

3) The **DENSITY-CALCULUS** module contains the methods to generate qualitative density values on the basis of a fundamental diagram and to modify and remove all objects of the “density calculus”.

4) The **MONITORING** module contains the methods to output the “density calculus”, to animate the simulation runs and to display the simulation results in form of a diagram.

On the other hand modularization facilitates parallel development of the components. So, interface methods manipulating objects in other modules have been implemented as *dummy* methods first, which generate pre-calculated output values for the corresponding input. Thus, it was possible to implement the simulator kernel before implementing the methods for event handling. The component for output of the simulation objects was completed by means of the simulator, thus simplifying the evaluation of the simulation runs performed for the purpose of tests.

6.2.2 Module Description

The methods and functions combined in modules can be divided into interface functions and internal (secret) functions. While interface methods are called in other modules to execute operations via certain objects, the internal methods and functions are required to perform tasks within the modules. To provide ready access to all objects in all modules the following global variables have been defined:

- **`*current-network*`** to access all network, simulation and “density calculus” objects,
- **`*current-density-calculus*`** to access all lists and matrices generated for the “density calculus”,
- **`*sim-coordinator*`** to access the coordinator object, which manages the simulation objects.

The methods of the coordinator object used to control intercommunication between the simulation objects and event handling in these objects, are combined in the **SIMULATOR** module:

- **`(simulation-install network)`** to generate the simulation objects corresponding to the network objects via **`(create-sim-object node network)`** and **`(create-sim-object lane network)`**,

- **`(build-tN-list coordinator)`** and **`(tN-insert coordinator - object timepoint)`** to generate and manage the list of entries on new events in the simulation objects,

- **`(simstep coordinator timepoint)`** to perform the simulation at an event time, where all objects signaling a next event for this point of time, are processed, i.e. output and internal transition functions of these objects as well as the external transition functions of all objects receiving a message are called,

- **`(compute-traffic-state coordinator time)`** to determine the list of traffic states of all simulation objects and **`(compute-simobject-state coordinator object-key time)`** to determine the state of a simulation object.

The generic functions[1] called by the coordinator for event handling in the simulation objects, which facilitate an event-oriented discrete simulation, are defined in the **EVENT-HANDLING** module:

- **`(OUTPUT sim-object time)`**,

- **(INT-TRANS `sim-object time`)**,

- **(EXT-TRANS `sim-object time message`)**,
  as well as

- **`(set-object-state sim-object init-state)`** to initialize a simulation object,

- **`(get-state-at-timepoint sim-object time)`** to determine the traffic state in a simulation object.

The corresponding object-specific methods are implemented for all classes of simulation objects. These methods access functions which facilitate the calculation of new events in the objects, and they are distributed to three submodules:

1) **PROFILE-EVENT-HANDLING** for event handling in marginal objects contains the methods **`(get-qualitative-density-value endpoint time old-density-value-key)`** to determine the qualitative density value, which is to be currently introduced into the network, and **`(get-next-profile-change-time endpoint time old-density-value-key)`** to determine the switch-over time to the next density value. Functions transforming quantitative traffic parameters into qualitative density values are effected internally. The new density values can be directly determined on the basis of profiles on the one hand, on the other

1. Cf. section 4.3.1

hand a qualitative description of a profile can be generated by **`(create-qualitative-profile endpoint)`** and the density values can be read from it.

2) **LINK-EVENT-HANDLING** for event handling in link objects contains the function **`(compute-next-link-event old-event)`** for calculating an event within a lane, i.e. the disappearance of generation of a density zone. New density zones are generated by means of **`(add-new-density-at-link-end link old-event next-event-time)`** at the start or end of a lane on the bases of a modification of the qualitative density value at the transition to the adjacent object. Contrary **`(compute-link-messages old-event new-event sending-object-key)`** can determine whether a modification of the density zone distribution has occurred at the start or end of a lane and whether a new density value must be signaled to the adjacent object.

3) **SPLIT-CALCULATION** for event handling in crossing objects contains the function **`(split-calculation ratios linkvalues inlinks outlinks)`** to determine a new assignment of density values at the transitions to adjacent lanes. The current turning ratio, which is required for the calculation, is determined via **`(get-new-ratios cross time)`**. **`(get-next-cycle-change-time cross time)`** calculates the time of the next phase change in the signal plan used to control the intersection. At this point of time a new assignment of density values must be calculated anyway. The messages on new density values sent to adjacent objects are determined via **`(compute-crossing-messages old-dens-list new-dens-list sending-object-key)`**.

During event handling functions will be called which allow access to and operations with the objects of the “density calculus”. These functions are defined in the **DENSITY-CALCULUS** module:

- **`(get-border-speed density-value-1 density-value-2)`** to determine the border speed between two density zones,

- **`(new-intermedium-zone-p density-value-new density-value-1 density-value-2)`** to check whether it is possible to generate a new zone between two density zones,

- functions to determine density values of higher or lower average values of traffic flow (**`flow`**), average speed (**`speed`**) or density (**`density`**),

- functions to determine the “minimum”, “maximum” or “next higher” density value according the order relation defined on the qualitative density values, The objects of the “density calculus” are generated via

`(create-density-calculus network)`

after loading the network, where a pre-defined fundamental diagram is loaded. Apart from functions for the generation of qualitative density values and traffic states as well as for the calculation of matrices containing the border speeds of the zones and the lists of density values, which can be inserted between two density zones, access functions to these objects are also implemented in this module. Additionally, it has functions which update the pre-calculated matrices and lists in case of modifications of the intervals assigned to the density values.

All functions and methods which allow the graphical output of the objects of the “density calculus” are defined in the **MONITORING** module.

- **`(show-fundamental-diagram fundamental-diagram)`** to display the fundamental diagram (Figure 6.7 (b)),

- **`(show-profile endpoint)`** to display the profile in a marginal point of the road network,

- **`(show-border-speed-table density-calculus)`** to display the matrix on the border speeds,

- **`(show-new-density-zone-table density-calculus)`** to display the matrix on the insertable density values.

The results of a simulation run produce the following functions:

- **`(show-event-list link)`** shows the development of the des bi-dimensional parameter (qualitative) traffic density on one lane in the time-distance diagram (Figure 6.7 (c)),

- **`(show-link-profile link)`** shows the qualitative profile, which was generated during simulation, for each node on a lane (Figure 6.7 (d)).

The module also contains the functions to visualize traffic on the lanes. similar to the animation during simulation, it is possible to perform the animation after the simulation for the entire network or parts of the network was terminated. The post-animation of a simulation run is started via **`(re-display-simulation-run coordinator Th)`**.

6.3 User Interface

To increase the acceptance of a simulation system, it is necessary to adapt the system to the ideas and the proceeding of the later user. This system has been conceived on the one hand for the planning traffic engineer, who wants to analyze the effects of modifications in the control strategy of a traffic guidance system off-line. Access to applications of the qualitative simulation must be facilitated in particular for this planner. For this reason it is necessary to design an ergonomic and clearly organized graphics interface. The system is to support the user in all working steps of a simulation study in a computer via this interface. On the other hand the simulator is to be used to forecast traffic states on-line, where the interface is of minor importance; a real-time-capable implementation of the simulator is required above all. The graphical display only serves control and checking purposes.

6.3.1 Computer-Supported Working Steps

The phases of a computer-supported simulation study comprise the following working steps in the *SAPPORO* system:

- modeling by means of:
    - a qualitative description of the interrelations between the macroscopic traffic parameters and
    - the generation of a model structure on the bases of the given road network,
- performing the simulation experiments,
- displaying the simulation results.

Thus, several computer-supported partial working steps in the simulator prototype result (Figure 6.4). The user starts to enter the road network in which the traffic flow is going to be simulated. Then, a “density calculus” can be generated using an entered fundamental diagram or an already generated “density calculus” can be loaded. Subsequently, the system itself generates the objects of the simulation network, i.e. it generates a simulation object for each node and each lane of the network and initializes it by means of an initial state. During the simulation the user can have the simulated traffic on the lanes displayed or only watch the simulation time. After termination of the simulation it is possible to perform a post-animation of the simulation run. The development of the density zone distribution can be displayed in time-distance dia-

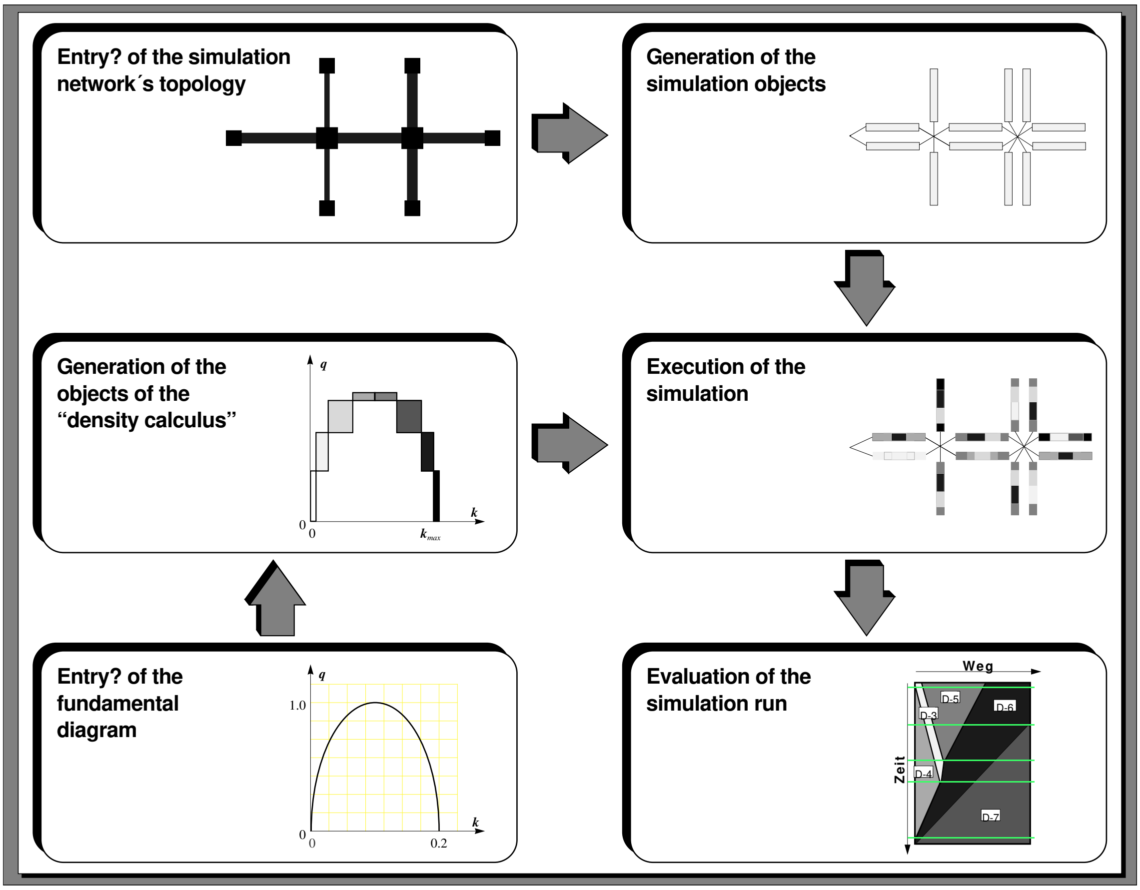


Figure 6.4 Computer-supported working steps in a simulator prototype

grams for each lane and the simulated profile can be shown for each node on the lane. To use the final state or an intermediate state of a simulation in a new simulation run as the initial state, traffic states can be saved as situations.

The dialogue structure of the simulator prototype allows a flexible structuring of the process of work, i.e. the user decides when to execute certain working steps and when to use or leave a certain interface. Certain work sequences, however, are supported by menu assistance (Figure 6.5).

6.3.2 Graphical Interface

When designing the user interfaces in *SAPPORO*[1], transparency was attached prime importance to facilitate easy orientation for the user. To guarantee ergonomic interfaces, the following principles governed the design:

- Each interface must be clearly identifiable.

1. An extensive explanation of the user interface and an example session in provided in the ***“SAPPORO User´s Manual”*** [Wild et al. 92b].

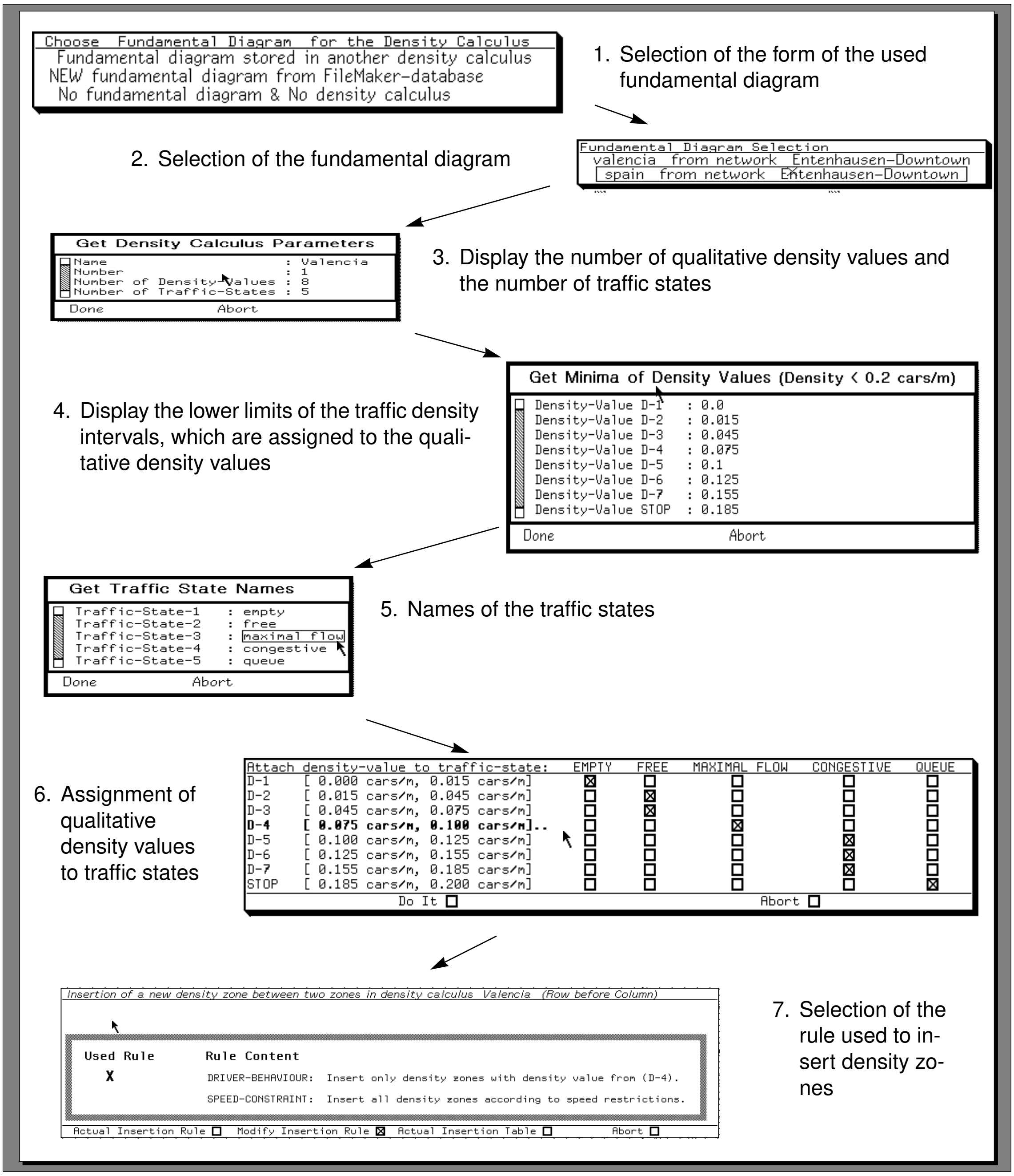


Figure 6.5 Dialogue structure of the generation of a “density calculus”

- The menus are small and clearly structured.
- The order of the menus always remains the same.
- All objects (in symbolic or graphical representation) can be selected in a graphical-interactive way.
- All actions can be invoked in a graphical-interactive way.

The reliability of the user interface is guaranteed in so far as missing or incorrect input parameters cannot start the actions linked to the selected menu item. An extensive error list and *HELP* functions were omitted within the framework of prototype design.

In *SAPPORO* the user interface divides into one main interface and several secondary interfaces. The structure of the user interface is based on the working steps of a simulation study. The involved actions are represented as items in the main menu.

The main interface of the simulator prototype fulfills several tasks. On the one hand it provides the entry interface for the user, on the other hand it facilitates a look at all generated objects by means of the *object browser*, drawing the street map as well as execution and animation of the simulation run. Correspondingly, the main interface consists of several functional areas (Figure 6.6). The upper area is reserved for the object browser. The entities of all object classes including their slots and slot values can be regarded there. Below, there are two windows for the display of the road network or the simulation network, showing the traffic state on the lanes at a certain point of time or the animation of a simulation run. Either the entire network or a part of it can be displayed. The window to the right pane contains a menu bar including all commands. The small window at the bottom of the screen displays the commands entered via the menu and facilitates textual entry of LISP commands.

While the secondary interface A (Figure 6.7 (a)) is a modification of the main interface, the secondary interfaces B, C and D (Figure 6.7 (b)-(d)) are designed anew: secondary interface B displays the objects of the “density calculus”, C the simulation results in form of trajectory diagrams time-distance diagrams and D pre-defined or simulated profiles. The secondary interface are constructed alike: the head line shows the description of the displayed diagram (with a relation to the corresponding object) and the footer contains the commands for modifying the display. A table or a diagram is located in the middle of the screen. These secondary interfaces can be called by clicking the appropriate symbolic representation of an object of the “density calculus” in the object browser, the graphical representation of a network object in the window containing the street map of a link object in the animation window by means of the mouse.

Apart from this there are pop-Up menus listing information on network or simulation objects. If a command requires values for entry parameter by the user, pop-up menus will also be generated, where the user can enter own values for these parameters or confirm pre-defined default values (cf. Figure 6.5).

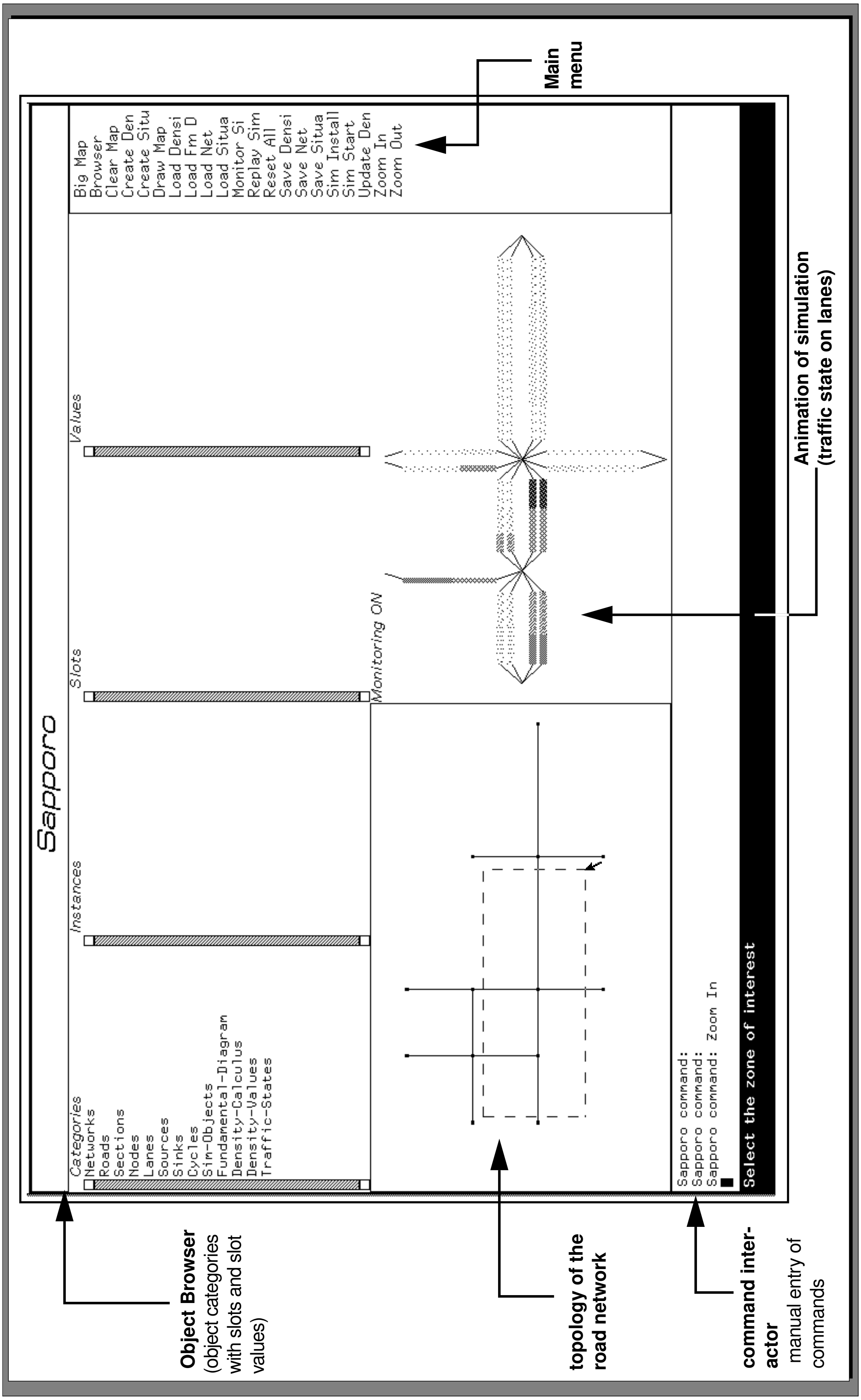
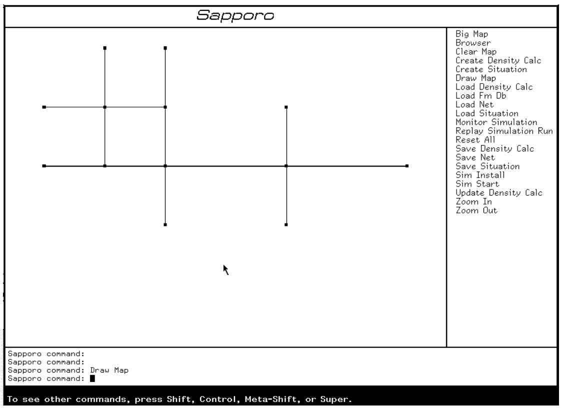
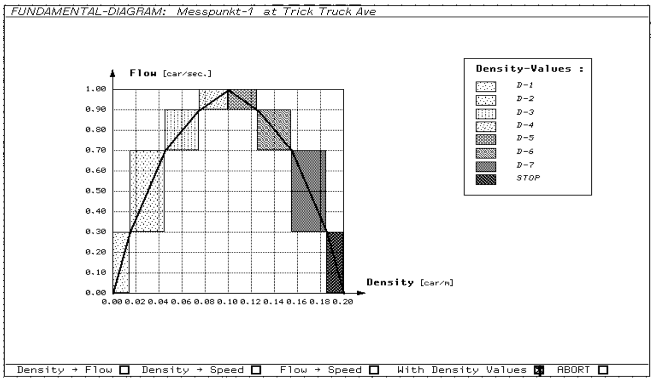


Figure 6.6 Main user interface of *SAPPORO*

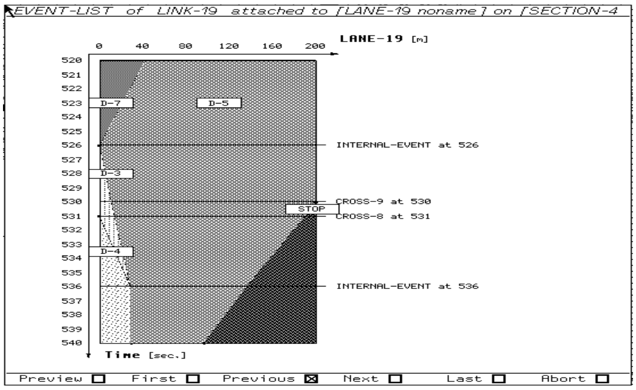
(a) Interface A: Topology of the road network



(b) Interface B: objects of the density calculus



(c) Interface C: time-distance diagram (event list)



(d) Interface D: profiles

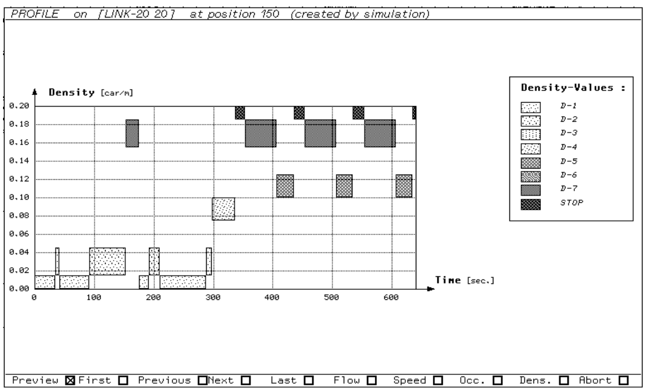


Figure 6.7 New designed additional interfaces

7 Prototype Test

7.1 Overview

We tried three street maps to test the functionality of the simulator (Figure 7.1). The test network **ENTENHAUSEN–DOWNTOWN**, a small fictitious street network, was used for functional tests and for a performance test to determine run-time intensive components of the simulator. **HAMBURG–SMALL** represents a part of the Hamburg traffic network, for which true measurement data on the traffic flow are available. To validate the qualitative simulation model, experiments were carried out in the test network **HAMBURG–SMALL** both by means of the macroscopic (qualitative) simulator of *SAPPORO* and by means of the microscopic (numerical) simulation system MISSION. A third network, **SAPPORO–DOWNTOWN**, where the simulation system is going to be used later on, was used to test the performance of the simulator prototype in larger traffic networks.

