PERGAMON

Transportation Research Part C 10 (2002) 373–398

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

Using BDI agents to improve driver modelling in a commuter scenario

Rosaldo J.F. Rossetti [a,*], Rafael H. Bordini [a,1], Ana L.C. Bazzan [a], Sergio Bampi [a], Ronghui Liu [b], Dirck Van Vliet [b]

[a] *Instituto de Informática, Universidade Federal do Rio Grande do Sul, Av. Bento Gonçalves 9500, CP 15064, 91501-970, Porto Alegre-RS, Brazil*

[b] *Institute for Transport Studies, University of Leeds, Leeds LS2 9JT, UK*

Abstract

The use of multi-agent systems to model and to simulate real systems consisting of intelligent entities capable of autonomously co-operating with each other has emerged as an important field of research. This has been applied to a variety of areas, such as social sciences, engineering, and mathematical and physical theories. In this work, we address the complex task of modelling drivers' behaviour through the use of agent-based techniques. Contemporary traffic systems have experienced considerable changes in the last few years, and the rapid growth of urban areas has challenged scientific and technical communities. Influencing drivers' behaviour appears as an alternative to traditional approaches to cope with the potential problem of traffic congestion, such as the physical modification of road infrastructures and the improvement of control systems. It arises as one of the underlying ideas of intelligent transportation systems. In order to offer a good means to evaluate the impact that exogenous information may exert on drivers' decision making, we propose an extension to an existing microscopic simulation model called **D**ynamic **R**oute **A**ssignment **C**ombining **U**ser **L**earning and microsimul**A**tion (DRACULA). In this extension, the traffic domain is viewed as a multi-agent world and drivers are endowed with mental attitudes, which allow rational decisions about route choice and departure time. This work is divided into two main parts. The first part describes the original DRACULA framework and the extension proposed to support our agent-based traffic model. The second part is concerned with the reasoning mechanism of drivers modelled by means of a **B**eliefs, **D**esires, and **I**ntentions (BDI) architecture. In this part, we use AgentSpeak(L) to specify commuter scenarios and special emphasis is given to departure time and route choices. This paper contributes in that respect by showing a practical way of representing and assessing drivers' behaviour and the adequacy

* Corresponding author.

E-mail addresses: rossetti@inf.ufrgs.br (R.J.F. Rossetti), bordini@inf.ufrgs.br (R.H. Bordini), bazzan@inf.ufrgs.br (A.L.C. Bazzan), bampi@inf.ufrgs.br (S. Bampi), rliu@its.leeds.ac.uk (R. Liu), dvan@its.leeds.ac.uk (D.V. Vliet).

[1] Current address: Department of Computer Science, University of Liverpool, UK.

0968-090X/02/$ - see front matter © 2002 Elsevier Science Ltd. All rights reserved.
PII: S0968-090X(02)00027-X

of using AgentSpeak(L) as a modelling language, as it provides clear and elegant specifications of BDI agents.
© 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Multi-agent systems; BDI agents; Driver decision making; Variable demand; Intelligent transportation systems; Traveller information systems; Microscopic traffic simulation

1. Introduction

The rapid growth of urban areas has necessitated special attention from the scientific and technical community over the last few years. Not surprisingly, transportation and traffic systems are objects of concern as they play an important and indispensable role in society. However, today's road infrastructures are insufficient to meet the increasing demand and traffic congestion is frequently encountered in most commuters' journeys. This implies considerable economic, social, and environmental losses, which must be minimised. In this sense, new management policies and planning strategies are required in order to tackle the problems that arise from today's urban scenarios. Physical modification to the road infrastructure is no longer the best alternative. Besides the high cost of implementation, it causes disruptions and damages the environment. Alternatively, some efforts have been identified in order to increase road capacity by improving the efficiency of traffic control systems. Although such efforts have contributed to the reduction of problems related to traffic jams, they cannot be considered a lasting solution.

In this way, current research still seeks alternative means to cope with traffic and transportation problems. First attempts to improve road capacity relied on dealing with the static part of the system, namely the road infrastructure and control system. As alternatives, different approaches have been experienced, which, on the other hand, rely on maximising the use of the present road capacity by directly influencing users' behavioural patterns. The contribution of intelligent transportation systems (ITS) has been increasingly significant in this view. Recent advances in communication and in computer technologies have encouraged the use of such systems as a means to tackle problems in the field of traffic and transportation engineering. One important goal under the ITS umbrella is to ensure productivity and efficiency by making better use of existing transportation systems. It is mainly concerned with the application of distributed solutions aimed at specific issues of users' needs on an individual basis (Chatterjee et al., 1999). Thus, a complex communication and on-line computing infrastructure is the instrument leading to ITS applications, in which traffic improvement results from directly influencing the real-time behaviour of drivers. Integrating all factors, both dynamic and static, which can somehow affect the traffic flow, is central to ITS. In this scenario, all components are expected to work together on a co-operative basis, seeking to maximise the efficiency of the system.

