Situation Assessment in Traffic Scenarios

Jacek Malec and Per Österling*
Department of Computer and Information Science
Linköping University
S–581 83 Linköping, Sweden
jam@ida.liu.se

Abstract

The problem of assessing situation by an intelligent agent involved in a traffic scenario is discussed. Traffic scenarios are definitely entailed by what is commonly understood as a "real-world activity". The area is perfectly suited for studying situated intelligent activity. It involves mostly routine behaviour of an agent situated in a large space of possible configurations usually involving other agents and subject to hard-real-time constraints. A three-layer architecture for implementing intelligent situated agents is presented.

1 Introduction

The paper addresses the problem of assessing situation by an intelligent agent involved in a traffic scenario. Traffic scenarios are an interesting application domain because they are definitely entailed by what is commonly understood as a "real-world activity". Moreover, this area is perfectly suited for studying situated intelligent activity. The static part of the traffic world (road network, rules of behaviour in traffic, traffic signs) is relatively well structured and easily comprehensible. Even the dynamic part of it, i.e. the possible behaviour of all the agents involved in a traffic scenario, is constrained in many ways, so that it is more predictable than some other application areas of intelligent systems. Nevertheless, so far there exists no artificial system able to effectively cope with all the complexity of the traffic world.

One of the major reasons is that problems of hard real time are of crucial importance here. The agent is required to find an adequate activity in any traffic situation within the time limits defined by the situation itself (and possibly some other factors). Therefore so called anytime algorithms are inevitable solution in this context. However, we don't know yet too much about this kind of algorithms to be able to propose any consistent methodology of designing intelligent systems efficiently coping with all the road traffic situations.

An agent acting in such domain has to be highly reactive. This observation strongly favours the conjecture that (at least in this type of activity) intelligence emerges from the interaction between a (possibly very simple) behaviour of an agent and the complex, dynamic, unpredictable behaviour of its environment (cf. [2]). However, an insight into the human problem solving in this kind of scenarios would rather suggest that mental states of an agent play an important role in its situated activity.

*This research has been supported by the Center for Industrial Information Technology (CENIIT).

The central issue for intelligent activity in traffic scenarios is classification of the current situation. It can be seen as a continuous pattern matching of the stream of input data with agent's:

- knowledge,
- beliefs,
- goals,
- intentions,
- resources.

A very interesting aspect of this classification is illustrated by human ability to change the time scale in the cases when time is the most critical resource. Slowing down translates a traffic situation or scenario into some other (and usually safer) area of the space of possible classifications and resulting activities.

A solution emerging from our research on these topics is a generic three layered software architecture suitable to host implementations of various forms of intelligent activity, including purely reactive, purely deliberative, and mixed ones [13, 11, 6]. We are also doing research on mental activities underlying goal-driven reactive behaviour [18]. Another interesting issue addressed in our group, namely the problem of designing provably correct agents (and, more specificly, provably correct symbolic classifiers), is very important in the context of traffic applications [20].

The paper consists of four sections. The first one discusses the domain of traffic scenarios as an important area of intelligent activity. The next one addresses the problem of situation assessment: whether it is just a classification of raw input data, or something more. The third section is devoted to a presentation of the three-layered architecture developed at our laboratory. Finally, the paper ends with some comments regarding efforts towards designing purely behaviour-based control for autonomous agents. This approach is yet(?) unsuccessful and this failure is attributed to the lack of explicit mental notions (intentions, goals, beliefs) in the design phase for those kinds of systems (with the eminent exception in the work by Rosenschein and Kaelbling [7]).

2 The traffic domain

Traffic domain is an interesting application area because on one hand it is definitely entailed by what is commonly understood as a "complex real-world activity", but on the other hand it is relatively well structured and easily comprehensible. The static part of the traffic world (road network, rules of behaviour in traffic, traffic signs) is either codified or subject to a set of well-defined rules. Even the dynamic part of it, i.e. the possible behaviour of all the agents involved in a traffic scenario, is constrained in many ways, so that it is more predictable than some other application areas of intelligent systems. For example, the agent's dynamics is limited, most of the traffic participants' actions obey the traffic code, most of the actions are planned in some advance and other road users are informed about them in some way. Nevertheless, so far there exists no artificial system able to effectively cope with all the complexity of the traffic world.