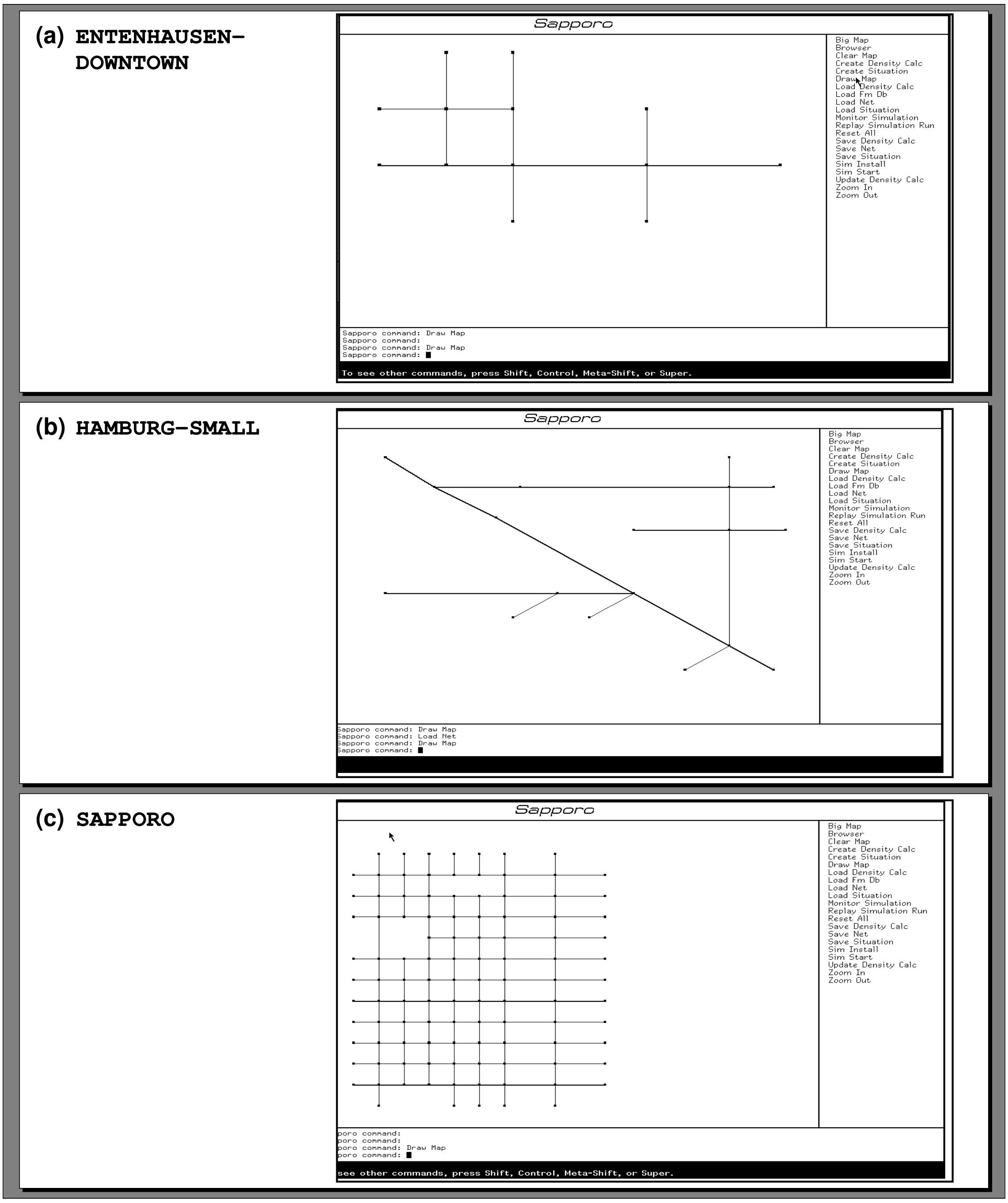


Figure 7.1 Street maps used for the tests

7.2 Test Environments

For the comparison of the two emulators the traffic network modeled in MISSION and its dynamic performance were mapped to the ***Sapporo*** model. A simulation with equivalent dynamic input parameters was performed. The test network is depicted in Figure 7.2. The width of the displayed sections varies between one and four lanes for each direction. The description of the MISSION network structure by network objects in ***Sapporo*** was more difficult. MIS-

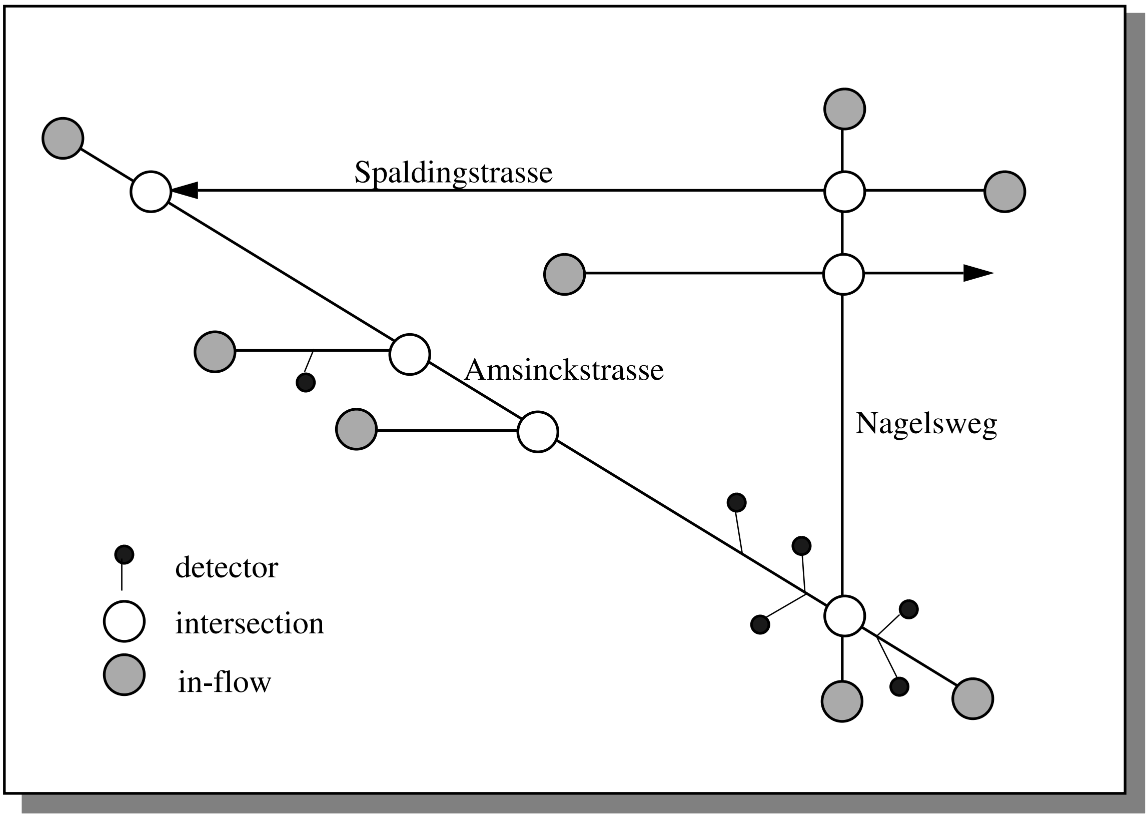


Figure 7.2 Test network for the simulation. Part of the street network of Hamburg

SION and ***Sapporo*** do not only represent the two different types of microscopic and macroscopic traffic simulators, but as a result of this the specifications of the underlying model also differ to a great extent.

In MISSION the road network is defined by “sections”, “links” and “turning decisions”, while in ***Sapporo*** the same network is defined by the data objects “roads”, “sections”, “lanes” and “nodes” as well as “cross-overs” and “turning ratios”. This causes the following problems:

- Differences in linear data of the lanes between start->end and end->start of individual sections cannot (yet) be reproduced in ***Sapporo***. These differences are caused by winding sections and the lanes´ length design, the geometry of the intersection included. This is not possible in ***Sapporo*** because of the desired simplicity of the model. We decided for the average value for each lane length for both directions. Furthermore, it is not possible to assess coordinates for the intersections on the basis of the network description in MISSION. Mapping them onto a ***Sapporo*** model, where the length of a section equals the geometric distance between two nodes, would cause a lot of trouble. After an arbitrary assessment of the intersections´node coordinates, we initialized the sections with their actual lengths. Therefore, it is not possible at the current state of the implementation to produce a display on the screen.

- Dedicated turning lanes in front of intersections are not yet feasible in ***Sapporo***, i.e. a constant number of lanes connect adjacent intersections on the entire section. These turning lanes are also important within the ***Sapporo*** system with regard to the modeling of turning events and the assignment of traffic lights to certain lanes on intersections. The introduction of a new category of the type “node” `(category :2-crossing)`, which describes the distribution of n lanes to m lanes, would provide a solution. In the test model the turning lanes have been neglected and their turning relations have been attributed to “ordinary lanes”.

- As the lane-changing behavior of vehicles will not be modeled in the ***Sapporo*** system, we included the distribution of the traffic to the various parallel lanes to the turning relations in ***Sapporo***. The traffic which turns from one lane into destination section is equally distributed to all available lanes of the destination section.

- The dynamical description of the traffic behavior is provided by a freely definable polygon curve of the speed distribution and the internal constants for the simulation of driver behavior of individual vehicles in MISSION. The behavior in the ***Sapporo*** system is based on a fundamental diagram of the macroscopic traffic parameters density, speed and flow. This difference in modeling is one of the greatest potential factors of inaccuracy in the comparing simulation of both systems. Therefore, the correct and favorable choice of the fundamental diagram deserves particular attention in ***Sapporo***. For this purpose we extracted a fundamental diagram, which is characteristic for the test network, based on extensive measurements carried out by means of the MISSION simulator (cf. below).

- Incoming flows are given in $[V/h]$ in MISSION, all lanes of a section added. The numbers are valid for a certain period of time until a new affluent value is defined. In ***Sapporo*** the sequence of the flow is indicated in $[V/s]$ per lane. Interpolations are carried out between every two sequence values. As MISSION does not indicate any other parameter (speed, density, occupation ratio), we assumed for our system that the higher speed, i.e. the lower branch of the fundamental diagram is selected.

- Signal plan objects are not yet available in ***Sapporo***. The changes in the *turning ratios* effected by the fixed-time signal plan of the MISSION model have been implemented (with great efforts) by a particular function which emulates the time-dependent changes of traffic parameters on intersections.

7.3 Generation of a Fundamental Diagram using MISSION

We tried to generate a fundamental diagram, which is characteristic for the test network, based on extensive measurements carried out by a simulation in MISSION in order to facilitate a comparison of the two systems´ simulations. The result is displayed in Figure 7.3.

We recorded to a measurement file and evaluated the microscopic traffic parameters observed at 17 detector sites in four simulation runs performed with the same test network and different initialization by the random-number generator during a simulation period of 3,600 seconds each. We integrated the recorded values to time intervals of 20, 30, 40 and 50 seconds for the computation of the macroscopic traffic parameters (average speed, local traffic density and local traffic flow) which will be included in the fundamental diagram and entered the corresponding average value into the diagram.

The enormous spread among the average value curve of the resulting fundamental diagram is conspicuous. The diagram´s curve shape differs considerably form the fundamental diagrams found in literature [Lapierre et al. 87]. Urban street traffic is characterized by interactions between several vehicles and accelerations and decelerations in the range of traffic lights, which occur in the marginal areas of the fundamental diagram. For reasons of its complexity, it is very difficult to describe the traffic by means of a standardized diagram. The two local peaks in the range of maximum flow of the generated curve can be explained by the hysteresis mentioned above and the great statistical error.

7.4 Comparative Measurements

For a comparison, based on a certain traffic situation, of the simulation by ***Sapporo*** and the one by MISSION, detectors were defined at the same sites of the section for the network in MISSION. A local measurement of speeds and traffic flow was performed at these sites during a simulation period of 1,200 seconds (20 minutes). The traffic density was deduced from the mathematical context/correlation of the traffic parameters. The traffic flow values were the actual traffic flow of the Hamburg morning traffic. To obtain comparative values for the macroscopic simulation by ***Sapporo***, the individual registered vehicle values were averaged via a

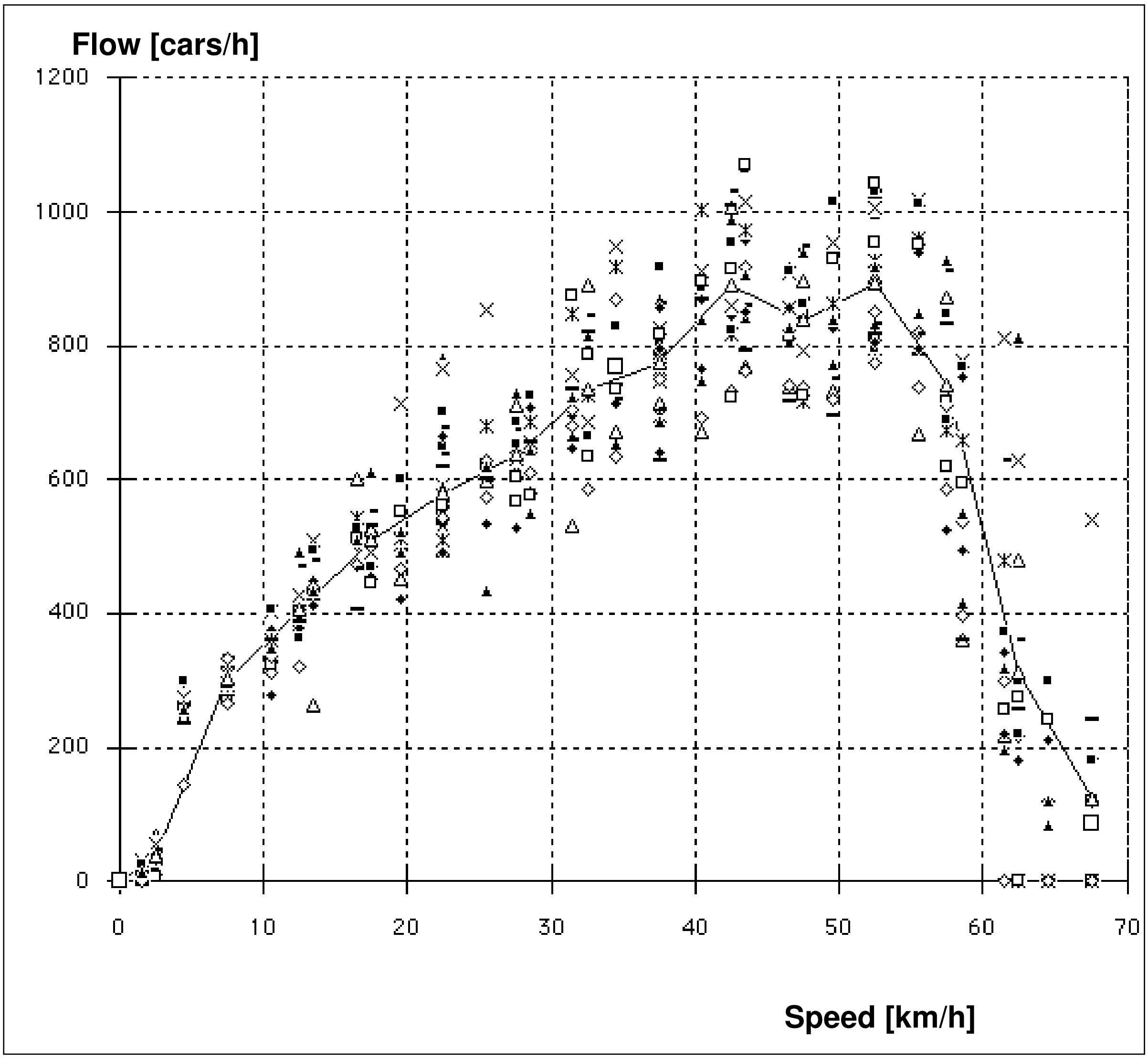


Figure 7.3 Fundamental diagram as average value resulting from single samples produced by MISSION

certain measurement interval. The lengths of the intervals varied between 10, 15, 20, 30 and 45 seconds. Shorter measurement intervals record fewer vehicles and generate therefore larger peak-to-valley values in the measurement curve. Longer intervals result in a more balanced curve, but even out sudden traffic flow changes in the range of the intersections, caused by a phase change in the signal plan.

The traffic was simulated with the same simulation time and the same affluent values in the equivalent traffic network in ***Sapporo***. For the comparison with the curve observed in MISSION a sequence of the density zones at the measurement sites was generated from the corresponding event lists of the section objects in ***Sapporo***.

For simulation in Sapporo the identical traffic was modeled via three different density calculi. The first experiment is based on a fundamental diagram from literature presented by Lapierre in [Lapierre 87]. It reproduces, however, the traffic relations for roads and freeways and does not apply to urban traffic. It shows a considerably higher traffic flow in the range of the optimal density with a higher speed level than the fundamental diagram that was produced by the MISSION simulation. The density calculus of the Lapierre diagram was divided into eight density zones. In the following it will be referred to as *LAPIERRE.* The experiments 2 and 3 are based on the fundamental diagram produced by MISSION. In experiment 2 it is also divided into eight density zones; the calculus is referred to as *MISSION8*. Experiment 3 divides the MISSION diagram into 13 density zones and is therefore referred to as *MISSION13* in the following. Figure 7.4 displays the various density calculi in ***Sapporo***, where the density zone intervals are depicted over the corresponding flow-density-curve of the fundamental diagram.

7.5 Comparison of Results

The simulation in the ***Sapporo*** system aims at a possibly realistic description of the traffic situation on a certain section by the available macroscopic traffic parameters density, speed and flow. There are different ways of how to appraise the quality of the simulation. We considered the correlation of the sequences, which were measured at various sites in the network. The sequences of certain spots on the section will later display the information produced by a simulation for a superordinate signal plan determination component; therefore the comparison of these sequence makes sense. To compare the two curves produced by MISSION detectors and ***Sapporo*** events, the curves were put into a standardized coordinate system. Thus, a visual assessment of the correlation is feasible.Figure 7.5 shows such a comparison by means of two graphics, one of which has a 15-seconds measurement interval in MISSION recording and the other one the density calculus *MISSION8* of the ***Sapporo*** simulation. It displays the correspondence of the parameters flow and speed, respectively, where the two graphics display results observed at two different detectors. The sequence which resulted from the MISSION simulation is depicted as a polyline between the measurement values averaged in 15-seconds intervals, while the rectangles represent the intervals of the qualitative density zones of the ***Sapporo*** simulation. This kind of displays were generated for all detector sites and all three traffic parameters and are contained in an internal test report [Dalchow 91].

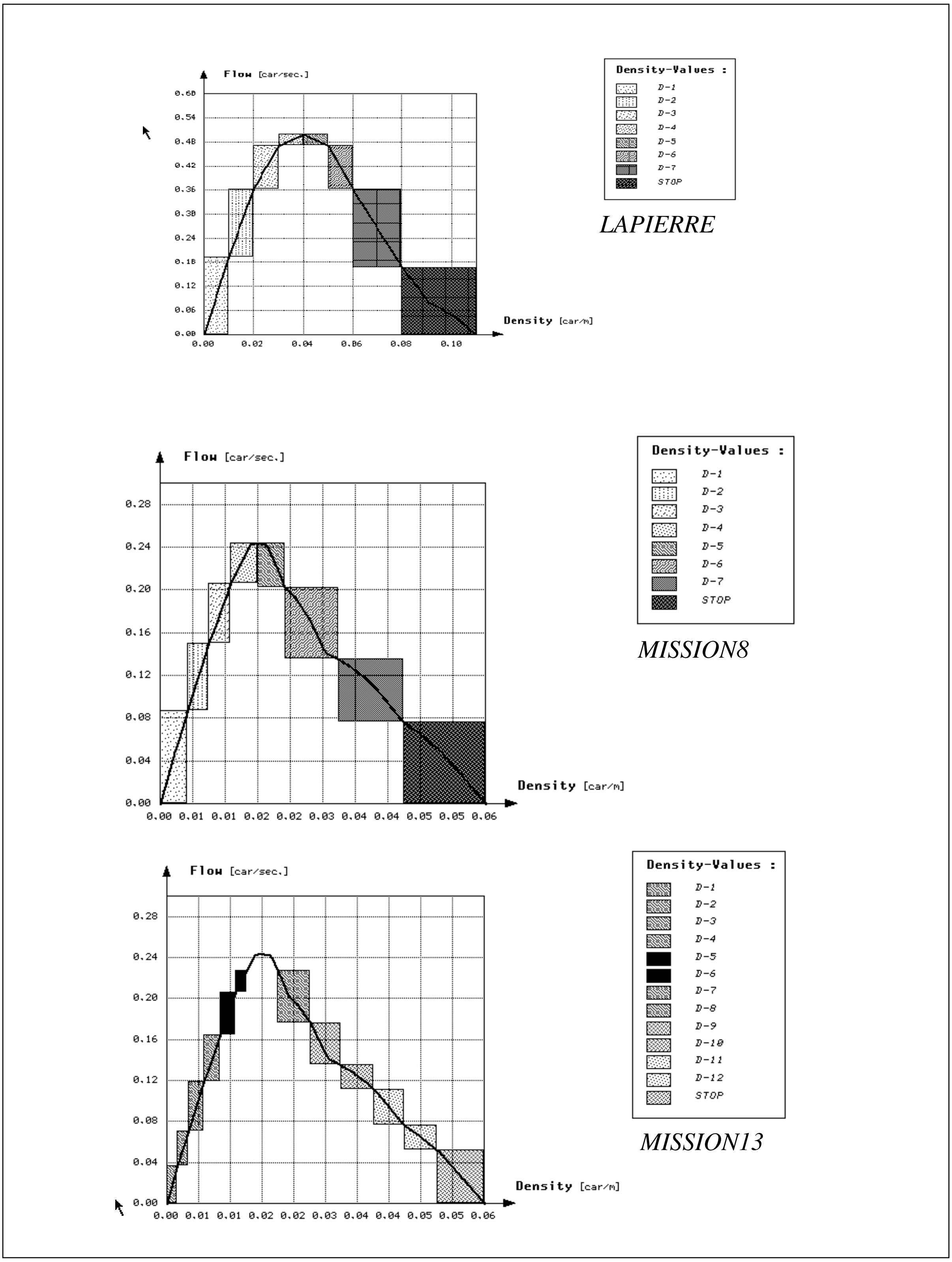


Figure 7.4 Density zone distributions of the three different calculi *LAPIERRE*, *MISSION8* and *MISSION13*, employed in ***Sapporo***.