As the driver has become an important facet of modern traffic systems, models used to represent such a domain need to consider the uncertainty inherent in human behaviour, which increases the modelling complexity. Owing to the use of simplified approaches, some traditional models have failed to represent such complex scenarios (Watling, 1994). This has motivated much research and efforts have been carried out in order to adapt these models to meet ITS requirements. More recently, new-generation traffic network models have also emerged from scratch,

attempting to explicitly incorporate better representations of driver behaviour (Cantarella and Cascetta, 1995; Liu et al., 1995; Mahmassani and Jayakrishnan, 1991). In this sense, agent-based techniques seem to be a very appropriate approach to model such a domain.

Multi-agent system (MAS) is a sub-field of the distributed artificial intelligence (DAI), which has received an increasing interest in the last few years. The rapid evolution in the available computational resources, both in hardware and in software, which support a widely physically distributed computing environment, has contributed to this trend. The increasing demand for suitable tools to represent the complexity inherent in some application domains is a factor that also inspires developments in the MAS field. In this work, we aim at exploring the potentials offered by the reasoning and autonomous characteristics of agent-based techniques to represent the uncertainty of human behaviour. In our representation of driver behaviour, we also consider the possible influence that ITS technologies may exert. We base our assumption on a BDI architecture (Rao and Georgeff, 1991) to represent driver reasoning mechanism and the ability to acquire information provided by exogenous sources, both through communication capabilities and through simple sensing of the environment. The two important parts of this work comprise the proposal of an extension to the original structure of **D**ynamic **R**oute **A**ssignment **C**ombining **U**ser **L**earning and microsimul**A**tion (DRACULA), and the use of AgentSpeak(L) as an specification language to model the driver reasoning mechanism. The model presented is particularly concerned with the departure time and route choice processes in a commuter scenario.

2. ITS from a multi-agent system perspective

The concept of a multi-agent system is a modelling approach devised to represent systems whose entities, coined agents, exhibit intelligence, autonomy, and some degree of interaction, both with one another and with the environment. The abstraction approach of MAS consists of representing a system by multiple agents that exist in a common environment and interact in order to achieve specific goals. An agent must be capable of perceiving facts through sensors and acting upon the environment through effectors. Some degree of interactions will also imply the presence of communication capabilities.

Basically, two extreme kinds of agent structures can be identified: the reactive and the cognitive agents. The former is based on a simple approach of mapping perceptions to actions whereas the latter could be fully endowed with reasoning capabilities. Nonetheless, depending on the application, an agent's structure can range from pure reactive to pure cognitive, mostly exhibiting both characteristics to some extent. With regard to cognitive structures, an agent could possess knowledge about its internal state, about the dynamics of the world, i.e., how the world evolves, the likely outcomes of its actions, and some definition of utility. This set of information is the basis for implementing a decision-making mechanism (Russell and Norvig, 1995). Also, MAS has the ability to represent mental attitudes, such as beliefs, desires, and intentions, which are intrinsic in the human reasoning process. In terms of spatial organisation, agent-based models offer good representation of entities that are geographically and functionally distributed. These characteristics ensure agent-based models' scalability and robustness, which are important to ITS applications.

Some examples of applied MAS in the field of traffic and transportation engineering can be found in the literature. However, most of the applications have been concerned with the control

system, not the driver. The basic assumption in such examples relies on the representation of adaptable control system as a community of controller agents, which co-operate in order to achieve an optimum plan to meet the variable demand (Bazzan, 1997; Pires et al., 1997; Roozemond, 1999). The movement is represented on the basis of simplified car-following and lane-changing models that, in the great majority, adopt a simple approach of using a reactive structure. Some attempts at modelling interactions between drivers and service providers, where communication mechanisms are of huge importance, are also identified (Burmeister et al., 1997; Haugeneder and Steiner, 1993). Nonetheless, modelling drivers endowed with mental attitudes has already been presented as a potential application field (Bazzan et al., 1999; Rossetti et al., 2000). In a broad perspective and envisaging the benefits that could arise from the use of such technologies, Bouchefra et al. (1995) suggested a methodology to represent intelligent vehicle highway systems (IVHS) as a multi-agent system using different levels of granularity.

When considering a modelling approach to represent decision-making mechanisms in traffic systems, the driver definitely will be the main subject of concern. Thus, modelling such an entity deserves special attention. Most of the components, e.g., basic traffic control systems, could be modelled as reactive agents, as their tasks are very specific and their reactions to events are well defined. However, this approach may be too simplified to model some features of drivers' behaviour. When answering to control systems or responding to some stimuli brought about by, for instance, the presence of surrounding vehicles, drivers' behaviour is basically reactive. The car-following and lane-changing models are traditional representations of such a reactive behaviour, as they use defined rules to map actions to specific events, such as the red light of traffic signals. However, let us consider the following situation in daily life of a traffic network user, say *agent A*.