One of the major reasons for this fact is that hard-real-time constraints are of crucial importance here. The agent is required to find an adequate activity in *any* traffic situation within the time limits defined by the situation itself (and possibly some other factors). Therefore so called anytime algorithms (see e.g. [3]) are inevitable solution in this context. However, not too much is known about this kind of algorithms to let one propose any consistent methodology of designing intelligent systems efficiently coping with all the road traffic situations.

Therefore most of the research done so far is based on the observation that an agent acting in such domain has to be highly reactive. This observation strongly favours the conjecture that (at least in this type of activity) intelligence emerges from the interaction between a (possibly very simple) behaviour of an agent and the complex, dynamic, unpredictable behaviour of its environment (cf. [2]). There exist several systems successfully employing the reactive behaviour paradigm to achieve high competence in some aspects of traffic behaviour (e.g. motorway driving including merging and overtaking, intersection crossing, target following, etc.). Most of those systems consist of a relatively simple discrete controller representing the "mode" the system is in, and a set of independent, low-level, continuous controllers coupling sensors to actuators and implementing basic behaviours. The efforts towards creating systems which could plan (or just perform non-trivial symbolic reasoning) while executing their situated actions have not yet given the expected results.

However, an insight into the human problem solving in this kind of scenarios would rather suggest that mental states of an agent play an important role in its situated activity. This is an interesting observation, quite in contrast to the fact that the only traffic activities requiring actual pre-planning are related to path determination, ranging from the journey level (planning a trip from Linköping to Rome) down to the maneuver planning (planning how to get into a parking slot). This contradiction can probably be explained by the fact that human beings tend to rationalize their activities, even the very reflexive ones. On the other hand, most of the reactive behaviours of an intelligent traffic agent should obviously be context (i.e. situation) dependent. Therefore in our research we assume the expanded architecture of a hybrid system: a discrete state transition system, where each state corresponds to some low-level, reactive behaviour (mode), and where the transition system itself can be modified by the reasoning layer supervising the agents activities.

3 The assessment of a traffic situation

The central issue for intelligent activity in traffic scenarios is classification of the current situation, i.e. recognition of the mode the system is in, and the actual state within the mode. This process can be seen as a continuous pattern matching of the stream of input data with agent's knowledge, beliefs, goals, intentions, and resources. This point of view is quite common, see e.g. [21]

So far we represent all the mental state aspects of an agent using the discrete states of a state transition system. This does not pose any problems: even quite complicated scenarios, like overtaking or intersection passing, could be squeezed into the framework of such reporesentations [15, 8]. Below we describe[1] the way the mode and state recognition is achieved in the actual implementation of an intelligent traffic agent: a driver support system [11, 16].

[1] This section has originally appeared in [11].

3.1 Understanding the traffic scenario

Reasoning in a continuous, dynamic and complex environment can soon become intractable if each individual change in the environment is taken account of. One way of getting around the complexity is to make simplifying assumptions about the environment. Another approach avoids restricting the application domain but attempts to extract significant events from the raw sensory information prior to the reasoning stage, and to provide each type of reasoning with suitable data. We will adopt the latter approach and minimize the restrictions put on the environment and on the behaviours of cars and drivers.

To be able to arrive at support decisions at all, the support system would need to "understand" the significant characteristics of the traffic situation. This understanding must be based on available sensory information and has to be built and maintained continuously during the ongoing scenario. In a driver support system (DSS) realized by a layered software architecture, each layer must base its decisions on its current interpretation of the scenario. Although the layers need models of the environment, each layer has different demands on the level of abstraction and the temporal resolution of its model. This leaves us with two possibilities. The first is to have one comprehensive model of the environment, complex enough to meet the requirements of all layers. The second approach, which we have adopted, is to have a multitude of coherent models, ranging from numeric to symbolic, each intended to fulfill the requirements of a particular layer. No part of the supportive autonomous agent has access to all models. Nevertheless, these models jointly constitute the agent's overall understanding of the surrounding environment. Moreover, the requirement put on the generated response to satisfy the time constraints of the application necessitates explicit modelling of the temporal aspects.

As an example of how models can be built from a continuous flow of sensory data via subsequent abstractions, let us consider the following. Some physical properties, such as positions, are directly observable (via sensors) in the environment. These properties constitute the "raw" quantitative model. One can then identify a class of derived conceptual properties which are *based* on a number of observable (and possibly other derived) properties, although not necessary on a momentary basis. For example, a "closeness" property could be based on positions and velocities of two cars being considered as potentially "close". Yet another class of properties, referred to as *trends*, monitors the development of derived properties over time (e.g. "getting closer"). Finally, a vital change in a trend may constitute a *significant event* which in turn can lead to the update of a qualitative model. By defining significant events in terms of trends, we are able to restrict the number of considered occurrences.