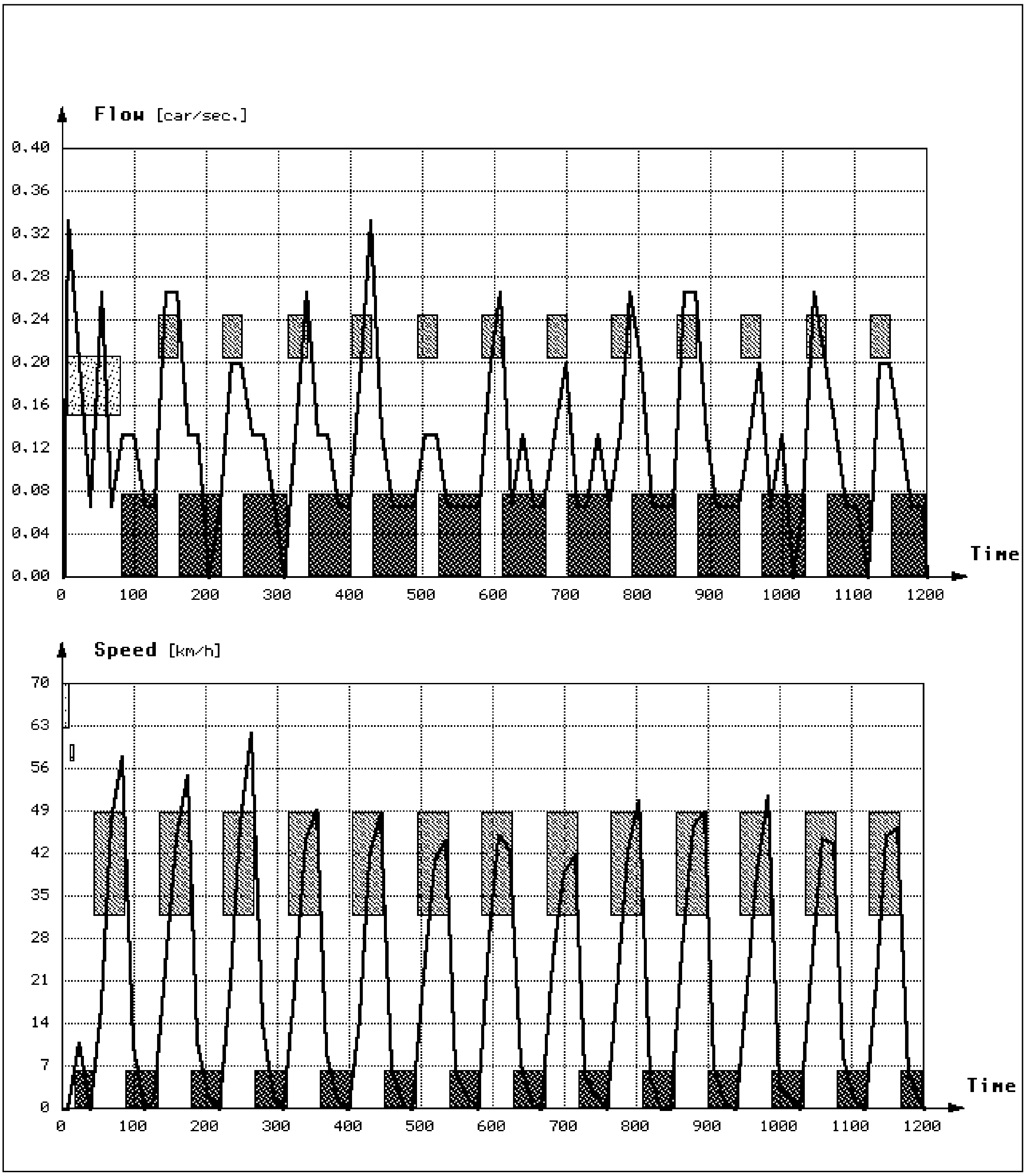


Figure 7.5 Comparison of the simulation results produced by to measurement sites in MISSION and ***Sapporo***

For the numerical assessment of the correlation of the two curves a measure was selected which assesses the percentage deviation of the density zones produced by ***Sapporo*** from the measurement values produced by MISSION for the different traffic parameters during the entire simulation period. For this purpose the amounts of the differences between the MISSION polyline and the average values of the ***Sapporo*** density zones were integrated over the entire simulation period first and then divided by the area below the MISSION polyline. The complete test result, which includes only five detector measurements, is shown inFigure 7.6 in a table. The various measurement interval lengths of the MISSION simulation can be told from

LAPIERRE

Detektor		10 Sekunden	15 Sekunden	20 Sekunden	30 Sekunden	45 Sekunden	Gesamt
212	F	67,8	69,3	72,2	84,7	80,9	75,0
LINK-56 (100)	D	46,3	46,2	50,5	63,5	55,6	52,4
100m vor LSA	G	228,0	185,1	192,6	168,7	186,7	192,2
1220	F	40,7	40,3	38,3	37,3	33,9	38,1
LINK-14 (387)	D	63,1	60,7	54,5	57,6	45,4	56,3
20m vor LSA	G	195,8	168,9	112,1	113,7	78,0	133,7
2220	F	56,6	56,8	56,8	62,4	53,8	57,3
LINK-23 (20)	D	60,8	57,9	56,3	64,3	53,3	58,5
20m nach LSA	G	185,2	136,6	103,7	117,4	46,0	117,8
3220	F	50,1	49,7	51,3	59,0	66,7	55,4
LINK-20 (180)	D	46,5	50,0	58,7	109,8	112,3	75,5
20m vor LSA	G	54,0	42,5	43,4	55,5	66,9	52,5
4220	F	36,6	36,8	37,6	39,5	41,5	38,4
LINK-11 (20)	D	10,7	8,8	12,2	30,0	26,2	17,6
20m nach LSA	G	158,9	170,0	118,3	107,9	150,2	141,1
Gesamt	F	50,4	50,6	51,2	56,6	55,4	**52,8**
	D	45,5	44,7	46,4	65,0	58,6	**52,0**
	G	164,4	140,6	114,0	112,6	105,6	**127,4**

MISSION8

Detektor		10 Sekunden	15 Sekunden	20 Sekunden	30 Sekunden	45 Sekunden	Gesamt
212	F	29,8	36,5	38,3	54,2	50,8	41,9
LINK-56 (100)	D	40,6	39,0	39,3	29,0	28,9	35,4
100m vor LSA	G	96,6	82,4	89,6	96,4	102,2	93,4
1220	F	65,8	65,2	62,7	64,4	56,3	62,9
LINK-14 (387)	D	79,5	78,9	75,3	75,8	69,0	75,7
20m vor LSA	G	170,0	145,4	93,6	95,1	62,4	113,3
2220	F	81,4	81,6	79,9	76,1	76,0	79,0
LINK-23 (20)	D	94,4	93,6	92,9	89,9	89,5	92,1
20m nach LSA	G	250,3	217,9	199,8	173,0	193,5	206,9
3220	F	30,2	30,4	30,4	30,8	31,0	30,6
LINK-20 (180)	D	36,2	36,6	33,2	39,5	46,0	38,3
20m vor LSA	G	20,1	18,8	26,4	39,2	50,0	30,9
4220	F	36,4	31,4	27,2	27,9	27,5	30,1
LINK-11 (20)	D	39,1	37,8	37,1	37,5	36,1	37,5
20m nach LSA	G	131,7	141,7	95,4	86,1	124,0	115,8
Gesamt	F	48,7	49,0	47,7	50,7	48,3	**48,9**
	D	58,0	57,2	55,6	54,3	53,9	**55,8**
	G	133,7	121,2	101,0	98,0	106,4	**112,1**

MISSION13

Detektor		10 Sekunden	15 Sekunden	20 Sekunden	30 Sekunden	45 Sekunden	Gesamt
212	F	40,1	42,6	44,7	57,4	53,1	47,6
LINK-56 (100)	D	38,0	35,9	37,0	28,8	27,8	33,5
100m vor LSA	G	81,7	70,9	77,8	85,9	89,7	81,2
1220	F	54,5	54,7	56,4	65,0	67,1	59,5
LINK-14 (387)	D	62,4	67,9	74,7	86,4	91,6	76,6
20m vor LSA	G	156,4	131,8	84,2	83,3	58,1	102,8
2220	F	61,1	59,4	59,1	72,8	65,6	63,6
LINK-23 (20)	D	65,8	63,3	57,5	68,4	60,7	63,1
20m nach LSA	G	150,0	107,3	78,5	90,5	27,9	90,8
3220	F	39,3	39,7	39,4	39,3	39,5	39,4
LINK-20 (180)	D	31,1	31,9	28,6	33,6	45,8	34,2
20m vor LSA	G	19,7	24,2	30,6	36,4	35,2	29,2
4220	F	37,1	36,8	36,2	36,3	36,1	36,5
LINK-11 (20)	D	48,2	48,0	47,5	47,7	46,6	47,6
20m nach LSA	G	142,0	152,3	104,1	94,3	133,9	125,3
Gesamt	F	46,4	46,6	47,2	54,2	52,3	**49,3**
	D	49,1	49,4	49,1	53,0	54,5	**51,0**
	G	110,0	97,3	75,0	78,1	69,0	**85,9**

Figure 7.6 Tables showing the test results produced by various density calculi

the columns and the various detector sites with the three traffic parameters flow (F), Density (D) and speed (G) can be told from the lines for all three ***Sapporo*** density calculi.

The first table for the density calculus *LAPIERRE* shows an incorrect simulation of the traffic flow values at detector 212, which is situated approx. 110 yards (100 meters) in front of the intersection, of 67.8 percent, if the measurement results from MISSION are averaged intervals of ten seconds at this detector. This means that Sapporo simulates a traffic flow, whose average value differs from the “actual” flow value of the MISSION simulation by 67.8 percent at this detector during the entire simulation period. The average values of the three traffic parameters are told at the end of each column of the three tables observed by the five detectors. The average values of the three traffic parameters during all time intervals observed at each detector are displayed at the end of the lines. The marked numbers in the right bottom corner of each table are the arithmetic average values of all table elements of the corresponding traffic parameter.

7.6 Assessment of Results

The high percentages of the differences between the two simulation results in the tables (Figure 7.6) and their obvious considerable spread result from a systematic error in the comparison of the corresponding underlying curves of ***Sapporo*** and MISSION. This is definitely due to the short measurement intervals in MISSION, which register only few, sometimes no vehicles during low traffic volumes on the one hand, on the other hand the already mentioned large spread of the fundamental diagram can be blamed. Moreover, only the average value of the density zone intervals produced by ***Sapporo*** were taken into account for the comparison of the curves. Therefore, the values listed in the tables may be only qualitatively interpreted for a comparison of the various tested density calculi.

The correlation of the test results produced by the considerably different density calculi *LAPIERRE* and *MISSION8* is also due to the enormous spread of the observed traffic parameters around the corresponding average value of the fundamental diagram. Here, the description of the traffic behavior by the fundamental diagrams and their distribution into eight density zones is simply not accurate enough.

A slightly more accurate correlation of the test values in the density calculus *MISSION13* with 13 different density zones becomes obvious in particular with regard to speed. A finer division of the density zone values thus results into an enhanced simulation accuracy.

At some measurement sites in front of traffic lights a minor time shift of 3 to 8 seconds between the two sequences was observed. This only becomes obvious, however, when looking at the two comparative graphics displayed in Figure 7.5. This may have contributed to the increased deviation in the table, as the shift is not taken into account there. This shift could be caused by the inaccuracy of the fundamental diagram mentioned above. Consequently too high or too low limit speeds between the density zones will be modeled and thus the density zones occur too early or too late at the measurement sites. It could be also caused by the neglect of the deceleration and acceleration behavior at intersections in front and after traffic lights, as the ***Sapporo*** traffic guidance system does not model these events. It is not possible to conclude the cause for this phenomenon from the available measurements. It would require further special measurements to do so.

Regarding the test results from the comparison with the simulation system MISSION on the basis of the generated graphics, a realistic reproduction of the actual traffic relations can be stated. The SAPPORO simulator proves to be a functional tool with good prospects suitable for a qualitative forecast of traffic situations in road traffic. The accuracy of the results depends first and foremost on the accuracy of the underlying fundamental diagram. The smoother the traffic flow in the modeled network, the smaller the spread around the average value in the describing fundamental diagram and the more accurate the simulation by the simulator presented in this paper. Moreover, the quality of the simulation can be enhanced by modeling more density zones. The resulting improvement, however, is relatively small. If non-negligeable time shifts still occur on intersection during the simulation with the optimum fundamental diagram selected, a detailed description and modeling of the acceleration and deceleration events in the range of the traffic lights must be examined.

8 Future Extensions

8.1 Extension of Simulation Model

There are two development trends for the improvement of the macroscopic qualitative model, which should be pursued in parallel:

- the introduction of new classes of simulation objects,
- the improvement of the qualitative model via the individual simulation objects.

In an extension of the object system additional elements of the traffic network are modeled and additional communication links are generated between the simulation objects, which help to explain changes in traffic flow which could not be deduced in the model up to now.

New simulation objects and links are provided by:

- sources and sinks on a section (affluents from car parks etc.)
- dedicated turning lanes on intersections for the distribution of the traffic stream

- accident objects to describe flow changes caused by an accident or a construction site
- direct links between the lanes to describe lane changing (modeling of the driver behavior)

The most important basis for a qualitative description of traffic flow on the lanes by density zones is an as accurate as possible representation of the quantitative relations between the macroscopic traffic parameters in a fundamental diagram. The fundamental diagram is determined by extensive measurements. Then, a suitable division of the fundamental diagram into qualitative density values must be determined. While new density zone distributions have been calculated to describe the traffic flow on the entire length of a lane up to now under the same conditions, an improvement of the model seems to be appropriate in particular for the area immediately in front of the intersections. The driver behavior would have to be described by further conditions which take e.g. the deceleration and the coming to a stop in front of a red traffic light into account.

8.2 Rule System

Modifications of the simulation model can be implemented more readily and faster in a rule-based implementation of the qualitative simulator as in the present functional implementation of the simulator prototype. By the introduction of a rule system and a rule interpreter, the “density calculus” could be replaced by a set of rules describing the lists and tables, the movements of the density zones, their generation and propagation in general. For an improvement of the model the general set of rules would have to be extended only by further rules. Apart from the general rules each simulation object is assigned a set of rules by means of which a new traffic state can be deduced in this simulation object. So, the new density zone distribution would have to be determined via object-specific rules for a link object, while modifications of the density values at the transition from an intersection to the lanes could be taken into account in own rules for the deduction of a new traffic state.

In the framework of the development of a component for the signal plan design (lit) a rule system has been implemented which generates dependent on the intersection type a small set of rules allowing to deduce consistent signal plans for the intersection. The used rule interpreter manages the used rules as objects. According to this approach particular sets of rules could be generated for the simulation objects. To deduce a new state in an object, only the small object-specific set of rules and the few general rules of the “density calculus” would be used, while the use of a powerful rule language like OPS5 (lit), for instance, would have to take into account the rules of all objects of a network to deduce new states.

8.3 Real-time Operation

To facilitate the on-line use of the simulation system to support decisions in the traffic guidances system SAPPORO, it must be considered whether the object-oriented implementation is to be performed in CommonLISP or in another object-oriented programming language. While CommonLISP was thought to be too slow for real-time systems and was characterized by unpredictable long response times (because of garbage collection), new implementations facilitate the use in real-time systems by an improved memory management, the use of macros etc. (lit). Parts of the control systems of the Space Shuttle and of Biospher2, for example, are implemented in CommonLISP.

When using another object-oriented programming language allowing a more efficient implementation of the simulation system, take care that the language uses the genealogy concept and the concept of generic functions. The Objective C and C++ languages are possible alternatives (lit). If the object-oriented rule system mentioned above is used to deduce the state transitions in the individual simulation objects, it can be readily transferred, as the rules are stored as objects and managed correspondingly by a rule interpreter.

9 Actor-Based Architectures

9.1 Overview

The following sections intend to provide a comprehensive synopsis of the existing Actor languages. As we must refrain from a detailed description of the individual approaches within the framework of this paper, the explanation of the individual characteristics includes bibliographical references. Section 9.2.8 explains the criteria the comparison is based on. They play an important part concerning the delimitation of the Actor language that has been developed in the course of this paper.

Actor languages have been developed for the most different purposes - to study communications-based object parallelism, for use in the context of animation and graphical representations of processes and as a mere expansion to the traditional object-based concept of elements of parallelism. Figure 9.1 tries to classify the different languages, which will be explained in the subsequent chapters, according to their field of application and the implementation language represented by a genealogical graph.

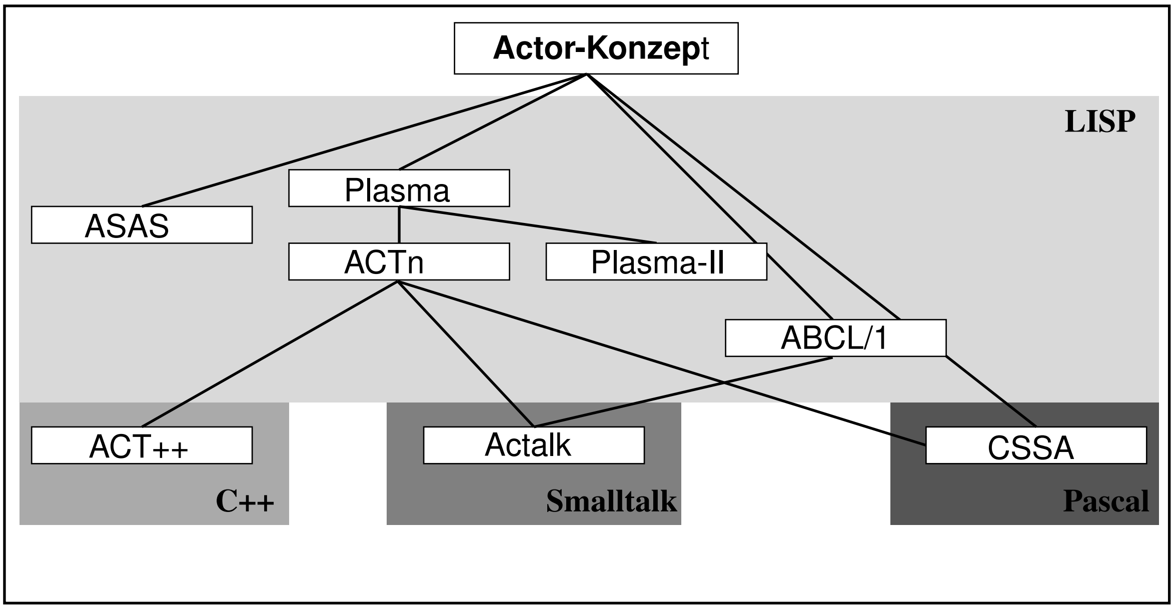


Figure 9.1 A genealogy of actor languages. The corresponding implementation languages are also shown.

Up to now none of the Actor languages has had an impact on software engineering beyond the scope of the laboratory. This is partially due to the lack of dedicated hardware and partially to the application-specific focus of the individual languages. Since none of the implementations disposes of the generality and purity of the computational model presented by [Agha 86], the existing variety of language concepts becomes comprehensible.

Basic criteria concerning an Actor language according to [Kafura & Lee 90]:

- active objects
- asynchronous data transfer
- dynamic creation of Actors
- dynamic reconfiguration of the Actor topology
- the become concept to specify substitution behavior

9.2 Actor Languages

9.2.1 PLASMA

9.2.1.1 Background Information

The Actor language PLASMA was developed at the MIT [Hewitt 77] as a medium to study the semantics of massage passing in Actor systems. An acronym, PLASMA stands for PLANNER-like System Modelled on Actors. It is to provide the tools required for the interactive creation of scenarios, scripts and justifications. The Message Passing Semantics Group around Hewitt wanted to develop a mathematical computational model capable of representing any discrete behavior that is physically feasible. In a first approximation the deduced axioms [Clinger 81] produced the language PLASMA; PLASMA was replaced by the ATCn languages later on. A research team in Toulouse [Lapalme & Sallé 89] developed the PLASMA concept into PLASMA II, which formed the basis for examinations of parallel message-passing-based architectures and applications for modelling and simulating discrete systems [Senteni et al 89] [Senteni et al 90] [Giroux et al 90].

9.2.1.2 Features

PLASMA is a language based on the lambda calculus. It substitutes any function call *f (arg1, arg2,...)* by sending a message *m* to an Actor *A*. According to the original Actor concept, PLASMA is not based on classification and genealogy, but rather on the principle of delegation. This results into a comparative expressive power. Basic objects and Actors, respectively, are defined as prototypes, which can then form clones. The deduced Actors (or clones) have to be specified by their difference between them and the corresponding prototype. In contrast to systems based on classification, a prototype is a “live” form of an object. If a PLASMA Actor receives a message it does not understand in its present state, it “delegates” it to its prototype, which will in turn invoke processing or further delegation. Filters will analyze incoming messages, integrate expected or arbitrary values to local variables and transmit a variable number of parameters. The use of lexical closures facilitates the definition of further Actors in the body of an Actor. Apart from serial Actors, which are of a persistent nature, PLASMA II disposes of “pure” Actors, which will be created when a corresponding message is sent and removed again after the script has been processed.

9.2.1.3 Implementation

In the beginning, PLASMA was a direct extension to Lisp, whose syntax elements are constructed by macros. While Lisp remained a basic implementation language, PLASMA programs started to be compiled in Lisp. The parallel nature of the Actors was reproduced by the introduction of an interpreter to a virtual machine (LILA) on Lisp level. The interpreter is responsible for processing the individual Actor scripts. Thus, Actor scripts have an atomic nature, i.e. an interruption is only feasible after a message has been completely processed. A particular system time Actor can assign a fixed date to the messages. This facilitates the use of PLASMA in the field of simulation [Senteni et al 89]. An expansion of the virtual machine towards a distributed implementation (SMART2), where communication is carried out via UNIX BSD 4.2 *sockets*, is being developed [Lapalme & Sallé 89].

9.2.1.4 Criticism

The slightly confusing syntax of PLASMA renders the language complicated despite its integration to Lisp. The basic principle of delegation is not directly disclosed by the syntax, it rather shows the explorative nature of the language. PLASMA has become useful for the implementation of more simple systems by the introduction of new system Actors (e.g. the time Actor), the integration to graphical interfaces and debugging interfaces. The aspect of research concerning the different concepts of discrete event simulation has been of prime concern.

9.2.2 ACTn[1]

9.2.2.1 Background Information

The language drafts ACT1 [Hewitt et al 79], ACT2 [Theriault 83] and ACT3 [Agha 86], which have been drawn up at the MIT between 1981 and 1986, were the basis for many of the Actor dialects explained in this paper. These dialects have been basically developed out of the specification language SAL (Simple Actor Language) used by Hewitt [Hewitt 77]. The ACTn languages never gained importance in practical life, but were exclusively used to examine to what extent *message-passing*-based languages were suitable for massively parallel computer architectures and for the construction of a computational model for the parallel computation of distributed systems [Hewitt & de Jong 83].

9.2.2.2 Features

As the features of ACTn languages largely correspond to the basic concepts of the Actor model detailed in chapter 3.4, a description can be omitted here.

9.2.2.3 Implementation

All ACTn languages can be regarded as Lisp expansions, where ATCn elements can be expanded to Lisp macros. There are also implementations where Actors are mapped on machine processes [Manning 87], but in general pseudo-parallel Actors suffice, the experimental nature of the language in mind.

1. Because of their similarity ACT1, ACT2 and ACT3 will be treated as one and are referred to as ACTn in the following.

9.2.2.4 Criticism

While Actor languages considerably expanded the minimalistic approach of the ACTn languages in particular in the field communication (cf. chapter 9.2.3), the mainly asynchronous and merely *continuations*-based communication in ACTn forced the user to split his problem into the corresponding Actor units; this impractical splitting was even necessary for more simple solutions. This problem aggravates in the case of the Actor computational model presented by Agha [Agha 85], which neither has local variables nor a direct value assignment. Although this model´s consequence convinces theoretically, current drafts of Actor languages show a series of compromises concerning homogeneity and parallelism, which are predominantly defined by the applicability of such languages.

9.2.3 ABCL/1

9.2.3.1 Background Information

The ABCL/1 language resulted from the development of an environment for the specification and creation of parallel and distributed software and algorithms at the University of Tokyo. ABCL/1 (**A**n Object-**B**ased **C**oncurrent **L**anguage) is the basis for research on suitable theoretical approaches in the field of reflective computational models, program transformations and formal semantics. The corresponding fields of application are symbolic-numerical processes, multiple particle systems, simulation of discrete systems, new operating system architectures and processes in the field of Distributed Artificial Intelligence (DAI).

9.2.3.2 Features

ABCL/1 is an Actor language, in which an object or an Actor is characterized by its state variables and its behavior script. An Actor can only process one message from its receive buffer at a time (i.e. one control thread), where a state sequence similar to a finite state automaton having the subsequent states is executed.

Table 9.1 The state of an actor in ABCL/1

State of Actor	Meaning
dormant	there is no message
active	a message is being processed
waiting	waiting for the answer to a query/poll

A message is only accepted, if a corresponding input pattern is found in its behavior script. It is also possible to send messages to Actors known to the Actor (*acquaintances*). Six different options are available, whereby all communications techniques can be implemented. Table 9.2

Type	Communications mode		Meaning
	normal	express	
past	A <= B	A <<= B	asynchronous
now	A <== B	A <<== B	synchronous
future	A <= B $ C	A <<= B $ C	asynchronous with a return value in container C

Table 9.2 The available communications modes in ABCL/1

shows the three basic options *past*, *now* and *future* with the modes *normal* and *express* and gives a short explanation. By the introduction of the *express*-mode, ABCL/1 allows the implementation of interrupt-controlled principles as required for an OS kernel, for instance [Doi & Kodama 90].

An important aspect concerning dynamically changeable objects and structures is the support of the meta object level, on which the internal mechanisms according to which an object “works” can be accessed and manipulated without restrictions [Takada & Yonezawa 90].

9.2.3.3 Implementation

ABCL/1 is implemented in Common Lisp and accepts Lisp constructs in addition to its own syntax elements. A built-in compiler translates the ABCL/1 program into Lisp source code. Thus, only a Standard-Common-Lisp Compiler is required. The processing of Actors is done in a pseudo-concurrent manner by the execution time system of ABCL/1, while more recent implementations are directly based on *lightweight processes* [REF].

9.2.3.4 Criticism

Like ACTn, ABCL/1 has neither an explicit classification mechanism nor a genealogy mechanism. Generator objects can function as prototype classes, but a genealogy relation is missing for reasons of distribution amongst other things. Contrary to the Actor Model of [Agha 86] the specification of a replacement behavior and thus the provision of new message acceptance patterns was omitted. Despite an enhanced expressive power to model parallelism, ABCL/1 rather shows similarities to Hoare´s [Hoare 78] CSP model. For reasons of generality the language lacks an explicit support of time. While almost all other languages are predominantly used in the laboratory, ABCL/1 is a useful tool to develop parallel object-oriented software. This is even enhanced as the program system is freely distributed.

9.2.4 ASAS

9.2.4.1 Background Information

The Actor/Scriptor Animation System ASAS uses the Actor concept to produce photorealistic animated three-dimensional scenes. The Actor concept is based on the metaphor of actors on a stage interacting according to a script. The ASAS specification language [Reynolds 82] developed by the Architecture Machine Group at the MIT aimed at replacing the complex and time-consuming creation of animated scenes by mathematical transformations on matrices with a more abstract and a description based on the semantics of the movement.

9.2.4.2 Features

ASAS describes all geometric objects participating in a scene as Actors, which can perform a temporal sequence of actions such as movements, color changes and rotations according to a script. It differentiates between operations which correspond to standard Lisp functions and real Actors. While an operation is used to implement the basic behavior or create an Actor, a ASAS Actor embodies a geometrical object including one or more operations. An Actor will never activate itself, nor does it represent an own process. It is activated by a time-controlled animation system, which activates all existing Actors at fixed time increment known as frames according to the *round robin* principle. In this context the Actor is responsible for the appropriate setting of local parameters of the represented object, for instance position, angle etc. Moreover, Actors can transmit messages to other Actors via a *message passing* interface. These messages are stored to a receive buffer, which operates according to FIFO, in the receiving Actor. Only one message can be extracted and processed per activation.

9.2.4.3 Implementation

Like other Actor languages developed at the MIT ASAS has been conceived as an expansion of Common Lisp, so that all Lisp constructs can be transparently used in ASAS. This allows the integration of the system to the three-dimensional modeling and animation software Leonardo by SYMBOLICS at a later point of time. This software is responsible for output or video clip generation, respectively.

9.2.4.4 Criticism

ASAS has been absolutely customized to the requirements of photorealistic video clips. It has a fixed frame time and numerous geometrical, movement, shading and color operators. An interesting aspect of ASAS is that it is based on the Actor model which allows a comprehensible and semantically expressive specification of dynamic processes. For this reason, including the explicit handling of time, ASAS gives the use of an Actor-based modeling and simulation system a decisive impetus and stimulus.

9.2.5 CSSA

9.2.5.1 Background Information

The CSSA (**C**omputing **S**ystem for **S**ocieties of **A**gents) language developed within the INCAS project [Nehmer et al 87] at the University of Kaiserslautern is used as a specification tool for applications based on asynchronous parallel and distributed processes. The development of distributed algorithms for the most different fields of application [Mattern 89] and the demand for transparency in programming with an almost unlimited scalability of the underlying computer architecture were the foundation for CSSA based on the message-oriented activation of atomic operations.

9.2.5.2 Features

In CSSA, an Actor is referred to as agent whose behavior is defined by a script. Scripts divide into facets, which correspond to the behavior in the original Actor model. These facets divide again into operations, which implement the actual behavior. At every point of time an agent is only in one active facet so that only one operation specified in this facet can be activated in case of message reception and a successful comparison of patterns. Both input variables for the agent and message parameters for the individual operations can be checked for compliance with certain restrictions before being accepted by the agent or the operation. The replacement behavior of an agent is invoked within an operation by specifying the name of the facet responsible for the next message.

In CSSA, each agent has only one control thread. Thus, parallel activation of an agent´s facets is avoided. Messages are transmitted asynchronously and stored to a receive buffer. Contrary to other implementations, messages can pass each other, provided that transmission is guaranteed. A message may contain a responding address. This allows the use of any higher communications protocol.

9.2.5.3 Implementation

CSSA is a PASCAL-oriented structured programming language, which has also expansions for script definition and communication in addition to the usual data and control constructs. A compiler translates a CSSA program into a C program, which is then mapped onto an executable program including the corresponding libraries via a standard compiler. The agents of this program can be located on distributed machines. In this context each agent represents an own UNIX process, while communication is performed via the UNIX sockets.

The weaknesses of the UNIX process concept (process switch-over time, absolute number of processes), which restricts the number of agents of a machine to less than 20 (< 20) each, are met with a reimplementation in *lightweight processes*.

9.2.5.4 Criticism

CSSA translates the Actor concept into a procedure-oriented language, which has the known restrictions (e.g. no dynamic reconfiguration of an agent or its behavior, strictly type-coded message arguments) and no genealogy concept. Against the background of a verifiable distributed programming concept such design decisions seem to be appropriate. However, they reduce the expressive power for flexible modeling of complex systems. Moreover, the relatively small number of possible agents restricts the practical use of CSSA.

9.2.6 Actalk, Actra

9.2.6.1 Background Information

Originally conceived for the demonstration of the flexibility of Smalltalk–80, the object-oriented language *per se*, Actalk is a demonstration for the expansion of an existing object-oriented language by Actor characteristics and the classification of Actor languages [Briot 88] [Briot 89] [Briot 89b]. In contrast, Actra is an approach to translate Actor characteristics to a multiprocessor system by means of a distributed virtual Smalltalk machine [Thomas et al 89]. Both languages maintained full compatibility to Smalltalk–80.

9.2.6.2 Features

Actalk uses the means provided by Smalltalk-80 (classes, single inheritance, methods) to put the language elements explained by Agha in [Agha 86] into effect. Moreover, the classes `Process`, `Semaphore` and `Scheduler` map the dynamical aspects onto the existing process level, aiming at the creation of an Actalk kernel, which can be gradually expanded by various communications and scheduling principles. This also allows synchronous communication and *futures* besides the asynchronous data transfer of the original Actor concept.

Apart form the traditional Smalltalk objects Actra introduces another class referred to as `Actor`, where the entire interaction with the underlying real time kernel “Harmony” is implemented. Therefore, the activation of an Actor is directly mapped onto a task of the real time system. Additionally, Actors can reside on different processes, where a distributed message system is responsible for intercommunication.

9.2.6.3 Implementation

The implementation of the asynchronous communication in Actalk only requires a minimal adaptation of the virtual machine of Smalltalk-80. The standard error handling routine `doesNotUnderstand`, which is sent to the super class `Object` on reception of a message having an unknown selector, must be changed so that the selector is first check for identity with an already defined behavior script. If the selector can be accepted as a message in the current behavior state of the Actor, the appropriate script will be split off in form of a Smalltalk block context, i.e. asynchronous processing of the behavior is invoked.

In Actra the asynchronous communication is analogously integrated to the actually synchronous language Smalltalk. The Actors, however, are not managed by the Smalltalk scheduler, but by the "Harmony" kernel, as they are instances of a real time task. The distributed nature of Actra, which is based on a dynamical and untypified language such as Smalltalk, requires a garbage collection in distributed systems. This garbage collection must be able to resolve inter-processor references.

9.2.6.4 Criticism

While Actalk can be still used as the basis for studying and classifying the Actor languages, more efficient implementations for more complex tasks require a large-scale adaptation of the virtual machine and the primitives of Smalltalk, which is exactly what Actalk wanted to avoid at all costs (see also [Yokote & Tokoro 86]).

In Actra the design focuses less on the development of a pure Actor language than on the expansion of Smalltalk by real time features and distributed characteristics. The problems of object mobility, of *load balancing* during method calls and the distributed *garbage collection* cannot be neglected. The smallest unit of parallelism comprises a method call in both languages, i.e. a disruption within a method body is impossible. While Actra permits only one active method per Actor (= 1 task), the use of the block context allows several tasks in Actra, where a mutual interlock of the unique instance variables does not occur.

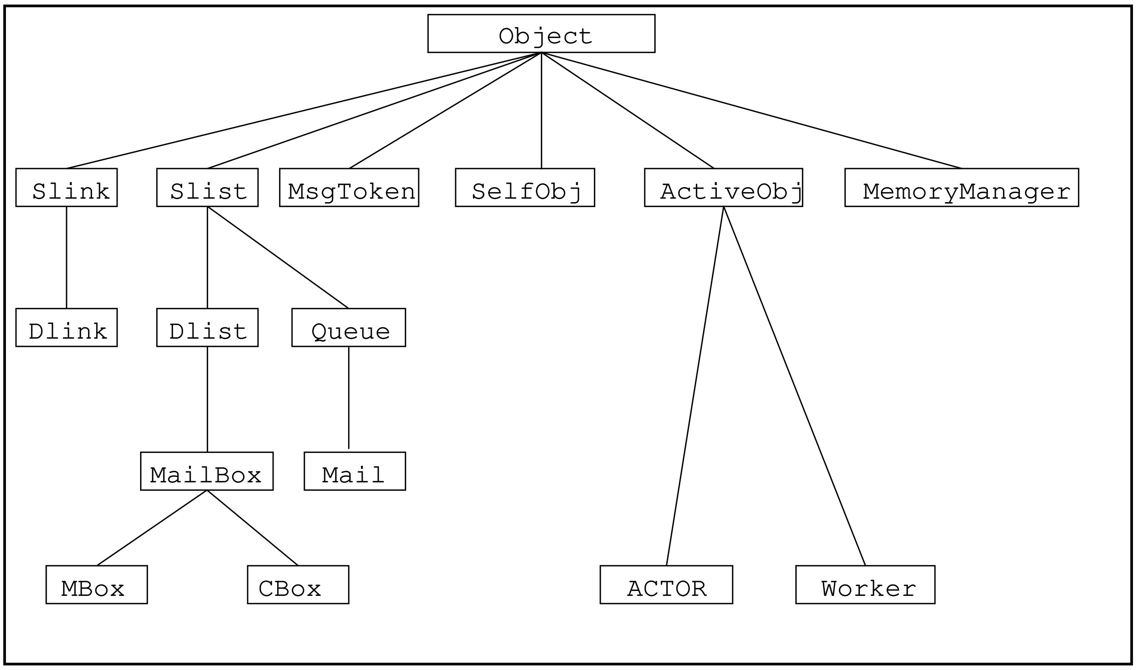


Figure 9.2 Hierarchy of classes in ACT++. An entire Actor consists of the components `MBox`, `CBox` and `ACTOR`, where in the current version only single inheritance is used in analogy to Smalltalk. The class `Worker` corresponds to the task Actor in Agha.

9.2.7 ACT++

9.2.7.1 Background Information

The numerous analogous characteristics of distributed real time systems and parallel object-oriented programming caused a research team at the Virginia Polytechnic Institute [Kafura 88] [Kafura & Lee 89] [Kafura & Lee 90] to take a suitable Actor concept as the basis for an object-oriented distributed real time system. Research resulted into the development of the Actor language ACT++ based on C++ [Stroustrup 86].

9.2.7.2 Features

As ACT++ is an expansion to C++, all constructs typical for C++ are available. Contrary to the original Actor concept, ACT++ uses classes and genealogy for modeling and methods for the implementation of types of behavior. An Actor, e.g. is implemented via an own class hierarchy (cf. Figure 9.2). An Actor is created either directly by the instantiation of the class

ACTOR or after a further specialization by a subclass. The granularity of parallelism in ACT++ is situated on the object level, which means that only one method or behavior can be active per active object. As C++ forms the basis, ACT++ does not have a meta level for the description of operations within an Actors. As communication divides into one-way-messages and queries with response, a separate message buffer object is provided for each message type. So, an explicit response address can be specified when sending a message (*forwarding*).

9.2.7.3 Implementation

ACT++ maps an Actor onto three objects (ACTOR, MBox, CBox). Parallelism of the individual Actor behavior is facilitated by *lightweight processes*. This allows a common address space for all Actors. While the current version still uses pure object pointers as Actor addresses, the introduction of logical names is to allow the implementation on a network cluster of workstations.

9.2.7.4 Criticism

ACT++ makes full use of the options provided by C++ such as classes, genealogy, polymorphism, function overlay and a *lightweight process* system. Unfortunately, constructs which allow real time use, a true distributed expansion and an efficient *garbage collection* are missing up to now. Moreover, C++ is a translated and strictly typified language. This results in restrictions concerning the use of untypified message arguments and the dynamical reconfiguration.

9.2.8 Comparison and Assessment

The approaches to Actor languages are compared according to certain characteristics in to present an overview.

Apart form the target field of application the support of characteristics such as genealogy, behavior replacement and the meta level play an important part as regards the expressiveness or the expressive power of a language for modeling.

Granularity denotes the range an abstract concept refers to. According to this, the term *Actor Granularity* denotes the scope of what is regarded as an Actor. So the ACTn family, for instance, regards any instruction as an Actor, while all other languages regard an Actor as an entire object with state variables and behavior instructions. Consequently, such an Actor object is *atomic*, i.e. it cannot be divided into further Actors.

Similar to this, *process granularity* denotes the object boundary, which can be regarded as an individual process in a multiprocess system or as a pseudo-process in a one-process-system.

While the vast majority is based on the implementation language Lisp, where Lisp already provides above all the dynamical features such as garbage collection, reconfiguration and dynamical code generation, CSSA and ACT++ represent serious approaches to combine the Actor model with strictly typified compiled languages. Such an implementation, however, is definitely more complex for the above reasons. Apart from the actual Actor model it requires an appropriate operating system. Correspondingly, the focus of CSSA and ACT++ shifts to the development of distributed systems to examine distributed basic algorithms (CSSA) and the development of parallel object-oriented real time systems, respectively. Precompilers, which translate the source code into C code, have been developed for both languages.

Of all examined languages ABCL/1 provides the decisive advantages of flexibility, expressive power, a development environment and documentation apart from its free availability. The explicit representation of a recursive meta level[1] facilitates on the one hand dynamical changes concerning the behavior of an object and on the other hand the formulation of particular techniques of message management on the meta level. These techniques are transparent for the specification of problems on the standard language level(e.g. the time warp mechanism for distributed simulation [Takada & Yonezawa 90]). Its own syntax and the lack of a replacement behavior as well as genealogy as a basic tool have a negative effect on our type of problem.

9.3 Application of Actors in Sapporo

The application of the actor model in the Sapporo framework is concentrated onto 2 domains:

1. This means that the behavior of an object is defined by its meta object, where the meta object itself is an object, which in turn has a corresponding meta object and so on. This process can be endlessly continued recursively, where after a maximum of two recursions a limit concerning efficiency, memory requirements and expressive power is reached.

	Plasma	ACTn	ABCL/1	ASAS	CSSA	Actalk	ACT++
Field of application	parallel algorithms	parallel and distributed systems	distributed operating systems and simulation	animation	parallel and distributed algorithms	research and teaching	parallel object-oriented real time systems
Basic language	Lisp	Lisp	Lisp	Lisp	own (similar to PASCA)	Smalltalk	C++
Availability	–	–	yes, via FTP	–	–	yes	no
Architecture	pseudo-parallel	pseudo-parallel	pseudo-parallel distributed	pseudo-parallel	parallel and distributed	pseudo-parallel	pseudo-parallel (designed as distributed)
Genealogy	no (delegation)	no	no	no	no	yes, single inheritance	yes
Replacement behavior	no	yes	no	no	yes	yes	yes
Actor granularity	atomic	instruction	atomic	atomic	atomic	atomic	atomic
Process granularity	pseudo	instruction	Actor cluster	pseudo	Actor	pseudo	lightweight process
Meta level	no	no	yes	no	no	yes	yes
Types of communication	1 asynchronous	1 asynchronous	3 past, now, future	1 asynchronous	1 asynchronous	variable	1 asynchronous

Table 9.3 A comparison of the examined Actor languages. As far as the used criteria cannot be derived from the individual descriptions of the languages, they will be explained in Section 9.2.8.

- floating cars
- control entities of the distributed contract network

9.3.1 Floating Cars

Floating cars are car objects which move in a microscopic way through the road network and its traffic situations. They belong therefore to the microscopic part of the traffic model. The reason for introducing such objects are the following:

- collection of individual statistical data relevant to the efficient control like delay times, number of stops, travel time, mean speed, etc.
- sensoring of various data at specific location within a link
- potential connection of the simulated car to real cars equipped with dedicated sensor and transponder hardware.

The actual state of the research on this field for the Sapporo project is not yet finished. The implementation of user interfaces integrated in the Sapporo prototype is currently done. Moreover the tight integration of results obtained from floating car actors is currently under active research.

A first glance of the user interface to a specific floating car actor can be seen in Figure 9.3.

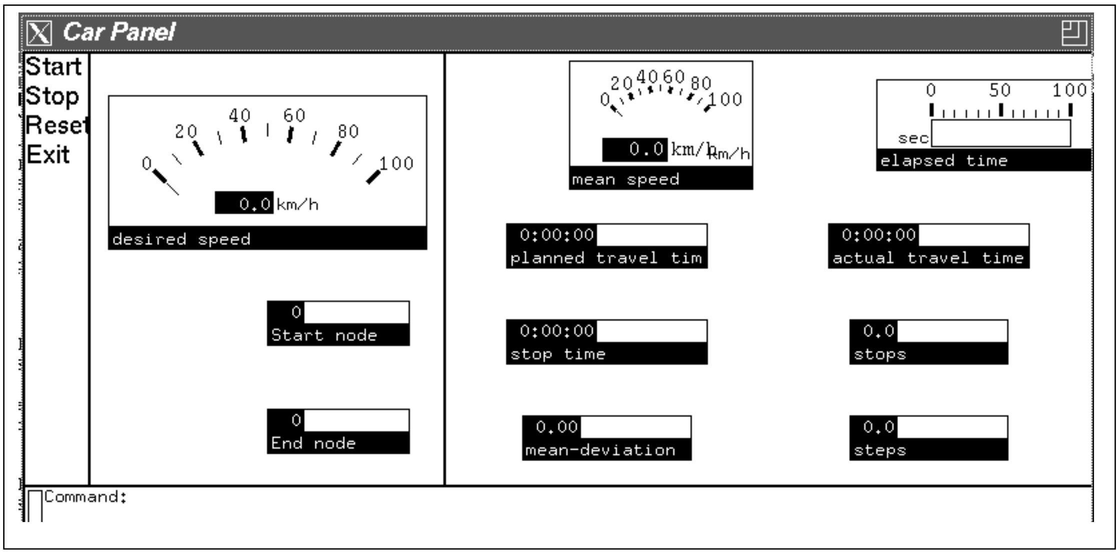


Figure 9.3 The user interface to floating car actors. The user can manipulate the start node and end node of the route to be followed and the desired speed by mouse or keyboard. He can regard on-line the parameters shown on the right pane:
• the accumulated mean speed of the actor's travel so far
• various time values important for optimization and control
• and the number of simulation steps executed.

10 The Intersection Designer

The main problems of the use (and programming, respectively) of rule-based systems are due to the slow runtime and the lack of a clear structure of the rule systems (cf. above). Two possible solutions to this problem are available. A reduction of the rule bases decreases the confusion to a minimum. A consequence resulting from this is that fewer rules must be checked on membership to the group of rules which can fire. This result in a reduction of the runtime.

An alternative solution is the use of rule system customized to the problem (or the problem solution type). This helps to avoid for instance unnecessary overhead which results from the implementation and consequently the management of unused or unnecessary functions. Moreover, a problem-specific internal representation of the data basis can help to find the condition parts which correspond to the data basis elements and simplify the implementation of all elements.

Both approaches have been tried in this paper.

10.1 Implementation of the System Components

All data structures used in our system **IDAR** (**I**ntersection **D**esign **a**nd **R**ule **S**ystem) are implemented as CLOS objects [Keene 89].

10.1.1 Intersection Geometry Objects

One object is instantiated for each intersection which is to be examined. It contains data on the legs of the intersection (called “sections” in the SAPPORO system), the lanes on the legs (known as “lanes” in SAPPORO) and the links between the lanes. Furthermore, it contains a reference to an object holding the data required for the rule system (cf. below) of IDAR.

10.1.2 Objects of the Rule System

For the implementation of the rule system and expert system was used for the configuration (tasks) (KONEX) [König 1990]. This system was adapted to the situation by removing the unnecessary functions and improving the remaining ones. A (restricted) general usability of the system was obtained.

The modeling of rules (cf. 10.1.2.2) was almost completely taken over from the system partially including the structure of the state tree (cf. 10.1.2.1). The classes (or objects), used for configuration, were implemented according to the requests of our system.

Contrary to the complete description of the configurator KONEX ([König 1990]) only the object areas and the structure of the state tree, as used in this context, will be explained in this paper.

10.1.2.1 State Tree

The current state within the configuration (of the configuration process) is stored in a (state) tree. This tree is empty (except for the root state) at the beginning of the configuration.

If a rule triggers, an object is added to the existing configuration and a new state is generated (see Figure 10.2).

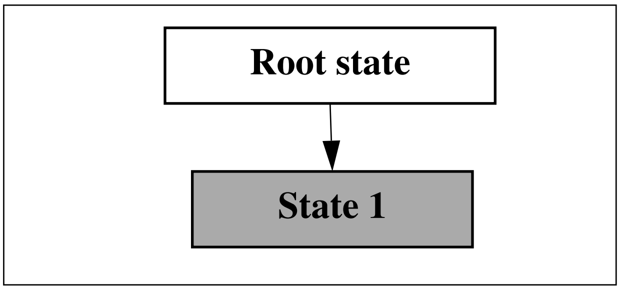


Figure 10.1 The state tree after the first addition of a new object

If the form of the rule allows several expansion options, several consequent states will be created (Figure 10.2). Contrary to KONEX our system does not produce faulty states. (Thus, the removal of faulty state or parts of the state tree is avoided.) This was above all reached by well formulated rules. Moreover, the user cannot add objects to the configuration (excluding a possible source of errors from the very beginning). For the exact formulation of the rules and the structure of the state tree refer to chapter 10.2.1.

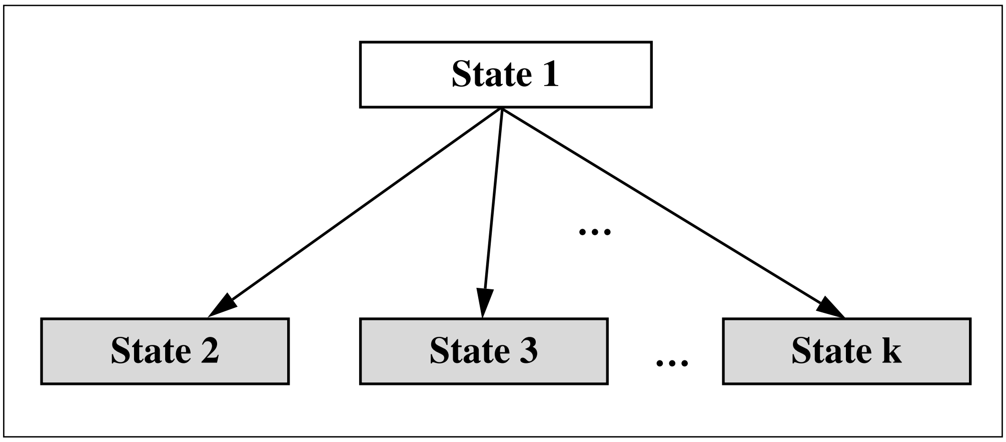


Figure 10.2 The state tree after several additions of a new object

10.1.2.2 Rules

A rule generated by our module consists of the following components:

1) **Name of Rule**
   It clearly identifies the rule.
2) **condition Part**
   It describes the conditions a state has to comply with so that the rule is accepted in the conflict set.
3) **Action Part**
   If the rule triggers, the listed actions will be performed.

The rules are also saved as CLOS objects internally. The above components are the slots of these objects. To enable a faster search of the rules, whose condition parts are fulfilled (in this context they are also referred to as firing rules), the objects, which are referenced in the condition or action part, also contain references to the corresponding rule.