To cope with the complexity of a dynamic environment, we identify the need to restrict the focus of attention to a number of objects in the nearest environment of the own car. Obviously, some of the surrounding cars are more interesting for us than others. In particular, we see that both the degree of importance and the interesting properties of a specific car are determined by this car's position relative to the own car. This interest is independent of the identities of the cars occupying the relative positions (i.e. regardless of whether a car is actually a blue Volvo or a red Saab). It is therefore advantageous to be able to refer to the cars symbolically through their location relative to the own car rather than through their actual identities. Such references, known as *deictic* references [1], are convenient means for maintaining references to objects in a dynamic environment. Moreover, to facilitate reasoning about the environment at a higher level of abstraction while still being able to refer to each individual explicitly, the deictic references can be grouped into aggregates of related objects.

As an example, references to all relevant cars in the same lane as the own car can be grouped together and then be treated as a whole. Hence, one of the main tasks of the process of building a model of the environment would be to classify and "name" all the cars according to those relations.

To conclude, we need a multitude of models "large" enough to enable the support system reach the desired response, but "small" enough to make real-time reasoning computationally tractable.

3.2 The characterizer

The task of extracting significant events in a dynamic environment is the objective of a dynamical classification component referred to as the *characterizer*. The characterizer resides in the process layer of our architecture (cf. section 4).

The approach is to transform "raw" continuous-valued input to a discrete-valued output, where both input and output change over a continuous (or evenly sampled) time. The characterizer arrives at a model of the environment by a series of abstractions ultimately resulting in the recognition of significant events. The abstractions and the transformations from continuous to discrete typically depend on the past history.

The choice of properties and the nature of snapshots are of course dependent on the type of support to be provided. Different types of support require recognition of different types of situations, where each situation can be described in terms of a finite set of properties.

3.2.1 Sensory information

From our standpoint, the output of the sensory systems is our input, independently of the actual sensory devices used. We can imagine using processed vision information, active inter–vehicle and vehicle–roadside communication, or any other sensory information about the cars and the environment. Accordingly, the characterizer receives as its input a (compound) piecewise continuous fluent (signal) from a number of sensors. This fluent includes measurements for a number of cars which happen to be in the range of the sensors. It is assumed that the resulting fluent consists of successive positions of the recognized cars, together with some information about their current signalling status.

3.2.2 Positions

The position of a car is defined using so-called *road segment coordinates* i.e. relative to the current road segment[2], which is an element of a road-segment-based map of the road network. The reason for choosing road segment coordinates instead of cartesian is that the former correspond in a more natural way to the traffic behaviour; that is, cars normally follow a lane, possibly with some aberrations from its center line (axis). As a separate project within our Prometheus group, a database management system for road segments has been developed [4]. The system provides an implementation of a map consisting of road segments, as well as transformation routines from cartesian coordinates to road segment coordinates and vice versa.

[2] A *road segment* can be understood as a homogeneous element of the road network, e.g. a curve, or a straight road element with a constant speed limit would be coded as separate road segments.

3.2.3 Configuration determination

In order to reason about the environment at a higher level of abstraction while still being able to refer to each individual explicitly, deictic referrals to the surrounding cars are grouped into aggregates called *configurations*. Therefore, a configuration consists of agents or objects which are related to each other in specific way over a (possibly limited) period of time. Configurations allow us to divide the objects in the environment into two classes: the *In* class consisting of the objects belonging to the configuration, and the *Out* class consisting of the remaining objects. This classification suggests a policy of using of using the sensing capabilities: concentrate on the *In* objects and check sporadically whether any of the *Out* objects is going to become an *In*. This policy silently assumes that the *In*/*Out* classification reflects the relevance of the objects in the environment. This is obviously dependent on the characterization algorithm and on the definition of configuration.

We distinguish the *Own* and the *Meeting* configurations which are always present. In addition, *Merging* and *Crossing* configurations are defined in the vicinity of an intersection.

Configurations are determined as follows. First an *object characterizer* determines individual properties of each car detected by the sensors in the environment. One such individual property can be the expected turning intention. Another example is the expected path — a sequence of road segments — which is likely to be traversed next. Recognition of both of these properties relies on the current status of the indicators, the current position, and the past history of the car.

Then a *relation characterizer* uses the individual paths of cars to determine which cars are in given binary relations with the own car. These relations are of the type "following", "meeting", and so on. Thus, the relation characterizer recognizes cars which are the candidates for playing some particular role in the current configuration .

In the final stage of the configuration determination the *configuration characterizer* checks which cars in the set of candidates are close enough for being considered as belonging to a particular configuration.

3.2.4 Properties and trends of individual cars

Next the characterizer determines properties of individual cars and trends associated with those properties. The reason for not determining them already in the object characterizer above is that they are dependent of how the cars are related to each other and, in particular, how they are related to the own car.

3.2.5 Properties and trends of configurations

The next stage of characterization consists of determining properties assigned to the environment on the basis of the properties of individual cars. Since the degree of being free ("freeness") of the road in either lane is a useful property for e.g. overtaking support, we specify "freeness" of each lane as a function of time with the range of values *free, soon-blocked, blocked* and *soon-free*. To define the concept of "freeness" we use the results of previous characterization phases, i.e. the current configuration, the properties of individual car, and the recognized trends. For example, the *blocked* property of the left (meeting) lane depends on existence, properties and trends of the cars in the meeting lane, the expected path of the cars in the right lane, and so on.

3.2.6 Significant events

The information relevant to the decisions to be made by the DSS (whether and how to act) is sent to the event-driven rule layer of our system. Each change of the degree of freeness from one value to another should be recognized as a significant event by the characterizer. Examples of significant events used in safety determination are: lane changes by the own car, changes in the degree of freeness of each lane, changes of speed limit, and changes of configurations. The intention of the own driver is either recognized from the actually observed driver's manoeuvres or communicated directly by the driver. The output of the characterizer is the a description of the whole traffic scenario in terms of a sequence of significant events.

4 A three-layered architecture

A solution emerging from our research on these topics is a generic three layered software architecture suitable to host implementations of various forms of intelligent activity, including purely reactive, purely deliberative, and mixed ones [13, 11, 6]. This architecture (see Figure 1) has been developed in our group [5, 17, 14, 13] in the context of PROMETHEUS project. The three layers are separated on the basis of the type of computations they are intended to perform.

4.1 Layers

The first layer, called the *process layer*, is the place of implementing numerical, periodic tasks, such as preprocessing sensory data or controlling actuators. Our model of computations assumes this layer to be synchronous, although periods of execution of various tasks may differ. Tasks can communicate with each other either via the *state vector* or directly with each other. The state vector is also used to store preprocessed sensory information and computed control values for actuators. According to the AI naming conventions we will call the signals processed by the process layer *periodic fluents* representing the *piecewise continuous fluents* observed at the inputs and sent as the outputs of the system.

The middle layer is called the *discrete response layer* (or in some of our previous publications the *rule layer*) and is intended to perform tasks which are by their nature asynchronous, or in other words, data driven. An example of such task is the change of control mode (of the process layer) due to the change of situation on the road; e.g. car catching up the car in front, or car entering a motorway from a curving land road.

The computational model assumed for discrete response layer is that of discrete event systems (DES). There exist several equivalent DES formalisms: automata, transition systems, rule-based systems, etc. All of them distinguish the notions of *state* and *transition* as central, although the details vary from model to model. We have adopted the rule-based approach as the most convenient for the purpose of specification of a knowledge-based DES. This does not preclude usage of other approaches, what has been shown in [9, 10, 19].

The signals processed by this layer are called *piecewise constant fluents*.

The last layer is called the *analysis layer* because it is intended to handle *symbolic* reasoning tasks such as prediction, planning and replanning. As *limited prediction* does not require the full reasoning capabilities of the system, it might be distinguished as a sub-layer interfacing the discrete response layer. However, it still needs the ability to reason *about* time flow and its effects and therefore has to be included in the highest layer of the architecture.