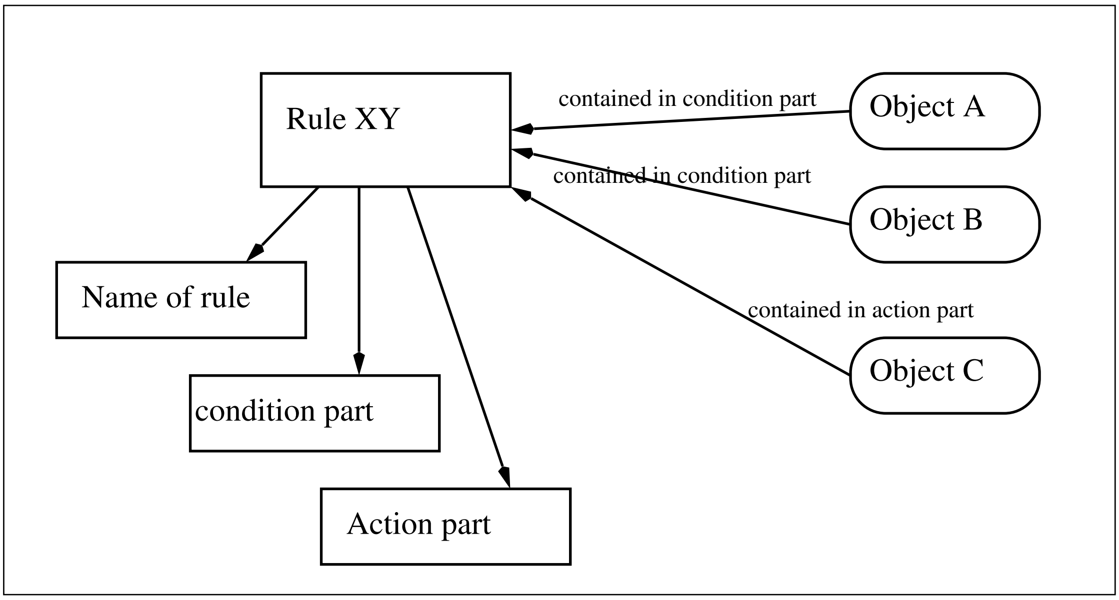


Figure 10.3 Rule structure of IDAR

Although syntactically identical, the condition part and the action part must be interpreted differently as regards semantics. The generated rules have the following components:

1) Functions (**`AND, OR, NOT`**)
2) Class or object identifiers.

Table 10.1 Function components of rules in IDAR.

Function	Parameter	condition part	Action part
AND	Class/object	An object of this class / object must exist in this state.	An object of this class / object is added to this state.
	Function	This function must be fulfilled.	This function must be fulfilled.
OR	Classes/ objects	One of the objects / classes must exist in this state.	One of the objects / classes is added to this state.
	Functions	One of the functions must be fulfilled for this state.	One of the functions must be fulfilled for this state.
NOT	Class/object	An object of this class / object must not exist in this state.	This class / object is locked for this state.
	Function	The function must not be fulfilled for this state.	The function must not be fulfilled for this state.

10.1.2.3 Configuration Objects

The configuration process involves the integration of an object to the presently found state and the generation of a new state. The objects which can be used during the entire configuration, divide into three different groups (or object states) during the current configuration step.

1) Configurable Objects
   This group comprises all objects which can be used in the current configuration step, i.e. they can be added to the current configuration state.
2) Locked Objects
   The group of locked objects contains all objects which must not be used any longer during configuration.
3) System-selected Objects
   All objects added by the triggering of rules to a state count among this group.

During configuration we will use two different class types. The first class type consists of a class which instantiates only one object during the entire configuration process. The integration of this object to the root state starts the configuration. Thus, the conditions required to enable the first rule to trigger, are created.

The classes belonging to the second class type are referred to as “lane type classes” in the following. The class type is defined as follows:
Each enter lane has (explicit or implicit) links to the exit lanes on an intersection. The links are classified according to the exit lane´s location on a leg on the intersection. (From the road user´s point of view on the lane they are classified as left or right turning or as straight ahead link.) A lane´s found links are converted into a lane type for this lane and assigned to the lane. This lane type is also used in the class definitions combined with an individual identifier of the intersection´s leg where the lane is located.

Two classes are defined on the basis of this lane type tupels to be used in the signal plan configuration. The first class definition uses this tuple and the extension “.0”. If an object of this class is added to a state, this lane is in the red phase of the signal plan (i.e. in this state. Therefore, these classes are referred to as “red” lane type classes in the following.) Equivalent in extension (“.1”) and use, a second class is defined which will be referred to as “green” lane type class later on.

10.2 System Modules

10.2.1 Description of the Rules

There are three different rule groups (or rule types) in our system:

The first rule, which is triggered at the start of the inference engine, creates various consequent states on the basis of the empty root state `k` (cf. above), where `k` is the number of green lane type classes (cf. above) which are available at the intersection. Each of the consequent states has exactly one object of one of these lane type classes. This rule is assigned highest priority within the system.

The second rule type (which is referred to as “restriction-rule-…”), does the most important job within the system. When triggered, each rule of this type determines the consequent states, which are obligatory because of the restrictions (cf. below) for class resulting from the intersection´s geometry, for one lane type class each. In other words, it determines which traffic light has the be in the red phase while this one is set to green.

The third rule type (“generic-rule-…”), which was assigned the most inferior priority, triggers only when no other rule of the previous tow rule types can be triggered any more. These rules produce two consequent states on the basis of the current state via adding an object of a lane type class that does not exist in this state (i.e. in this phase) once in red (1st consequent state) and once in green (2nd consequent state).

10.2.2 Rule Generation Module

The reduction of the number of rules was effected by the implementation of a module which instantiates only the rules (or rule objects) relevant to the presently examined intersection. Instead of a lot of general rules only few particular rules will be used.

The TRALI system ([Zozaya 1987]), for instance, continuously holds approx. 200 rules in the memory, of which only a part (e.g. via context control) is ready to be triggered, but considerable speed reductions will be caused by the required context changes. (A real-time application of the TRALI system is, therefore, not useful.)

Our module bases the rule generation on the intersection´s geometry. It is either taken over from the node of a currently loaded network of the SAPPORO system or arbitrarily designed.

The restrictions for each lane on the intersection are determined on the basis of the available links between the lanes within the node or other limitations. These restrictions are entered to a slot of the corresponding lane type in the form of lanes colliding with this lane.

To determine these restrictions we will first examine an “ideal” intersection.

The intersection has an identical number of enter and exit lanes. An identical number of reserved exit and enter lanes are available to each lane type (i.e. left and right turning, straight ahead, …). Thus, the relation between the enter and exit lanes of a lane type is an n:n relation. Figure 10.5 displays an example of such an intersection having four enter legs and three different lane types.

We intend to determine the number of lane types colliding with the individual lane types on the intersection (Figure 10.5). The resulting number of lane types colliding with the individual lane type is minimal with regard to all existing intersections. This means that when calculating the colliding lane types for any lane type in any intersection, at least this number will be obtained.

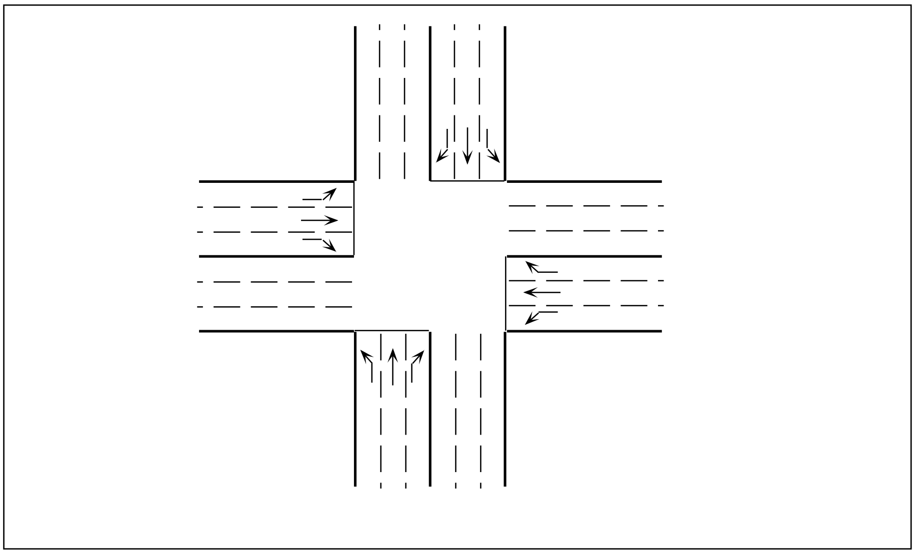


Figure 10.4 An intersection with four legs and 3 types of lanes.

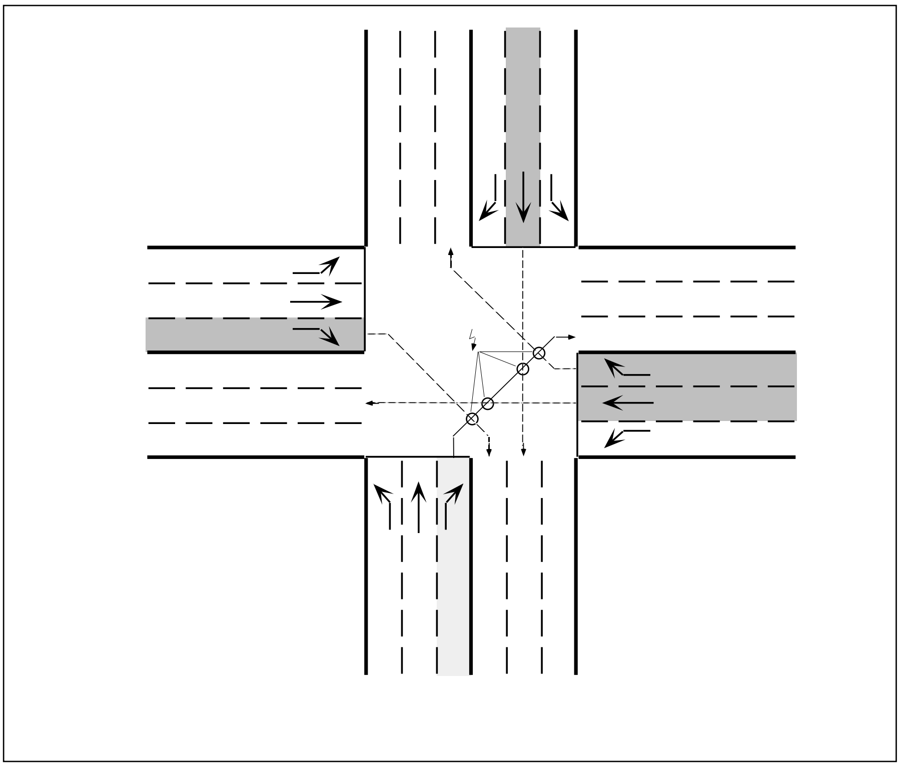


Figure 10.5 Determination of the conflict set for a given lane.

The basis list of colliding lanes, which is made on the basis of the above minimum number, is stored in a list in the rule system object. This list can be modified for each intersection by the system user. (For instance to handle colliding right turning lanes (Figure 10.5) according to the current situation on the intersection.)

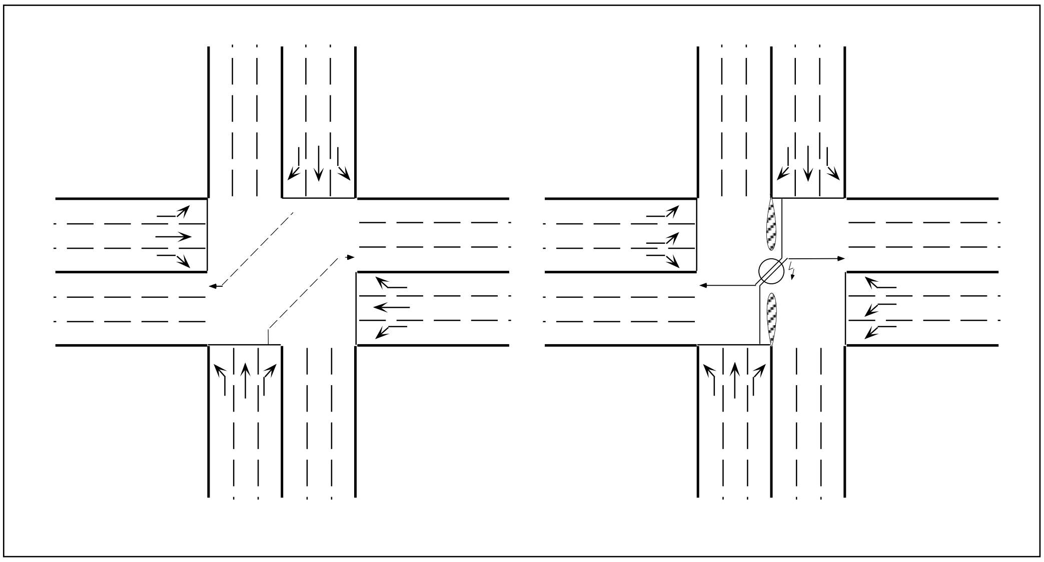


Figure 10.6 The right (or left, respectively) turn conflict set.

On a real intersection lanes colliding with the examined exit lane because of the particular intersection geometry must be added to the (minimum) number. This can happen, when, for instance, more than one lane approach an exit lane (Figure 10.7) (i.e. there are less reserved exit that enter lanes).

To determine the other colliding lanes the environment of the SAPPORO system was used. SAPPORO has connections between the enter and exit lanes. We determine the connections to exit lanes for the enter lane which is to be examined and for which we want to determine the colliding lanes. We search all enter lanes, to which connections exist, for these exit lanes (except for the lane we are presently looking at).

If we combine the resulting number with the minimum number, we will obtain all lane colliding with the examined one. The subsequent example in Figure 10.8 depicts this procedure.

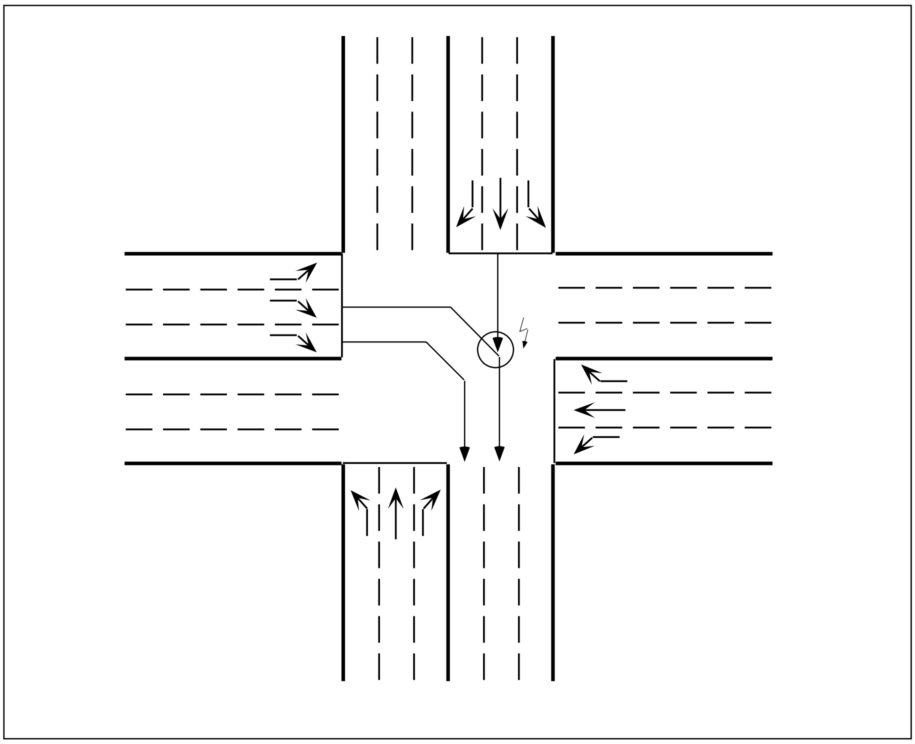


Figure 10.7 The merge of 2 or more incoming lanes into 1 outgoing lane.

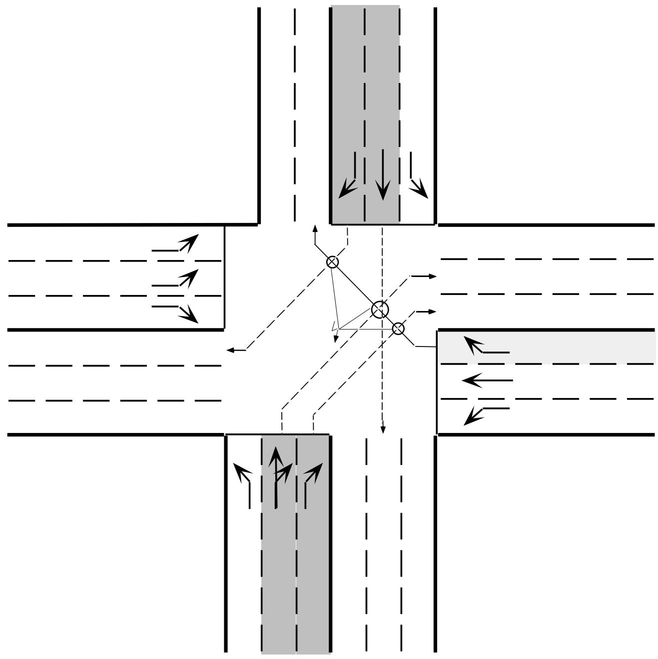
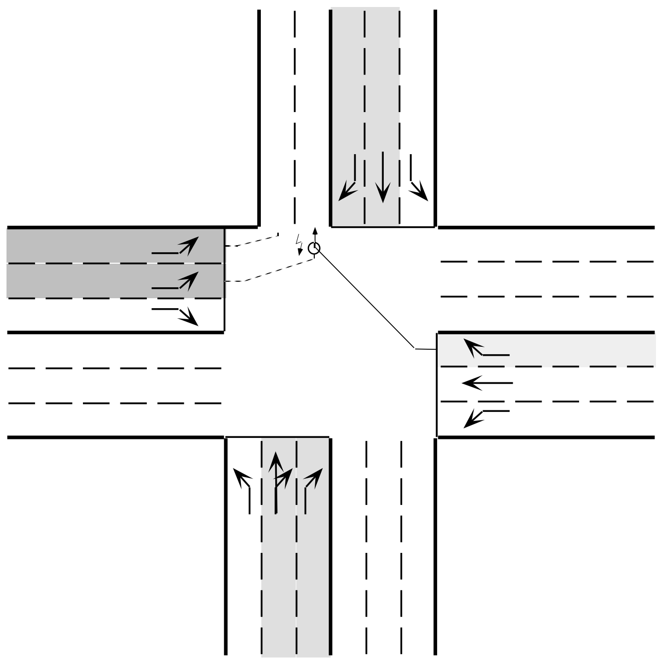
First Step: Determine minimal conflict set for the lane	
Second Step: Determine additional restrictions cause of the particular intersection geometry	

Figure 10.8 Determination of the complete conflict set for one lane.

10.2.3 Inference Engine

The inference engine used in our system is a cycle as detailed in [Saffran 92].

The modification of the data basis always involves the integration of an (additional) object to the current configuration state. This corresponds to an extension of the state tree by a terminal node (or a partial tree). A priority-controlled rule selection strategy is used in the application. (Thus, the transition behavior between the rule groups becomes more clearly structured.)

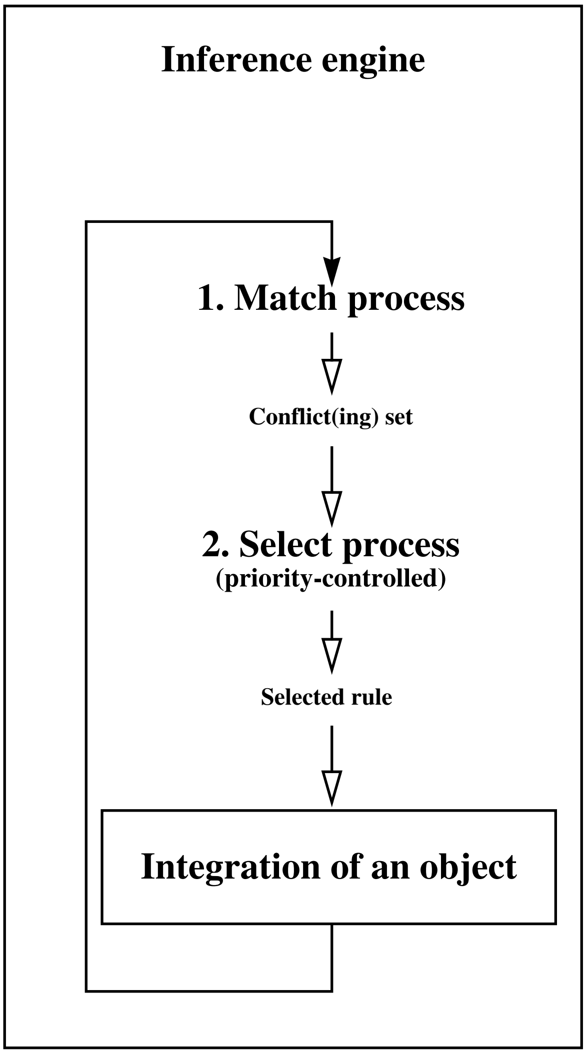


Figure 10.9 The life cycle of the inference engine used in IDAR.

10.2.4 User Interface

The following figure shows the user interface of the Intersection Designer **IDAR**.

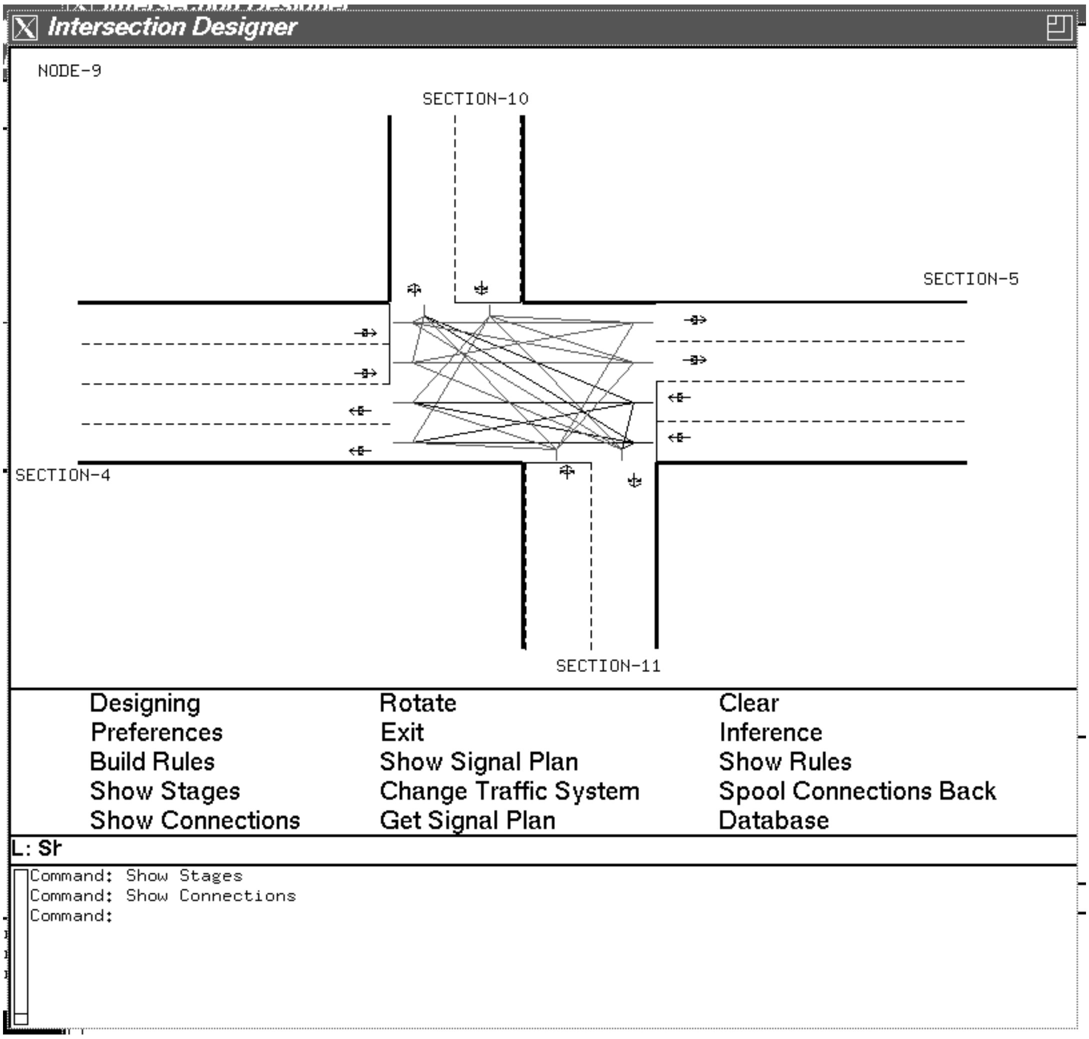


Figure 10.10 The User Interface of **IDAR**.

11 The Signal Control Architecture

11.1 Contract Net Protocol

The contract network is one of the concepts of distributed AI of Davis and Smith ([Davis & Smith 83]). By means of the contract communication protocol agents contract each other. The allocation of resources is the subject matter of the contracts. In this context resources do not only include memory and calculation time, but also knowledge, information and capability.

An agent which is the *manager* who wishes to solve a task divides it into partial task. He delegates partial tasks he does not want to solve or cannot solve, as he lacks the resources, to other agents. These agents are referred to as *contractors*. The manager contracts employees via bilateral negotiations. A contractor who is not able to solve a task completely can delegate in turn partial tasks.

The contract net protocol is executed according to the subsequent steps (see also Figure 11.1)

1) Invitation to bid
   - The manager sends the description of a task.

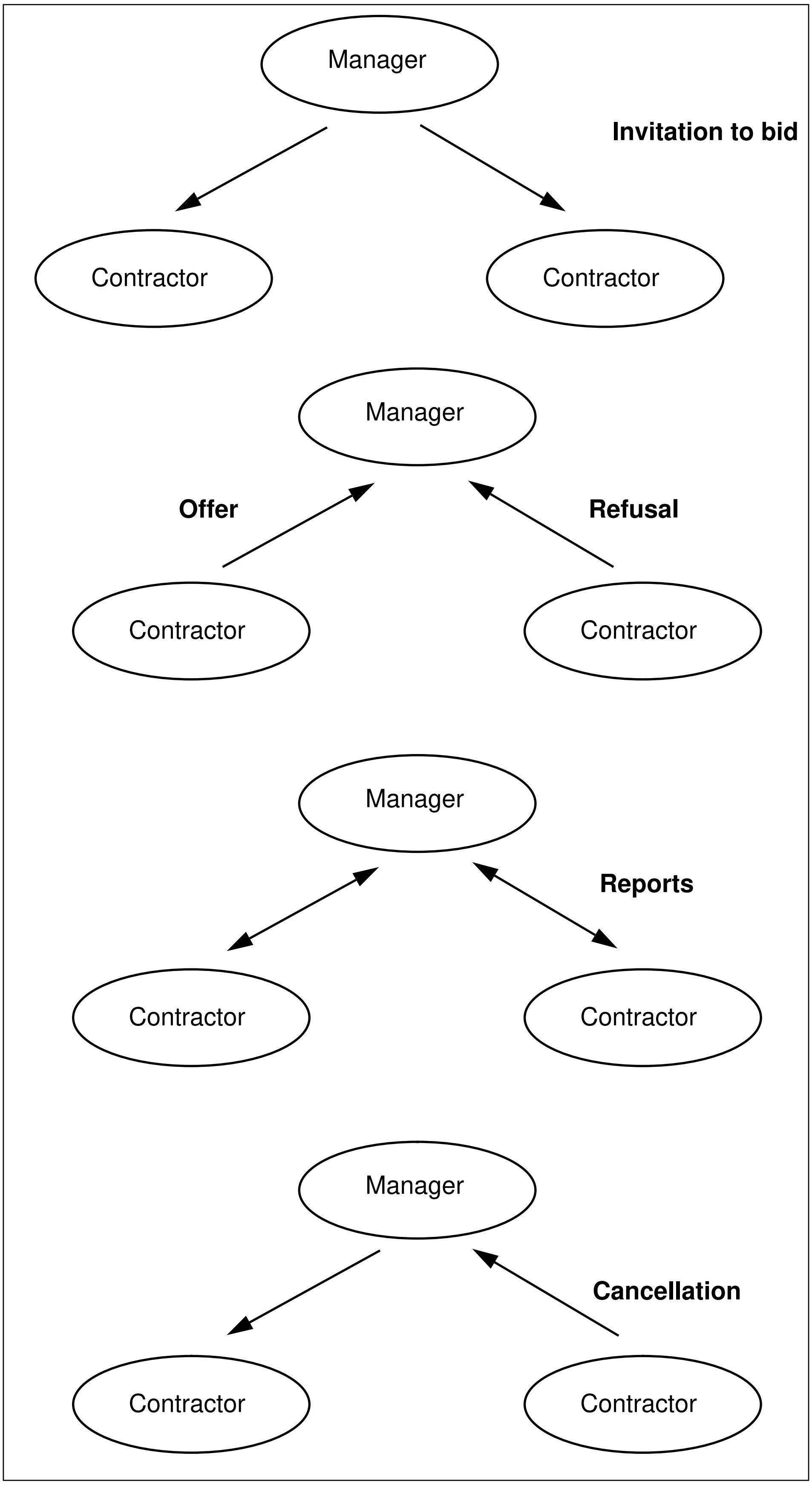


Figure 11.1 The stages of the contract net protocol.

2) Offer, Refusal

- Potential contractors evaluate the task. If they are capable of processing the task, they will sent an offer. Otherwise, they will refuse.

3) Contract

- The manager evaluates the offers and sends a contract to a contractor

4) Report

- The contractor processes the task according the contract. In the meantime he can interchange reports with the manager.

5) Cancelation

- Cooperation between the agents can be mutually canceled.

An application-specific language must be defined for the description of tasks, offers, reports and cancellations. This language can either be a simple finite language or a language similar to the natural languages. Not all agents must be equally capable of the language, as the technical terms of an expert system for integral equations must not necessarily understood by an expert system for art. Agents that do not understand a task should not make offers.

Agents of a contract network are isolated and independent of each other. They can represent knowledge on their roles, state, and environment anyway they like. Agents can be integrated to the network or removed from it, without the necessity for the programmer to know the entire system. He must only know the contract net protocol and know the corresponding partial language of his part.

Thus, a contract network is a heterogeneous, distributed and open system.

11.2 Applicability of Contract Networks

Although contract networks are highly adaptive systems, they are not suited for all types of tasks. The following will present criteria for the applicability of such systems. The following aspects will be examined:

- Division of Tasks
  - How can a global task be divided into partial tasks?

- Volatility)
    - How fast does a task change?

- Specialization
    - To what extent can agents having specific and restricted roles be used?

- Optimum Solution
    - Is an optimum solution of the task aimed at?

- Inconsistencies
    - Is the data provided to the agents consistent?
    - Are they complete?

11.2.1 Division of Tasks

The principle of the contract network is the division of tasks. For parallel and distributed processing, the partial tasks must be independent of each other. Thus, contract networks are suited for tasks of a well-defined ´top-down´ division/analysis. These kind of tasks can be described as a data and task hierarchy.

11.2.2 Impulsiveness

Real systems continuously change their state. The individual agents of the contract network have only local information on the current state of the system. An optimum solution of a task, however, can only be provided by means of global knowledge.

A low impulsiveness of the system provides the agents the necessary time to acquire global knowledge. Consequently, the task can be centrally processed.

A high impulsiveness renders the contract net protocol too slow to react to system modifications.

11.2.3 Specialization

A manager assumes that several agents with similar roles and similar behavior are available to process a task. If only one agent is available, the contract net protocol can be replaced by a procedure call. If the system has several agents of a similar behavior, all agents will make an offer. This causes an unnecessary load on the system.

11.2.4 Optimum Solution

The agents are assigned tasks by means of the contract net protocol. An optimum assignment, however, cannot be guaranteed.

Only local information on the system state will be taken into account by the contractor when elaborating the offer. Managers send their invitations to bid without prior consultation. Thus, optimum assignments will only take place locally. Local optimum decisions must not necessarily result in global optimum solutions. The following example is based on the assumption of two managers A and B and two potential contractors X and Y. A assigns the offer of X a 0,9 and the one of Y a 0,8. B assigns the offer of X a 0,8 and the one of Y a 0,2.

Table 11.1

	A	**B**
X	0.9	0.8
Y	0.8	0.2

Both A and B will try to contract X. From the global point of view the assignment of (A, Y) and (B, X) is an optimum solution. This problem known as “*prisoners dilemma*” can only be solved by an information interchange between A and B. A large number of Actors can render the communication rather time-consuming.

Another problem is the impossibility to forecast future system states. The following assumes that A and B are tasks and X and Y contractors suited to process these tasks. First, task A will be negotiated, then task B.

Table 11.2

	A	**B**
X	0.9	0.8
Y	0.8	0.2

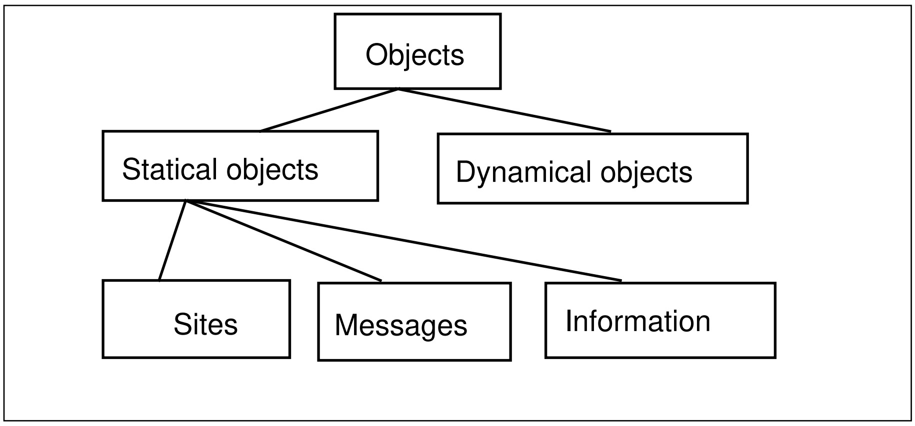


Figure 11.2 System objects

X will win the contract, as it has got the better assignments than Y. Task B will then be assigned to agent Y, as X is already occupied at this point of time. The optimum assignment, however, would be (X, B) and (Y, A). The problem of temporal assignment is inherent in all dynamic systems. It cannot be provided an optimum solution without forecast. Inconsistencies

Each agent of the network has only local information. The agent is not able to determine whether the available data is current, consistent and complete. Inconsistencies in distributed systems can result from transmission errors, sensor errors, synchronization and delay errors. For instance, an agent on intersection X is informed that 10 vehicles move towards intersection Y. The agents on intersection Y can only have the obsolete information that 2 vehicles are approaching and will make an offer on the basis of the obsolete information. The contract net protocol does not provide mechanisms to solve inconsistencies. Therefore, and agent must assume that the available data is consistent and complete.

11.3 Organization Structure

Figure 11.2 displays the different object classes. The objects of the system divide into dynamical and statical objects. Statical objects are:

- Immobile Sites

    - Agents act at different immobile geographical sites or mobile objects such as machines or vehicles. They can be regarded as objects. The following will focus on immobile sites.

- Messages
    - Agents intercommunicate via messages. Messages are, thus, objects.

- Information
    - The system state is available to the agent on various levels of abstraction. It is treated as information.

The agents are the dynamical objects in the system. Figure 11.3 shows how the organizational structure of the network divides into sub-organizations:

- Control Network
    - The control network consists of agents which control a common process. The agents are referred to as experts in the following. The relations between them are "controlled" and "communicates with".

- Sites
    - The site objects divide into different abstraction levels governed by the relation "part of".

- Information Pool
    - Experts need information on the current, past and future state of the system to perform their tasks. The contract net protocol is not suitable for the acquisition of information and would increase the communications overhead. It seems to be useful to introduce an *information pool* as suborganization. It consists of one or several cooperating or competing publishers. A publisher has to collect, evaluate and distribute information so that it is available to the agents at the right time and place. A publisher wants to contract/acquire as many experts as possible. The information pool is a dynamical organization. Do not confuse it with the blackboards, which are statical objects.

- Organization Network
    - Organizational knowledge can be implemented in the control network. For reasons of a clear structure organizational tasks are performed by a separate suborganization.

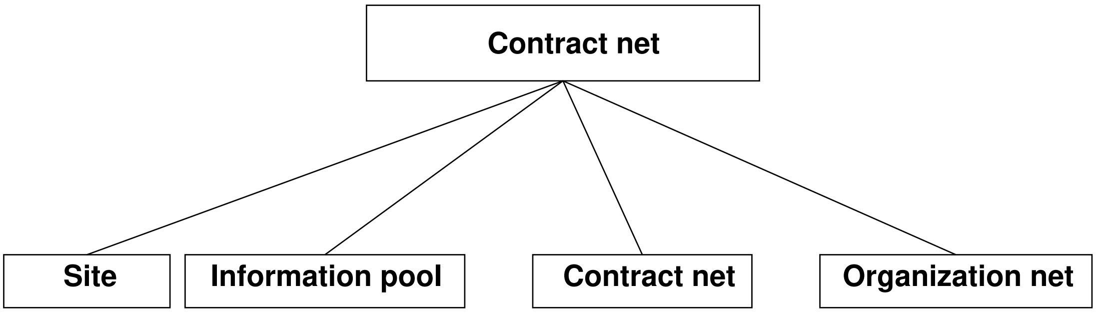


Figure 11.3 Suborganization

The following describes the organizational structure of the control network.

11.3.1 Control Network

Various procedures or algorithms can be used to solve difficult control and optimization problems. The approach is chosen according to the marginal conditions. Furthermore, certain problems are solved differently by various experts without prior knowledge on which is the more suitable approach. One agent, suited to solve a problem under certain marginal conditions, will be used for each procedure. Such an agent is referred to as expert. The manager divides the problem which is to be solved into partial problems and asks for bids. The experts who deem themselves suited submit their offers. The manager will commission the most suitable expert. As the manager himself is also an expert, it will be activated by an hierarchical superordinate manager and has to compete for orders like his experts do. The organizational structure continues up to a group of agents who form the user interface. As the users themselves are often governed by a similar organization, the user-to-machine transition is seamless concerning this matter.

Figure 11.3 provides a schematic display of the control networks organizational structure, which is a strict hierarchical one. Each local group is subordinate to a manager who controls the agents of his group. This manager is in turn controlled by a manager on a superordinate level. The top level is controlled by the user.

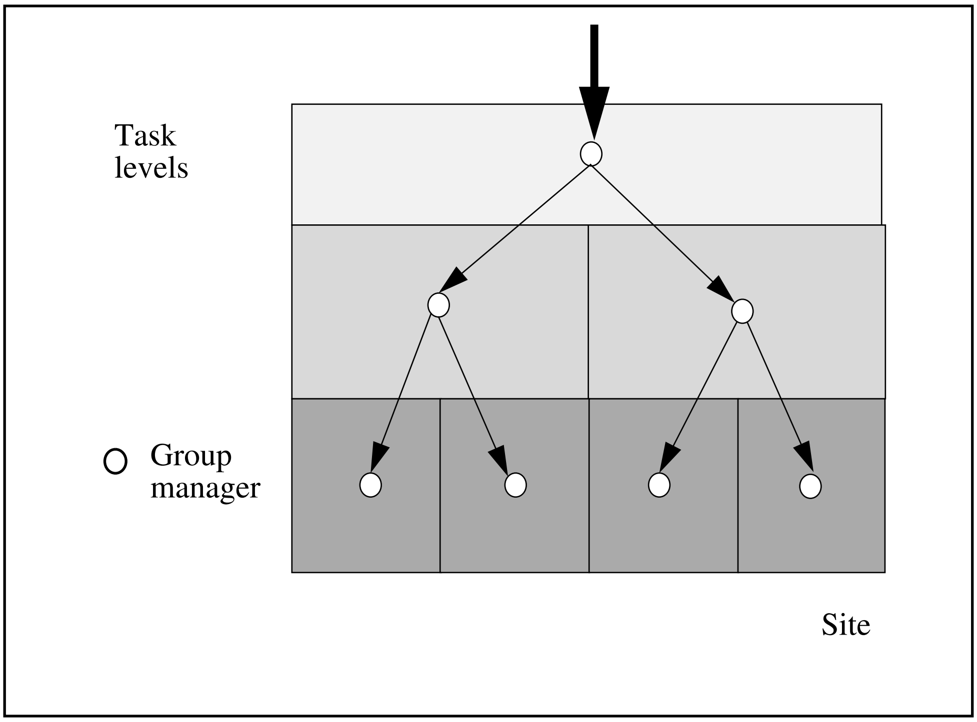


Figure 11.4 Organizational structure of the control network

11.3.1.1 Change of Experts

The change of an expert must be performed carefully when controlling technical processes, as the current process may not be submitted to a sudden change and because a change in control must be temporarily and logically coordinated. For this reason an expert is not allowed to terminate its activity on his own. He must continue until he is told to terminate by the manager. Before, the manager must provide for a new expert to take over at the right time and to continue.

The following events can cause a manager to induce a change of experts:

- His own initiative:
    - The manager who monitors and evaluates the activities of his experts might draw the conclusion that the expert does not perform properly or that a connection was disrupted.

- The expert´s initiative:
    - An expert decides that the marginal operation conditions do not match his qualification.

- The request of another expert:
    - Another subordinate expert asks the manager to change the active expert, since he cannot cooperate with him or because he can perform better as the marginal conditions have changed.

- The request of another manager:
    - A manager of the same hierarchy can set marginal conditions for the use of experts to ensure coordination. If the marginal conditions are not fulfilled, the expert must be changed.

- The request of the own manager:
    - The manager is requested by the superordinate hierarchy to terminate his activities. To do so, he must ask his experts to terminate their activities.

11.3.1.2 Conflict Solutions via Adaptive Learning

The following problem-independent conflicts will arise for each manager:

- Selection of a suitable expert if several offers have been submitted

- Selection of a suitable expert if 0 offers have been submitted

These problems can be solved by means of a qualification system according to which the quality of an expert is assessed on the basis of his activities. In case of several offers the manager chooses the expert who has obtained the best marks. Invitations to bid will be provided with qualification levels, which denote how critical the manager will judge the contractor. If the invitation is not answered by any experts, the manager will lower the qualification level until a sufficient number of possible contractors answer. Qualification level 0 means that an expert will not be judged. Consequently, all experts must submit an offer.

Managers, who are at the same time experts, differ in the way to analyze problems and in their qualification strategies.

11.3.1.3 Advantages of the Hierarchical Control Structure

This organization structure provides the subsequent advantages:

- An open system
  - New experts can be introduced. An adaptation of the upper hierarchies is not necessarily required.

- Robustness
  - Incompatible approaches can be implemented in the same system.

- Capability to learn
  - The system has the capability to learn via the qualification system. It might happen, however, that the system runs into a local minimum. This will mainly depend on the qualification strategy.

- Simplicity
  - The communication in the system is clearly defined and reduced to the minimum.

11.4 Organizational Structure for Traffic Control

11.4.1 Sites

The following site objects will be used:

- Link
  - Links are regarded as part of intersections. There are two types of links:
    - approach links
    - exit links

- Crossing

- A crossing is the smallest unit to be controlled.

- Area
    - An area is a set of intersections which are governed by a joint coordination strategy. For the sake of simplicity only overlapping areas will be regarded.

- Network
    - A network is a set of areas.

A site has the following organizational structure:

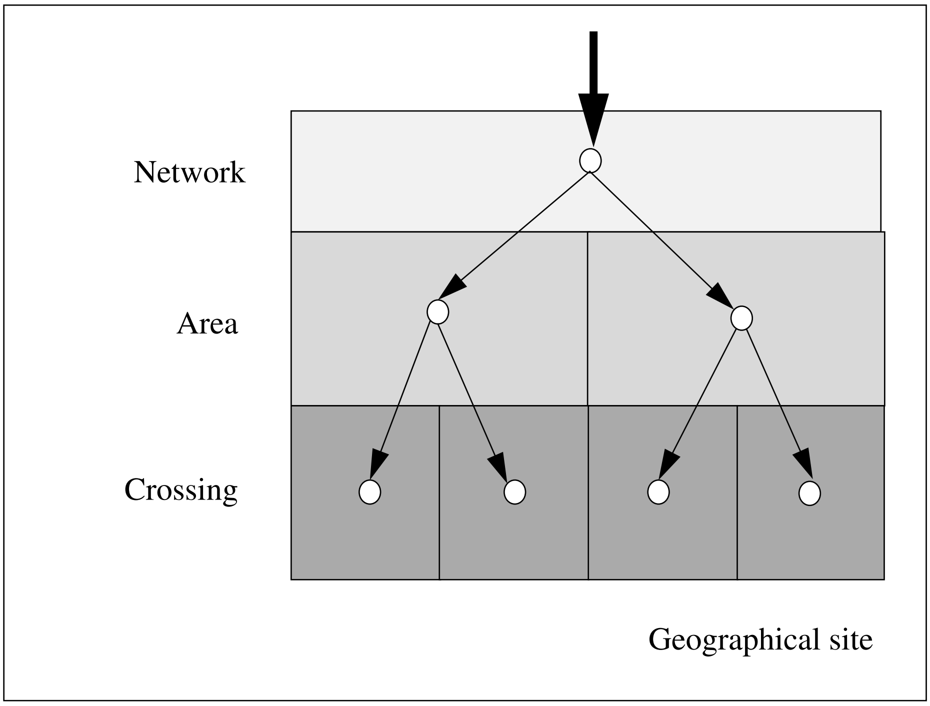


Figure 11.5 Organizational structure of a site.

11.4.2 Information Pool

An information pool consists of a publisher. The agents of a publisher are referred to as *blackboard* because of their simplicity. Each site is assigned a blackboard. They divide into:

- Crossing blackboard
    - Management of crossing state data
    - Storing the signal plan for the crossing
    - Management of information on the group of agents assigned to the crossing
- Area blackboard
    - Management of area state data
    - Management of information on the group of agents assigned to the area
- Network blackboard
    - Management of network state data
    - Management of information on the group of agents assigned to the network

11.4.3 Control Network

Each site is assigned a group of agents. The following groups are employed:

- Crossing group
    - Determination of crossing state data
    - setting of the signal plan
- Area group
    - Coordination at crossings
    - Determination of area state data
- Network group
    - Division of the network into areas
    - Determination of network state data

The control network has the following organizational structure:

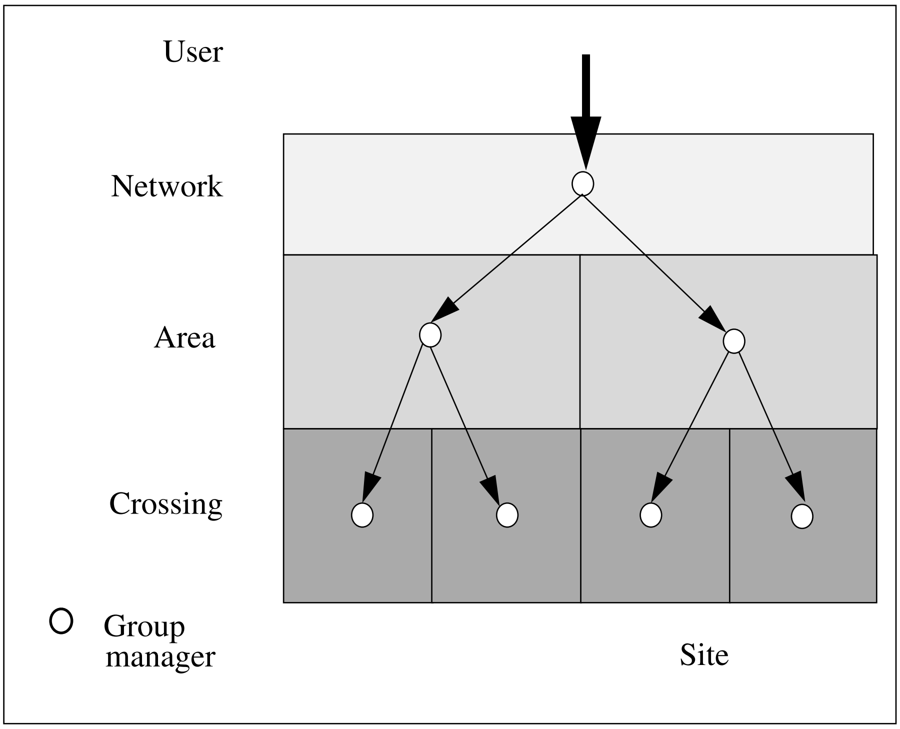


Figure 11.6 Organizational structure of the control network

The tasks of a network can be hierarchically structured:

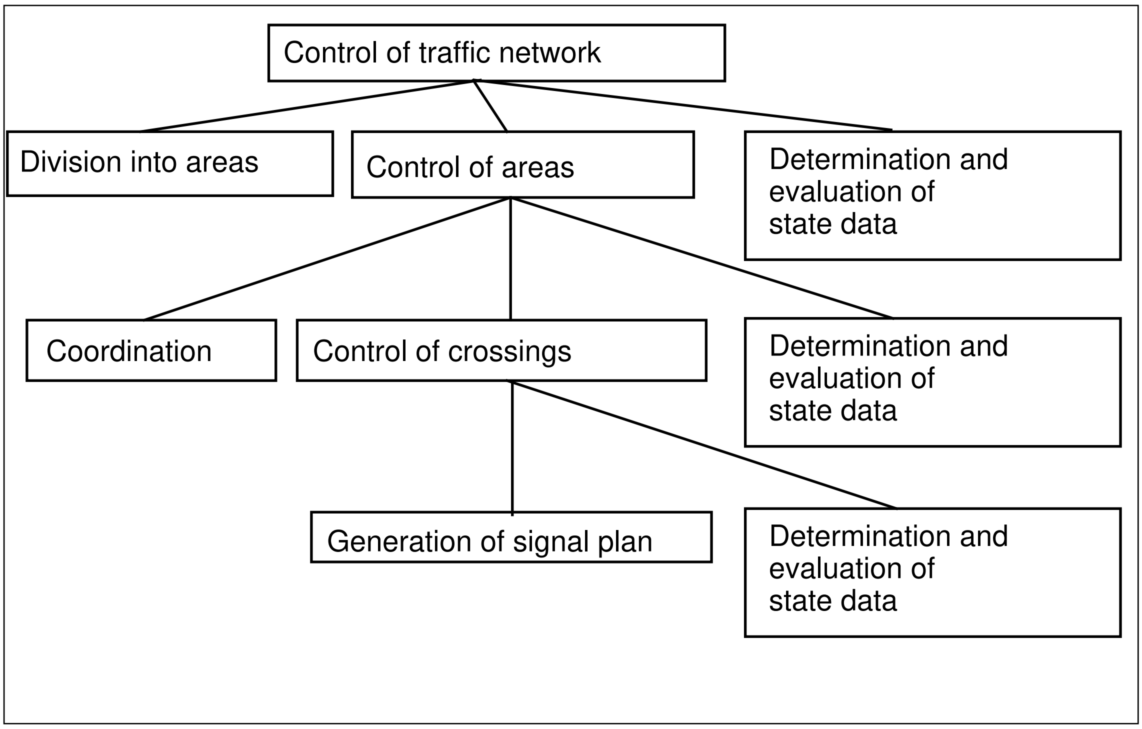


Figure 11.7 The hierarchical structure of network tasks.