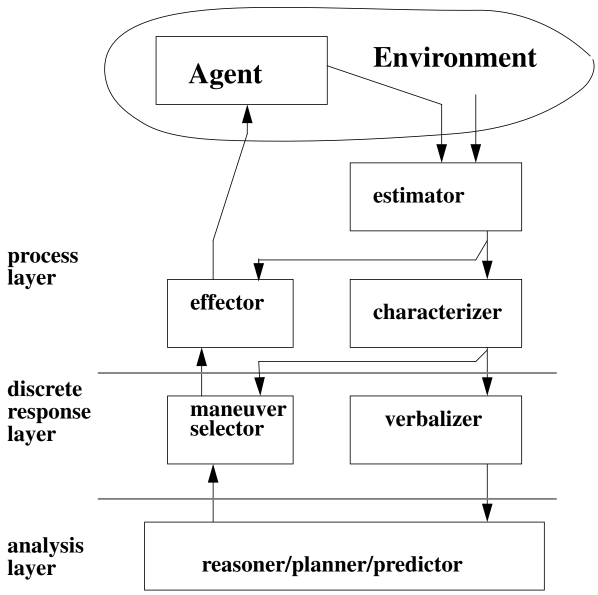

Figure 1: A three-layer architecture of an autonomous system.

This layer is expected to produce controlling sequences for the rule layer. This layer would also host eventual learning mechanisms which might adapt the discrete response layer while maintaining its critical response requirements.

4.2 Real-time software tools

During our previous research done within the Prometheus programme we have thoroughly studied this architecture and its implications on software engineering issues [12, 13]. One of the conclusions was that it facilitates prototyping of systems, especially because it allows development of generic software tools which can be used for implementation of each particular application system. Along this line we have developed software kernels, or *engines*, for development and implementation of the process layer and the discrete response layer. The set of tools includes:

- Process Layer eXecutive (PLX): a multi-threaded time-triggered real-time engine for implementation of process layer software. It has been implemented on a VME-bus system equipped with the pSOS$^+$ real-time operating system, and on a PC-compatible micro with VDX operating system. There also exists a simulator of the PLX which runs under Unix;
- Process Layer Configuration Language (PLCL) and its compiler: a language for specification of process layer module interconnections and interfaces to both sensors and actuators on one side, and to the rule layer software on the other side. The modules themselves are programmed in some conventional language, such as C or C++;
- Rule Layer eXecutive (RLX): an engine for implementation of rule-based discrete-event systems. It has been implemented as a LISP program on a Unix workstation and, recently, on a PC with VDX;

- Rule Layer Configuration Language: a logic-based rule language for declarative description of discrete-event control imposed on process layer by the discrete response layer.

5 Concluding remarks

An agent acting in the traffic domain has to be highly reactive. This observation strongly favours the conjecture that (at least in this type of activity) intelligence emerges from the interaction between a (possibly very simple) behaviour of an agent and the complex, dynamic, unpredictable behaviour of its environment (cf. [2]). The "behaviour-based" line of research on autonomous agents, although successful in the case of simple, insect-like robots, is yet to show its potential for the case of more complex systems. This failure is sometimes attributed to the lack of explicit mental notions (intentions, goals, beliefs) in the design phase for those kinds of systems (with the eminent exception in the work by Rosenschein and Kaelbling [7]). However, a reactive agent needs not to be deprived of its mental activities. We are doing some research on mental activities underlying goal-driven reactive behaviour [18].

A very interesting aspect of the situation characterization process is illustrated by the human ability to change the time scale in the cases when time is the most critical resource. Slowing down translates a traffic scenario into some other (usually safer) area of the space of possible classifications and resulting activities.

Another interesting issue addressed by our group, namely the problem of designing provably correct agents (more specificly, provably correct symbolic classifiers), is very important in the context of traffic applications [20]. We are aiming at a methodology allowing the designer of an intelligent system to specify the set of posible behaviours of the system and of its environment, and then to prove that some particular implementation will satisfy some reasonable safety conditions. Some interesting results will appear in Nadjm-Tehrani's forthcoming Ph.D. Thesis.

The conclusion of the paper is that situation assessment of a traffic scenario can be reduced to a complex classification based on input data and current context (state) of the intelligent agent, where the context needs not to be necessarily explicit (i.e. symbolic). We have proposed a software engineering methodology for designing intelligent autonomous agents based on experience and results from both artificial intelligence and systems engineering. This methodology, and the associated design and software tools (languages and engines) are intended to make this task simpler and more manageable.