The task is performed by the following organizational structure:

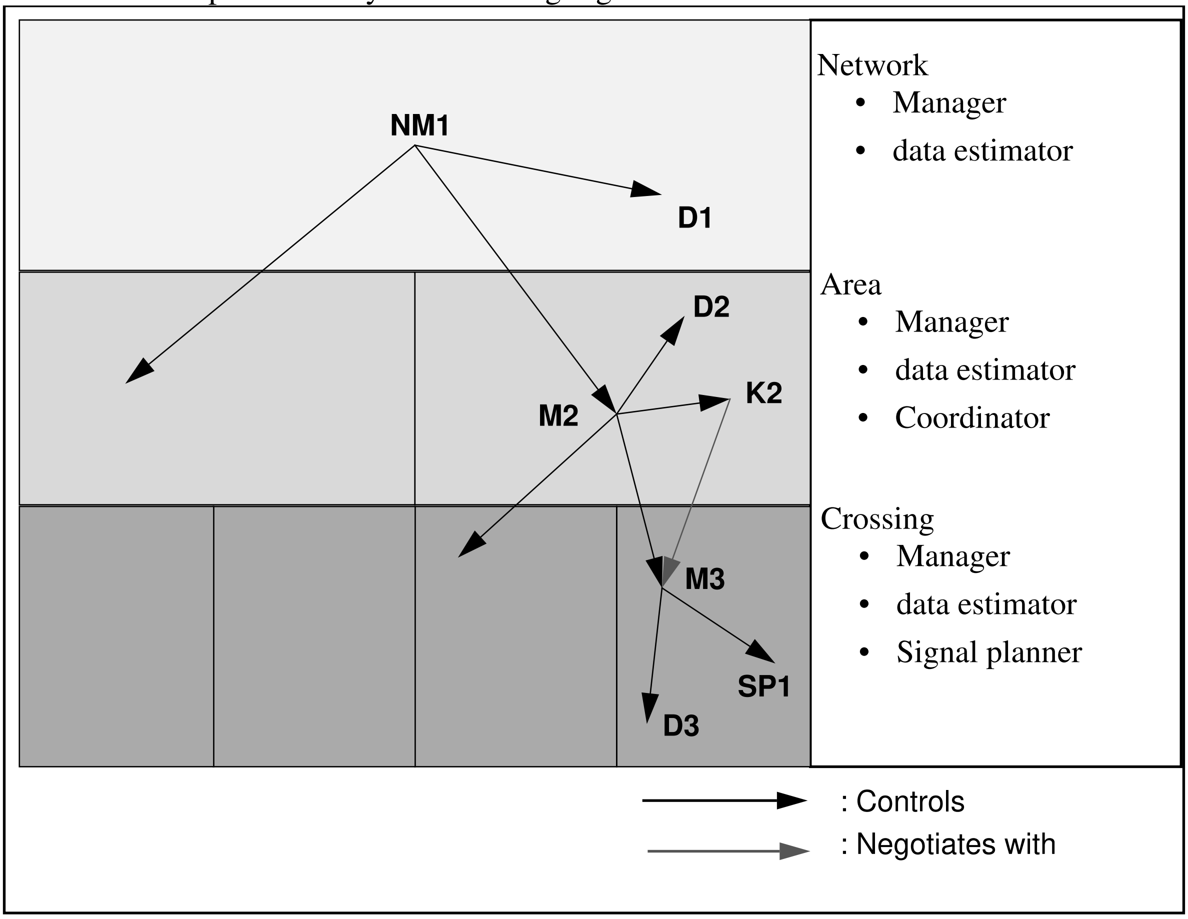


Figure 11.8 Organizational structure of network tasks.

The following sections explain how the network operates; the roles of the individual groups are refined and it will be shown that the contract net protocol and the organizational structure are suitable for this control task.

11.4.3.1 Operation of the Control Network

The top level is controlled by a network manager. The user sends his invitation to bid for the task of controlling the current traffic. He will chose a suitable offer in accordance with the desired strategy and instructs the manager to control the task execution.

The network manager divides the traffic network into areas and will perform the required activities according to the contract.

Each area is assigned a manager. The area manager in turn assigns each crossing a crossing manager. The crossing manager selects a signal planner. In case a crossing blackboard needs a signal plan, it will turn to the corresponding signal planner.

If necessary, a signal planner can delegate partial task to other experts, e.g. he can use a signal plan optimizer or an expert on traffic forecasts.

11.4.3.2 Suitability of the Contract Net Protocol

The criteria of section 11.2 are largely complied with:

- Task analysis
  - A task can be hierarchically analyzed. Some tasks, however, required synchronization:
    - The determination of a level´s state data depends on the data of the lower level.
    - The starts of the signal plans must be synchronized.
    - A change in area division cannot happen at any time.
  - The task on a level are independent of each other and can be processed simultaneously distributed to the sites.

- Impulsiveness (*volatility*)
  - The reactions of the system occur within seconds.

- Specialization
  - The agent classes are largely specialized.
  - Different types of behavior are to be implemented in an agent class.

- Optimum Solution
  - All present control systems are suboptimal.
  - Until now no generally valid criteria exist to assess traffic guidance systems.

- Inconsistencies
  - Qualitative state parameters are used.
  - Inconsistencies resulting from transmission errors can be removed by an appropriate communications protocol.

- In case the connection between a site and the above level is disrupted, the agent group can continue processing the task without coordination. A reorganization of the site can possibly remedy the damage.
- A little shift of the agents´ internal clocks is not of importance, as the impulsiveness is rather low.

11.4.3.3 Suitability of the Organizational Structure

The presented organizational structure complies with the criteria of section 11.

- Coverage
    - Each partial task was assigned to a agent class.
- Connectivity
    - All required connections were defined in the organizational structure.
- Feasibility
    - The tasks of the individual agents are relatively simple. The arithmetic effort depends on the agent´s behavior.
    - The communications effort is clear and calculable.

11.5 Implementation of Behaviors for Signal Plan Selection

A simple signal program selection strategy has been implemented within the framework of a simulation. The ACTRICE simulation environment for Actors was used. The examination is to show how and whether the concept of contract networks can be used for traffic control.

The implementation intentionally avoids the problems of synchronization. Instead of the internal clocks synchronous calls are used to synchronize the Actors. Figure 11.9 shows a taxonomy of the system´s Actor classes.

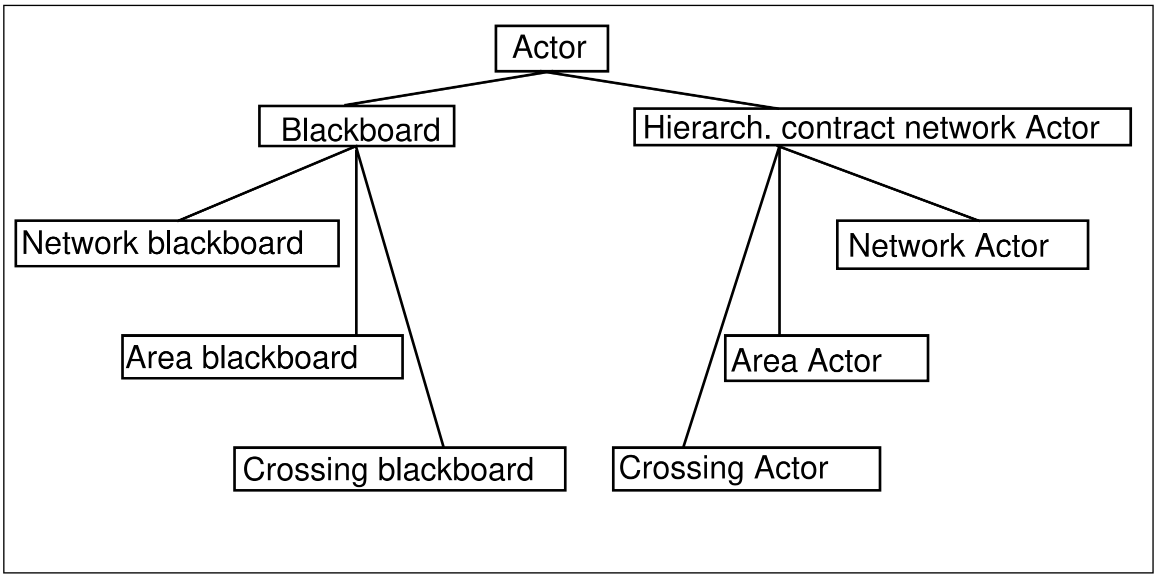


Figure 11.9 Taxonomy of the system´s actors.

11.5.1 Contract Network

The definition of the contract network actors consists of the class definitions shown in Figure 11.10.

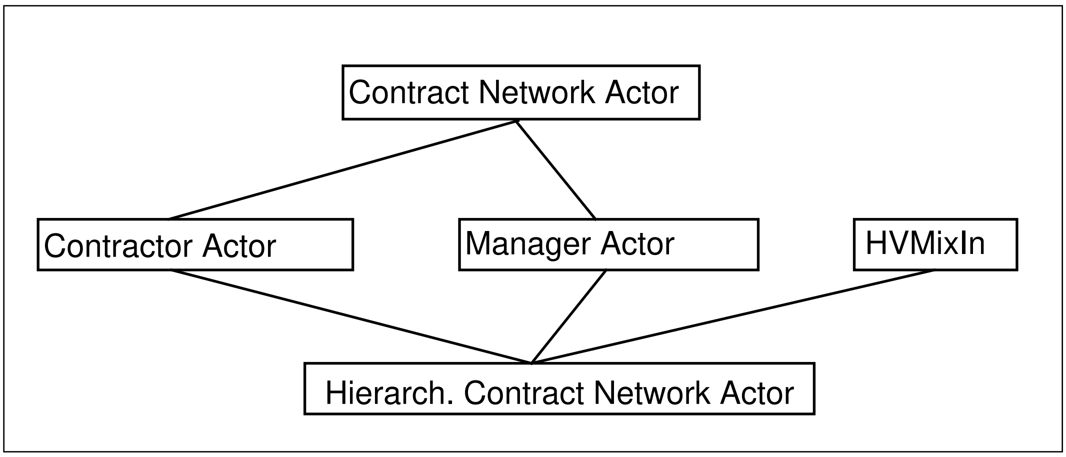


Figure 11.10 The structure of a network actor.

An hierarchical contract network Actor has the following states and behaviors:

- Waiting
    - Invitation to bid
    - Contract
- Negotiating
    - Offer
    - Refusal
- Processing
    - Termination

It has the following state transitions:

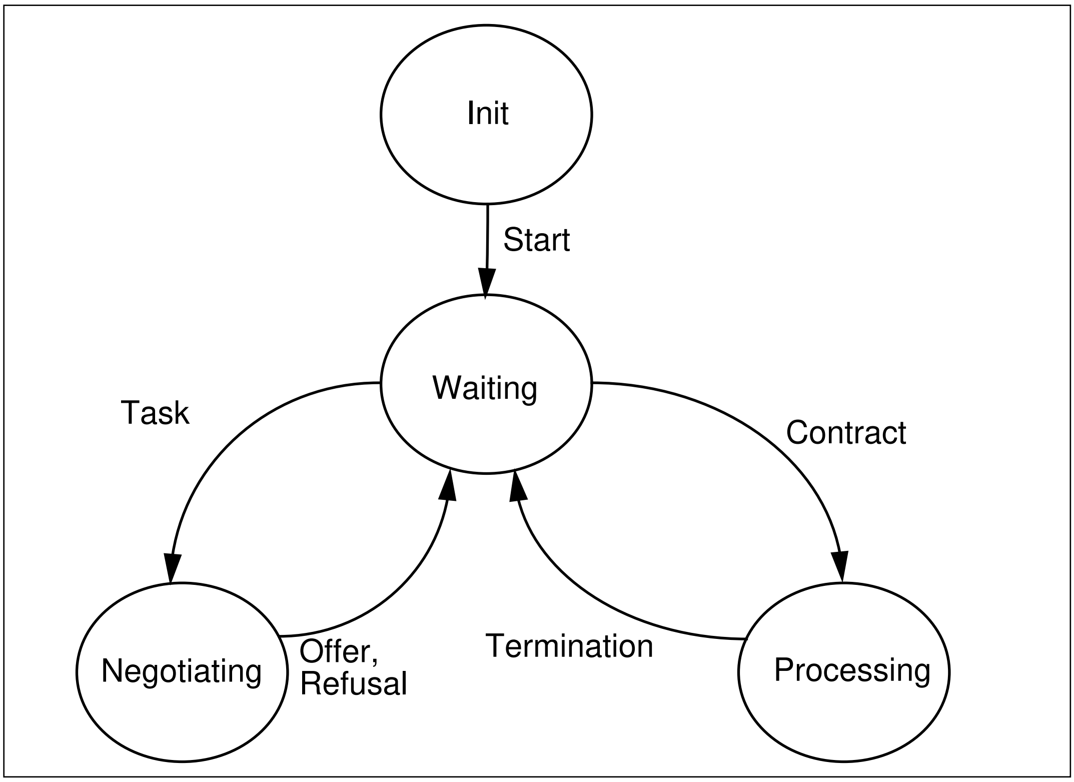


Figure 11.11 State transitions of a network actor

Actions for the following behaviors must be defined for each Actor class of the control network:

- Task

- Contract

11.5.2 Qualification System

The qualification levels range from 0 to **`*max-noting-level*`**. In case of level 0, the contractors will not be judged. In case **`*max-noting-level*`** is set, they will be critically judged.

Qualification levels are real numbers between **`*min-note*`** and **`*max-note*`**.

Let us assume the $\Delta X_{k+1} = [X_{k+1} - X_k]$ is the qualitative modification of the target parameter at time between k and k+1. $\Delta X_{k+1} \in \{+, 0, -\}$.

Let us assume $\Delta Y_{k+1} = [Y_{k+1} - Y_k]$ is the qualitative modification of the state parameter.

Let us assume that N_k is the qualification of Actor A at the time k and B is the current qualification level.

The qualification of the Actor at time k+1 results from:

$$N_{k+1} = [\ N_k + \alpha\ B\ T[\Delta X_{k+1}\ ,\ \Delta Y_{k+1}\ ]\ ]$$

Where T stands for a table. For typical values refer to the subsequent table:

Figure 11.12 Qualification transition

ΔY \ ΔX	**-**	**0**	**+**
-	0	+0,5	-1
0	-0,5	0	-0,5
+	+1	+0,5	0

11.5.3 Blackboards

Blackboards have the following state transitions:

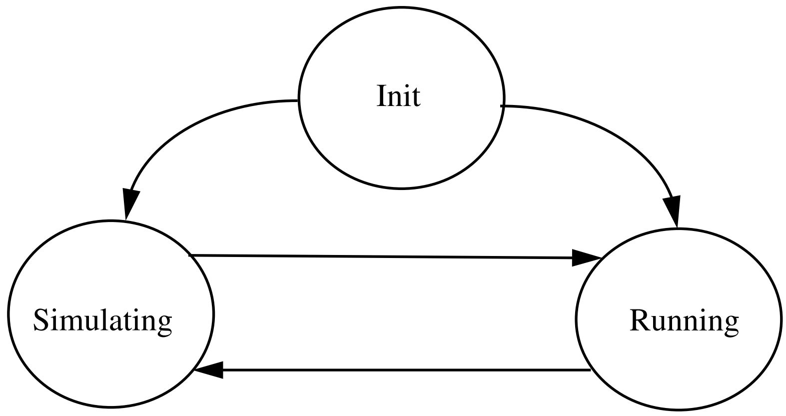


Figure 11.13 State transitions of a blackboard

The state “simulating” manages simulation data, while the state “Running“ handles the current data.

The duration of the state is entered during the state transition.

The internal clocks of the Actors will not be used within the implemented simulation. Instead, the state modifications are transmitted via synchronous messages of a blackboard level to the next one. The active network manager synchronizes the state modifications of the network blackboard.

11.5.3.1 Crossing Blackboard

The crossing blackboard manages the following data:

- Control Network Actors
    - List of active Actors
    - List of waiting Actors
- State Parameters
    - Density on approach roads
    - Mean density on approach roads
- Target Parameters
    - Delays on the approach roads

    - Total delay on the approach roads

- Signal Plans
    - Signal plan for simulation
    - Signal plan for the run

A signal plan consists of a list of events:

- Signal plan event
    - Point of time (absolute, relative)
    - List of green traffic lights

The signal plans are read by the simulation. If a signal plan is void, the active planner is asked to complete the signal plan.

11.5.3.2 Area Blackboard

The area blackboard manages the following data:

- Control Network Actors
    - List of active Actors
    - List of waiting Actors
- State Parameters
    - Density on intersections
    - Mean density in the area
- Target Parameters
    - Delays on intersections
    - Total delay in the area

11.5.3.3 Network Blackboard

The network blackboard manages the following data:

- Control Network Actors
  - List of active Actors
  - List of waiting Actors
- State Parameters
  - Density in the areas
  - Mean density in the network
- Target Parameters
  - Delays in the areas
  - Total delay in the network

11.5.4 Crossing Behavior

11.5.4.1 Crossing Manager

The crossing manager knows the following types of behavior:

- In the state “waiting”
  - Task
    - Negotiation with data estimator
- In the state “processing” the subsequent reports are processed
  - Data evaluation
  - Request to change planner by the signal planner
  - Request to change planner by the coordinator
  - Instruction to change planner by the coordinator

Data are evaluated as shown below:

1) Synchronous call of the data estimator
2) Synchronous call of the signal planner

3) Evaluation of the signal planner on the basis of the data provided by the blackboard
4) Requests to change the planner are forwarded to the area manager.
5) A change of the signal planner is performed as follows:
6) Send notice to terminate to the current planner
7) Sent contract to the new planner. The termination time of the previous planner is entered as starting time.

11.5.4.2 Density Estimator

The density estimator simulates sensors on the approach roads.

On the basis of the simulation´s event list the mean traffic density is calculated for a defined site on the section valid for the time of the simulation. The mean density of the class is used for each density class.

Let us assume that T is the duration of the simulation, that T_i is the duration of the density zone i at the sensor and that d_i is the mean density of this zone.

The mean density of section D results from:

$$D = \frac{T_i \times \sum_i d_i}{T}$$

11.5.4.3 Delay Estimator

The delay on a section is estimated to be the area of STOP density zones which are generated during simulation:

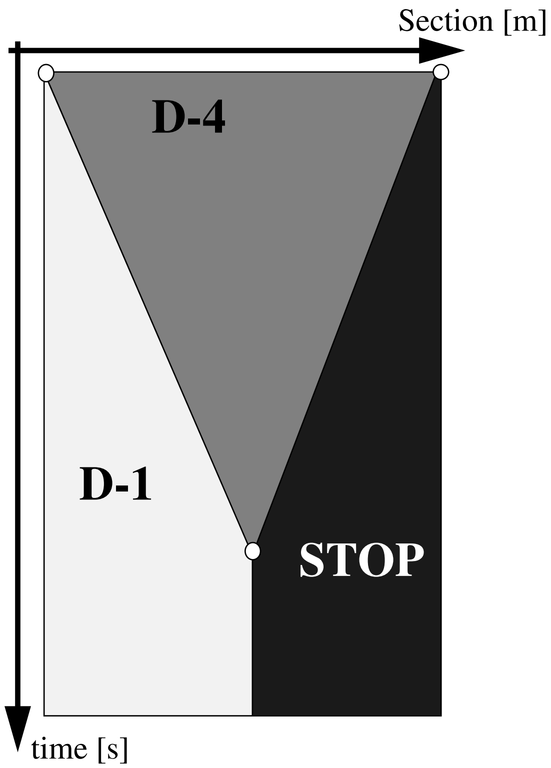


Figure 11.14 The stop zone gives the delay directly.

11.5.4.4 Signal Planner for Signal Program Selection

11.5.4.4.1 Criteria

The following criteria are taken into account:

- Current Planner at the Intersection
    - The signal planner may only become active after certain other planners.
- Current Traffic State
    - The marginal conditions for the use of the planner are entered for each qualification level.

```
(and   (lower (density "LINK-19")
              (density "LINK-20"))
       (greater (density (actual-cross)) "D-4"))
```

Figure 11.15 An example of marginal conditions.

The marginal conditions are arbitrary Lisp-expressions. The following Lisp functions have been predefined:

- Density
    - provides the estimated density of a simulation object
- actual-cross
    - provides the site of the planner
- Greater
    - compares two density classes
- Lower
    - compares two density classes

11.5.4.4.2 Signal Plan

A signal plan for the signal plan selection has the following parameters:

- A list of signal plan events; the event times in relation to time 0.
- Relative starting time. The time in a relative plan when the signal plan is started.
- Relative termination time. The time in a relative plan when the signal plan can be terminated.
- Cycle time of the signal plan.

When entering the signal plan into the blackboard the absolute event times are determined. One signal plan is generated for each cycle.

On the termination of his activity the signal planner removes all entries from the blackboard starting with the defined termination time. This point of time is transmitted to the manager to facilitate synchronization with the succeeding planner.

11.5.5 Area Behavior

11.5.5.1 Area Manager

An area manager has the following behavior types:

- In the state “waiting”
    - Task
        - Negotiation with data estimators
        - Negotiation with crossing managers
        - Negotiation with coordinators
- The following reports are processed in the state “processing”
    - Evaluation of data
    - Request to change planner by a crossing manager
    - Negotiations on coordination

The evaluation of data is performed as follows:

- Synchronous call of the data estimator
- Synchronous call of the signal planner
- Evaluation of the crossing managers and the active coordinator on the basis of the data of the blackboard

The requests to change a planner are collected. If their number exceeds the number of intersections in the area, negotiations with the coordinators will be started. If the negotiations produce a change of the coordinator, the active coordinator will be given notice to terminate and a new one will be contracted.

11.5.5.2 Area Density Estimator

The density values are extracted from the crossing blackboards, entered into the area blackboard and the mean density in the area is calculated as follows:

$$D = \frac{1}{I} \times \left(\sum_i d_i \right)$$

I : Number of Intersections in the area

11.5.5.3 Area Delay Estimator

The delays are extracted from the crossing blackboard, entered into the area blackboard and the total delay in the area is calculated.

11.5.5.4 Area Coordinator

Each coordinator has a list of signal planners which are compatible according to his strategy. An empty list complies with the strategy not to execute centralized coordination. When negotiating with the crossing managers on the change of the signal plan, only signal planners of this list (provided it is not empty) will be offered. If the signal planner has an offer for each intersection, it can submit an offer to the area manager.

If the coordinator wins the contract, it sends instructions to change the planner to the crossing managers.

If a coordinator is given notice to terminate, it is set to the state “waiting”.

11.5.6 Network Behavior

11.5.6.1 Network Manager

If a network manager receives a task announcement, it will start negotiations with:

- Density estimators
- delay estimators

- area managers

If a manager wins a contract, he will contract data estimators and area managers. Then, he will start to control the network.

In this context he performs the following state modifications:

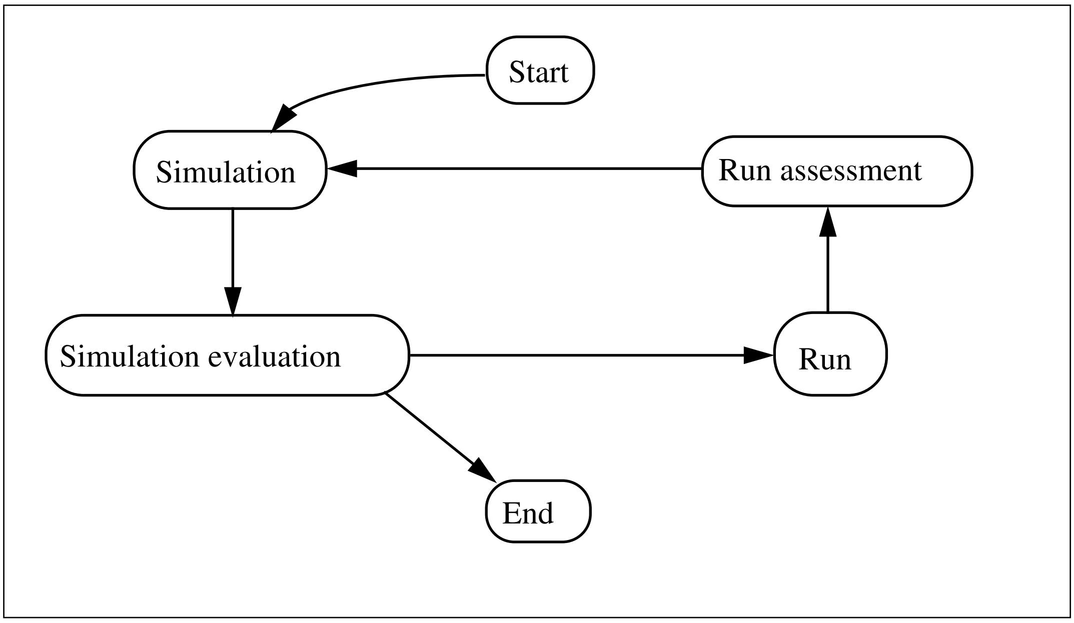


Figure 11.16 State transitions of a network manager actor.

The subsequent actions are performed:

- Simulation
    - Change the blackboard states to “simulating”
    - The simulation is called with the simulation duration
- Simulation evaluation
    - The area managers are asked to evaluate the simulation.
    - Subsequently, the network data estimators are asked to perform their calculations.
    - Terminated messages are expected from the area managers which negotiate coordination.
- Run

- Modification of blackboard states to “processing”
- The simulation is called with the simulation duration
- The time is increased by the run time

- Assessment of the run
  - The area managers are asked to evaluate the simulation.
  - Then, the network data estimators are asked to perform their calculations.

11.5.6.2 Network Data Estimator

The data are extracted form the area blackboards and averaged.

11.5.6.3 Network Delay Estimator

The data are extracted from the area blackboards and the total delay is calculated.

11.6 User Interface

The following hardcopies show the current state of the actor-based signal control component for the Sapporo prototype. The user is able to browse through the contract net hierarchy and to inspect an arbitrary actor simply by clicking on it. The debugging interface to the actor system has already be shown in [Wild & Berning 91].

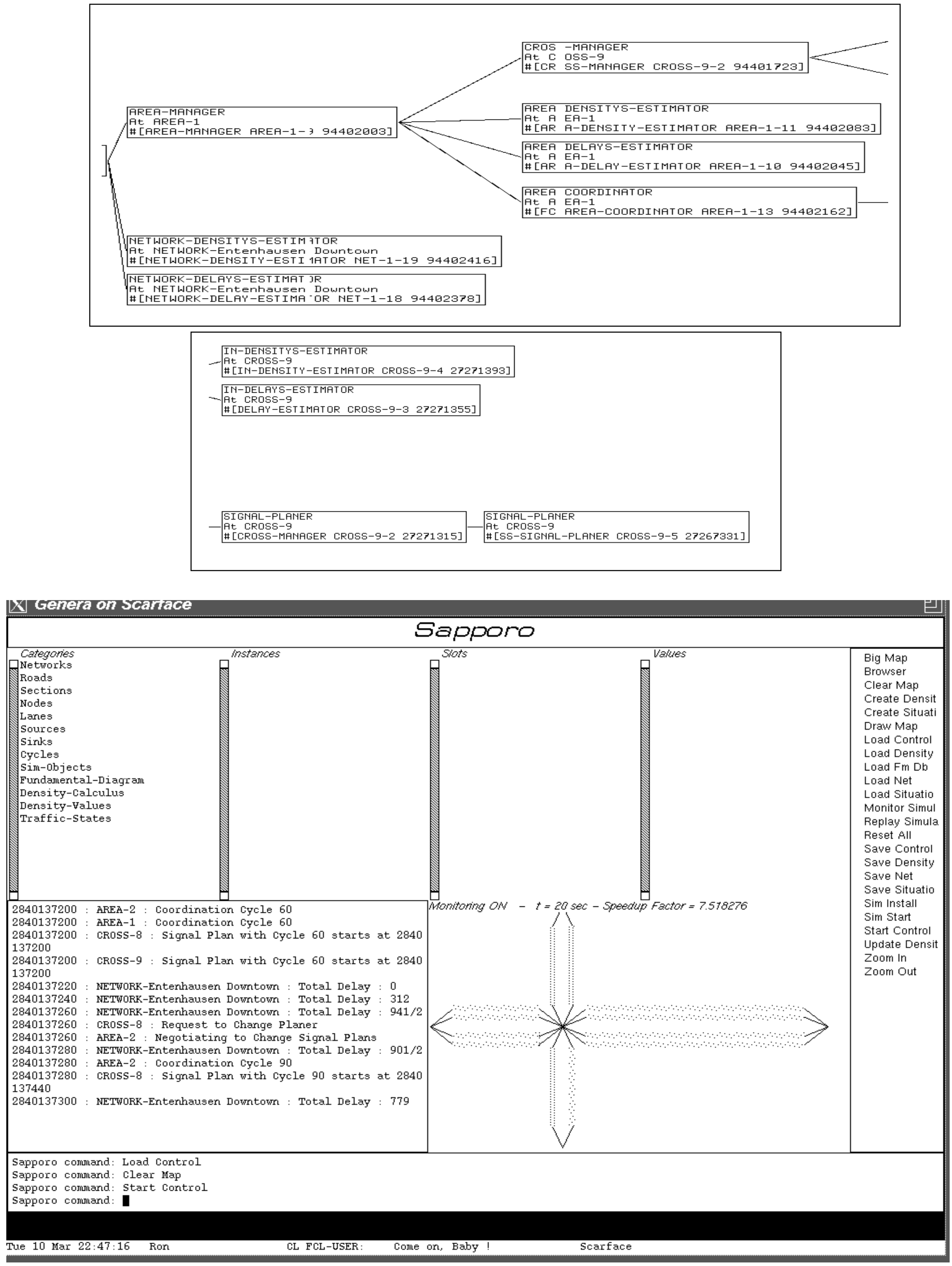


Figure 11.17 The actual user interface to the actor-based contract network for signal plan control within the Sapporo prototype.

12 References

Acronyms used:

AAAI	=	American Association for Artificial Intelligence,
ACM	=	Association for Computing Machinery,
AI-Journal	=	Artificial Intelligence: An International Journal,
CACM	=	Communications of the ACM,
JACM	=	Journal of the ACM,
KIFS	=	Künstliche Intelligenz, Frühjahrsschule (German)

[Agha 85] G. Agha
Semantic considerations in the actor paradigm of concurrent computation
Seminar on Concurrency, Springer Verlag, 1985

[Agha 86] G. Agha
ACTORS: a model of concurrent computation in distributed systems
MIT Press, 1986

[Ask 90] Ask, A.
Wissensbasierte Vorhersage auf Basis von typischen Ganglinien an Induktionsdetektoren
Diplomarbeit, Forschungszentrum für Informatik FZI, Universität Karlsruhe (1990).

[Barr et al. 89] Barr, A., Cohen, P. A., and Feigenbaum, E. A. (Eds.)
The Handbook of Artificial Intelligence Volume IV
Addison-Wesley, Mass. (1989).

[Bobrow 84] Bobrow, D. G.
Qualitative Reasoning about Physical Systems: An Introduction
AI-Journal 24, 1-3 (1984) 1 - 5.

[Bocionek 1990] Bocionek, S.
Modulare Regelprogrammierung
Vieweg Verlag, Braunschweig 1990.

[Briot 88] J.-P. Briot
From Objects to Actors: Study of a limited Symbiosis in Smalltalk-80
Technical R
Report LITP 88-58 RXF , September 1988

[Briot 89] J.-P. Briot
From Objects to Actors: Study of a limited Symbiosis in Smalltalk-80
ACM SIGPLAN Vol.24 No. 4, April 19 89

[Briot 89b] J.-P. Briot
Actalk: a testbed for classifying and designing actor languages in the Smalltalk-80 environment
Proc. ECOOP 89, 1989

[Brownston 1985] Brownston, L.; Farrell, R.; Kant, E.; Martin, N.
Programming Expert Systems in OPS5 - An Introduction to Rule-Based Programming
Addison-Wesley Publishing Company 1985.

[Bürle & Lehmann 88] Bürle, G. und Lehmann, A.
Künstliche Intelligenz und Simulation
Interner Bericht 2/88, Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe (1988).

[Clinger 81] W. D. Clinger
Foundation of Actor Semantics
AI-TR-633 MIT Artificial Intelligence Lab, May , 1981

[Dalchow 91] O. Dalchow
Testbericht - Simulation mit Sapporo
Interner Bericht, Forschungszentrum Informatik, Universität Karlsruhe, 1991

[Dalchow 92] Dalchow, O.
Qualitative Modellierung und Simulation von makroskopischen Verkehrsgrößen auf Kreuzungen
Diplomarbeit, Forschungszentrum für Informatik FZI, Universität Karlsruhe (1992).

[Davis & Smith 1983] Davis,R. and Smith R.G.
Negotiation as a Metaphor for distributed Problem Solving
Artificial Intelligence 20, 1983.

[de Kleer & Bobrow 84] de Kleer, J. and Bobrow, D.
A Qualitative Physics Based on Confluences
AI-Journal 24, 1 - 3 (1984) 7 - 83.

[Doi & Kodama 90] N. Doi, Y. Kodama
An Implementation of an Operating System Kernel using Concurrent Object-Oriented Language ABCL/c+
in "ABCL An Object-Oriented Concurrent System", ed. A. Yonezawa, MIT Press, 1990

[Fishwick & Modjeski 91] Fishwick, P. and Modjeski, R. (eds.)
Knowledge-Based Simulation: Methodology and Application
Springer-Verlag, N. Y. (1991).

[Forbus 84] Forbus, K.
Qualitative Process Theory
AI-Journal 24, 1-3 (1984) 85-168.

[Forgy 1982] Forgy, C.L.
Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem
Artificial Intelligence, 19, (1982) pp.17-37, North-Holland 1982.

[Futo & Gergely 90] Futo, I. and Gergely, T.
Artificial Intelligence in Simulation
Ellis Horwood Ltd., Chichester (1990).

[Giroux et al 90] S. Giroux, A. Senteni, P. Sallé
On part hierarchies and prototypes in an actor language
Proc. ESM 90, Nürnberg, SCS 90, 10.-13. Juni 1990

[Henry et al. 83] J.J. Henry, J.L. Farges, J. Tuffal
The Prodyn Real Time Traffic Algorithm
Proc. 4th IFAC/IFIP/IFORS Conf. on Control in Transportation Systems, Baden-Baden, 1983

[Heuser 84] H. Heuser
Lehrbuch der Analysis
Teubner Verlag, Stuttgart, 1984

[Hewitt 73] C. Hewitt, P. Bishop, R. Steiger
A Universal Modular ACTOR Formalism for Artificial Intelligence
Proc. IJCAI 73, 1973

[Hewitt 77] C. Hewitt
Viewing Control Structures as Patterns of Passing Messages
Artificial Intelligence Vol.8 pp.323-364, 1977

[Hewitt et al 79] C. Hewitt, G. Attardi, H. Lieberman
Specifying and proving properties of guardians for distributed systems
in *Semantics of Concurrent Computation*, Springer-Verlag, Lecture Notes in Computer Science 70, 1979

[Hewitt 80] C. Hewitt
The Apiary network architecture for knowledgeable systems
Proc. Lisp Conference, 1980

[Hewitt & de Jong 83] C. Hewitt, P. de Jong
Analyzing the Roles of Descriptions and Actions in Open Systems
Proc. AAAI 83, 1983

[Hoare 78] C.A.R. Hoare
Communicating Sequential Processes
Communications of the ACM, 1978

[Iwasaki 89] Iwasaki, Y.
Qualitative Physics
in: [Barr et al. 89] Chapter XXI (1989) 323 - 414.

[Kafura 88] D. Kafura
Concurrent Object-Oriented Real-Time Systems Research
Technical Report TR 88-53, Virginia Polytechnic Institute and State University, 1988

[Kafura & Lee 89] D. Kafura, K. H. Lee
Inheritance in Actor Based Concurrent Object-Oriented Languages
Technical Report TR 88-53, Virginia Polytechnic Institute and State University, 1989

[Kafura & Lee 90] D. Kafura, K.H. Lee
ACT++: Building a Concurrent C++ with Actors
JOOP, Vol.3 No.1, März/Juni 1990

[Keene 89] Keene, S. G.
Object-oriented Programming in Common Lisp: A Programmer´s Guide to CLOS
Addison-Wesley, Reading, Mass. (1989).

[König 1990] König, R.
Redesign eines Expertensystems für Konfigurationsaufgaben
Diplomarbeit Nr.643 an der Universität Stuttgart, Januar 1990.

[Kuipers 84] Kuipers, B.
Commonsense Reasoning about Causality: Deriving Behavior from Structure
AI-Journal 24, 1-3 (1984) 169 - 204.

[Kuipers 86] Kuipers, B.
Qualitative Simulation
AI-Journal 29, 3 (1986) 289 - 338.

[Lapalme & Sallé 89] G. Lapalme, P. Sallé
Plasma-II: an Actor Approach to Concurrent Programming
ACM SIGPLAN Vol.24 No. 4, April 1989

[Lapierre & Steierwald 87] Lapierre, R. und Steierwald, G. (Hrsg.)
Verkehrsleittechnik für den Straßenverkehr, Band 1, Grundlagen und Technologien der Verkehrsleittechnik
Springer-Verlag, Berlin (1987).

[Lapierre & Steierwald 88] Lapierre, R. und Steierwald, G. (Hrsg.)
Verkehrsleittechnik für den Straßenverkehr, Band 2, Leittechnik für den innerörtlichen Straßenverkehr
Springer-Verlag, Berlin (1988).

[Lawless & Miller 91] Lawless, J. and Miller, M.
Understanding CLOS: The Common Lisp Object System
Digital Press, Bedford, Mass. (1991).

[Leichter 81] Leichter, K.
Aufbau eines Simulationssystems zur Bewertung einer integrierten Verkehrslenkung in zusammenhängenden Kernstadtnetzen
Schriftenreihe Straßenbau und Straßenverkehrstechnik, Bundesminister für Verkehr, Abt. Straßenbau, Bonn-Bad Godesberg, Heft 322 (1988).

[Leutzbach 88] Leutzbach, W.
Introduction to the Theory of Traffic Flow
Springer-Verlag, Berlin (1988).

[Lighthill & Witham 55] Lighthill, M. and Witham, G.
On Kinematic Waves II. A Theory of Traffic-Flow on Long Crowded Roads
Proc. Roy. Soc. Series A No. 1178, London, Vol. 229 (1955) 317 - 345.

[Manning 87] C. Manning
Traveler: The Apiary Observatory
Proc. ECOOP 87, European Conference on OOP in Lecture Notes in Computer Science Vol.276, pp89-97, Springer Verlag, 1987

[Mattern 89] F. Mattern
Verteilte Basisalgorithmen
Informatik Fachberichte Bd. 226, Springer-Verlag, 1989

[McDermott 1982] McDermott, J.
R1: A Rule-Based Configurer of Computer Systems
Artificial Intelligence, Vol. 19, 1982, pp. 39-88.

[McDermott 1984] McDermott, J.; Bachant, J.
R1 Revisited: Four Years in the Trenches
AI-Magazine 5, 21-32, Fall 1984.

[McKay 91] McKay, S.
CLIM: Common Lisp Interface Manager
CACM 34, 9 (1990) 58 - 59.

[Moreno et al. 90] Moreno, S., Toledo, F., Rosich, F., and Martin, G.
Qualitative Simulation for Temporal Reasoning in Urban Traffic Control
Proc. 10th International Work-shop on Expert Systems and their Applications, Avignon (1990) 97 - 111.

[Nehmer et al 87] J. Nehmer, D. Haban, F. Mattern, D. Wybranietz, D. Rombach
Key Concepts of the INCAS Multicomputer Project
IEEE Transactions on Software Engineering Vol. SE-13, No.8 , August 1987

[Österle 88] Österle, H. (Hrsg.)
Anleitung zu einer praxisorientierten Software-Entwicklungsumgebung, Band 1: Erfolgsfaktoren werkzeugunterstützter Software-Entwicklung
AIT Angewandte Informationstechnik, Hallbergmoos (1988).

[Parnas 72] Parnas, D.
On the Criteria to be used in Decomposing Systems into Modules
CACM 15, 12 (1972) 1053 - 1058.

[Post 1943] Post, E.
Formal Reductions of the General Combinatorial Problem, American Journal of Mathematics 65, 197-268, 1943.

[Puppe & Voß 87] Puppe, F. und Voß, H.
Qualitative Modelle in Wissensbasierten Systemen
in: Christaller, T., Hein, H. und Richter M. (Hrsg.): *Künstliche Intelligenz: Theoretische Grundlagen und Anwendungsfelder, KIFS-85 und KIFS-86, Proceedings*
Informatik-Fachberichte 159, Springer-Verlag, Berlin (1987) 183 - 241.

[Puppe 1988] Puppe, F.
Einführung in Expertensysteme
Springer-Verlag Berlin 1988.

[Puppe 1990] Puppe,F.
Problemlösungsmethoden in Expertensystemen
Springer-Verlag Berlin 1990.

[Reynolds 82] C. W. Reynolds
Computer Animation with Scripts and Actors
ACM Computer Graphics Vol.16 No.3, 1982

[RiLSA 91] RiLSA
Richtlinien für Lichtsignalanlagen RiLSA - Lichtzeichenanlagen für den Straßenverkehr
Forschungsgesellschaft für Straßen- und Verkehrswesen, Köln (1991).

[Saffran 92] Saffran, A.
Intelligente Kreuzung
Studienarbeit, Forschungszentrum für Informatik FZI, Universität Karlsruhe (1992).

[Schönthaler & Németh90] Schönthaler, F. und Németh, T.
Software-Entwicklungswerkzeuge: Methodische Grundlagen
Teubner-Verlag, Stuttgart (1990).

[Senteni et al 89] A. Senteni, P.Sallé, G. Lapalme
Simulation with Actors using Time-Referenced Message-Passing
Proc. ESM 89 , 1989

[Senteni et al 90] A. Senteni, P. Sallé, G. Lapalme
An extension of an actor language towards discrete event simulation
Proc. EMC 89, Advances in AI and Simulation, SCS 90, 1990

[Sevenic 90] Sevenic, S.
DEVS-CLOS: Implementing DEVS Concepts in Common-LISP Object System
Simulation Digest 21, 1 (1990) 14 - 19.

[Shortliffe 1984] Shortliffe, E.; Buchanan, B.; Feigenbaum, E.
Knowledge Engineering for Medical Decision Making: a Review of Computer Based Clinical Decision Aids
in Clancey, W.; Shortliffe, E. (eds.): *Reading Artificial Intelligence,* Chapter 3, Addison-Wesley, 1984 (1979).

[Steele 90] Steele Jr., G. L.
CommonLISP - The Language
Second Edition, Digital Press, Burlington, Mass. (1990).

[Stroustrup 86] B. Stroustrup
The C++ Programming Language
Addison-Wesley, Reading, Massachusetts, 1986

[Struß 89] Struß, P.
Structuring of Models and Reasoning about Quantities in Qualitative Physics
Dissertation, Universität Kaiserslautern (1989).

[Symbolics 90a] Symbolics, Inc.
Book 8 - Symbolics Common Lisp Programming Constructs, Chapter 2: Symbolics CLOS
Symbolics, Inc., Burlington, Mass. (1990).

[Symbolics 90b] Symbolics, Inc.
Book 10 - Programming the User Interface
Symbolics, Inc., Burlington, Mass. (1990).

[Symbolics 90c] Symbolics, Inc.
Book 12 - Program Development Utilities
Symbolics, Inc., Burlington, Mass. (1990).

[Symbolics 91] Symbolics, Inc.
Common Lisp Interface Manager (CLIM): Release 1.0
Symbolics, Inc., Burlington, Mass. (1991).

[Takada & Yonezawa 90] T. Takada, A. Yonezawa
An Implementation of an Object-Oriented Concurrent Programming Language in Distributed Environments
in “ABCL An Object-Oriented Concurrent System”, ed. A. Yonezawa, MIT Press, 1990

[Theriault 83] D.G. Theriault
Issues in the Design and Implementation of ACT2
Technical Report 728, Artificial Intelligence Laboratory, 1983

[Toledo et al. 91] Toledo, F., Moreno, S., Rosich, F., and Martin, G.
Qualitative Simulation in Urban Traffic Control: Implementation of Temporal Features
Internal Paper, Departamento de Informática y Electrónica, Univ. de Valencia, (1991).

[Tomlinson et al 89] C. Tomlinson, W. Kim, M. Scheevel, V. Singh, B. Will, G. Agha
Rosette: An Object-Oriented Concurrent Systems Architecture
ACM SIGPLAN Vol.24 No. 4, April 1989

[[Walker et al. 87] Walker, J., Moon, D., Weinreb, D., and McMahon, M.
The Symbolics Genera programming environment
IEEE Software 4, 6 (1987) 36 - 45.

[Webster & Cobbe 66] F.V. Webster, B.M. Cobbe
Traffic Signals
Road Research Technical Paper, 56, HMSO London, 1966

[Wiedemann 91] Wiedemann, R.
Steuerung des Straßenverkehrs
Vorlesungsskript, Institut für Verkehrswesen, Universität Karlsruhe (1991).

[Wild & Berning 91] Wild, B. and Berning, M.
Intelligent Traffic Control for Urban Networks - Development of AI Concepts for Traffic Management Systems
Annual Report 3/91, FZI, Universität Karlsruhe (1991).

[Wild et al. 92a] Wild, B., Dalchow, O. und Schütze, B.
SAPPORO - Software Documentation
Interner Bericht, FZI, Universität Karlsruhe (1992).

[Wild et al. 92b] Wild, B., Dalchow, O. und Schütze, B.
SAPPORO - User´s Manual
Interner Bericht, Forschungszentrum Informatik FZI, Univ. Karlsruhe (1992).

[Winston & Horn 87] Winston, P. and Horn, B.
LISP
Addison-Wesley, Reading, Mass. (1987).

[Yokote & Tokoro 86] Y. Yokote, M. Tokoro
Concurrent programming in ConcurrentSmalltalk
in “Object-Oriented Cuncurrent Programming”, ed. A. Yonezawa, M. Tokoro, MIT Press, 1986

[Zeigler 76] Zeigler, B. P.
Theory of Modeling and Simulation
Wiley, New York (1976).

[Zeigler 84] Zeigler, B. P.
Multi-facetted Modeling and Discrete Event Simulation
Academic, New York (1984).

[Zeigler 87] Zeigler, B. P.
Hierarchical, modular discrete-event modeling in an object-oriented environment
Simulation 49, 5 (1987) 219 - 230.

[Zeigler 90] Zeigler, B. P.
Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems
Academic Press, Boston (1990).

[Zohar 92] Zohar, R.
Verkehrssteuerung mit Verteilter Künstlicher Intelligenz
Diplomarbeit, Forschungszentrum für Informatik FZI, Universität Karlsruhe (1992).

[Zozaya 1987] Zozaya-Gorostiza, C.; Hendrickson, A.M.
Expert System for Traffic Signal Setting Assistance
Journal of Transportation Engineering, Vol.113, No.2, March 1987.

13 Acknowledgements

We would like to thank all people who have contributed to this report and to the design, implementation and testing of the prototype system Sapporo.

Many thanks to Ron Zohar for his work on DAI architectures for traffic control and to Andrea Jenning-Matalla for the translation of most of the texts into a readable English.

Bernd Wild	
Bernd Schütze	
Olaf Dalchow	
Axel Saffran