Acknowledgement

Most of the research presented in this paper has been done in collaboration with, or simply by, other members of the Autonomous Systems Group of the Laboratory for Representation of Knowledge in Logic of our department. The group consists currently of: Magnus Morin, Simin Nadjm-Tehrani, Per Österling, and the author. Erik Sandewall has been the source of many of the ideas developed further by the group members.

References

[1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In

Proceedings AAAI-87: Sixth National Conference on Artificial Intelligence, pages 268–272. Morgan Kaufman, 1987.

[2] Rodney A. Brooks. Intelligence without reason. In *Proceedings of the Twelvth International Joint Conference on Artificial Intelligence, Sydney.* Morgan Kaufman, 1991.

[3] Mark Drummond and John Bresina. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proceedings AAAI-90: Eighth National Conference on Artificial Intelligence*, pages 138–144. Morgan Kaufman, 1990.

[4] Reza Farahani. A Road Segment Database Management System and Simulation Tools for a Map-based Copilot. In *Proceedings of ProArt Workshop on Intelligent Co-pilot*, Grenoble, France, December 1991.

[5] Johan Hultman, Anders Nyberg, and Mikael Svensson. A software architecture for autonomous systems. In *Proc. 6th International Symposium on Unmanned Untethered Submersible Technology*, pages 279–292, 1989.

[6] IEEE Industrial Electronics Society. *Intelligent Vehicles'93 Symposium*, Tokyo, Japan, July 1993.

[7] Leslie Pack Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6:35–48, 1990.

[8] Jacek Malec. How to pass an intersection. In *Proc. of the Prometheus Pro-Art Workshop on Intelligent Co-Pilot*, pages 167–182, Grenoble, France, 1991.

[9] Jacek Malec. Applied knowledge representation. *CC-AI: The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology*, 9(1):9–41, 1992.

[10] Jacek Malec. Complex behavior specification for autonomous systems. In *Proc. IEEE International Symposium on Intelligent Control'92*, pages 178–183, Glasgow, Scotland, August 1992.

[11] Jacek Malec and Per Österling. Driver support system for traffic manoeuvers. In G. Ambrosino, M. Bielli, and M. Boero, editors, *Artificial Intelligence in Traffic Engineering.* VSP International Science Publishers, Zeist, The Netherlands, 1994. Available also as Technical Report LiTH-IDA-R-93-21, Department of Computer and Information Sciences, Linköping University.

[12] Magnus Morin. Predictable cyclic computations in autonomous systems: A computational model and implementation. Licenciate thesis 352, Department of Computer and Information Sciences, Linköping University, 1993.

[13] Magnus Morin, Simin Nadjm-Tehrani, Per Österling, and Erik Sandewall. Real-time hierarchical control. *IEEE Software*, 9(5):51–57, September 1992.

[14] Magnus Morin and Erik Sandewall. A software architecture supporting integration of subsystems. In *PROMETHEUS: Proceedings of the Fourth Workshop*, 1990. Also available as Technical Report LAIC-IDA-90-TR21, Dept. of Computer and Information Science, Linköping University.

[15] Simin Nadjm-Tehrani. Analysis of the overtaking scenario: Specification of an autonomous car and a driver support system. Technical Report LAIC-IDA-90-TR31, IDA, 1991. Superseded by LAIC-IDA-91-TR1.

[16] Per Österling. Symbolic modelling of the dynamic environments of autonomous agents. Licenciate thesis 386, Department of Computer and Information Sciences, Linköping University, 1993.

[17] Erik Sandewall. System Block Specifications and their PROMETHEUS applications. In *Prometheus: Proceedings of the Fourth Workshop*, 1990. Available also as LAIC-IDA-90-TR22.

[18] Hua Shu. A logical approach to planning in dynamical systems. In *Proceedings of 2nd European Workshop on Planning*, Vadstena, Sweden, December 1993.

[19] Hua Shu and Jacek Malec. From Process Transition Networks to Behavior Automata. In *Proceedings of the 1991 IEEE International Symposium on Intelligent Control, Arlington, VA*, pages 257–262. IEEE Control Systems Society, 1991.

[20] J. E. Strömberg and S. Nadjm-Tehrani. Towards Reliable Models of Hybrid Systems. Research Report LiTH-IDA-R-93-48, Department of Computer Science, Linköping University, 1993.

[21] A. Yavnai. Context recognition and situation assessment in intelligent autonomous systems. In *Proc. 1993 IEEE Int. Symp. Intelligent Control*, pages 394–399, Chicago, IL, August 1993.