On a normal working day, agent A is about to leave home for another home-work journey. Albeit A is familiar with the streets to *his* work, *he* decides to check the traffic conditions before starting the trip in order to avoid a possible traffic jam during the morning peak-hour. Thus, agent A logs onto the Internet and accesses a traffic information system website. Knowing where to get to and estimating the time he will need to perform the journey, A can plan his trip. Thus, he selects a course of actions that will result in reaching his place of work. He chooses a route to follow and a time to leave so that he can arrive by a desired arrival time. Once he has planned his trip, he can execute it. While A has not found any obstacle within the journey, he can keep executing his original plan. However, he has just found that certain road on his route is interrupted. As A is not able to drive through that road anymore, he has to reconsider his options and find another alternative route to get to his destination. Therefore, A abandons his original plan and starts executing a new one. In another point in time, A just realises that he has not got enough petrol to complete his trip. Albeit he knows he will probably be late, he prioritises changing his plans once again in order to stop somewhere to get more petrol. After doing so, he is finally able to arrive at work.

Analysing the example above, it is possible to identify some interesting characteristics in agent A's behaviour, which we describe next.

- autonomy—agent A could identify what his objectives were and which actions he needed to carry out to yield the expecting results;
- social ability—as in the above example, agent A could ask for some help in order to ease the execution of his actions, for instance, by contacting a service provider such as a traveller infor-

mation centre. Another aspect of social behaviour is the necessity of cohabiting with other drivers and of respecting traffic rules to avoid accidents;

- reactivity—responding to traffic signals and breaking in order to avoid colliding with others are some well known examples of reactive behaviour. Albeit perceiving an interruption and adopting another route could be modelled as reactions from the driver, deciding whether to divert may also imply more sophisticated inference mechanisms.
- adaptability—agent A is adaptable in the sense that he may reconsider his options and adopt another strategy in order to accomplish his goals, in the case the original plan becomes inadequate;
- pro-activity—agent A also must be able to prioritise the execution of an action to the detriment of his original plans, for instance, arriving later at work after adopting another route that is more convenient owing to some other reasons.

Although these characteristics are present in drivers' behaviour, it is not easy to represent such assumptions in existing models. In order to capture those features into the driver representation, we adopt a **B**eliefs, **D**esires, and **I**ntentions (BDI) architecture to model the reasoning mechanism. All agents in the model will interact with each other and with the environment in order to improve the system performance. Despite living in the same environment, a driver does not depend directly on others to carry out its own tasks and to seek its goals. Nevertheless, collaboration and co-operation are also present in our drivers' behaviour model, which can be easily identified in the interaction among drivers and service providers, such as dynamic route guidance systems (DRGS) and park guidance systems (PGS).

The interaction between agents and between an agent and the environment plays a relevant role in the system. It is essentially necessary as a mechanism for exchanging information. Besides the built-in knowledge of drivers, they can acquire information by accessing, for example, variable message signs (VMS), and also by observing the environment in previous journeys, in terms of the travel time experienced. Another important issue is the time-dependent nature of such interactions. This characteristic becomes more evident and significant in ITS technologies. They must rely on efficient and trustworthy means of communication in order to deliver information in a timely basis. As consequence of failure in an advanced traveller information system (ATIS), for instance, drivers would experience increases in travel time by virtue of an unadvised traffic jam.

3. An extension to the DRACULA framework

3.1. The original DRACULA framework

The basic structure of some microscopic models, such as DRACULA developed at the Institute for Transport Studies, University of Leeds, UK (Liu et al., 1995), takes into account two concepts that are of central importance: the within-day decision-making process and the day-to-day dynamics. The former focuses on the travel choices made by an individual to perform a journey. Travel goals, travel needs, perceptions, behavioural pattern, and cognitive abilities that influence the choice processes result from the state of those variables at the instant the choice is being undertaken. The dynamic formulation, on the other hand, is concerned with modelling how the

state of the network changes from day to day and evolves over time. In addition, the spatial knowledge of a driver is constantly evolving in response to trips made throughout the network.

The DRACULA framework gives special emphasis to the microscopic simulation and is basically composed of both a demand and a supply model. In the demand model, drivers are individually represented. For each driver, daily departure time and route choices are made on the basis of both past travel cost experiences and the knowledge built from perceptions of the network conditions. Contrary to models based on a fixed matrix approach, the demand stage predicts the level of individual demand for a day i from a full population of potential drivers. In the supply model, on the other hand, individual vehicles move throughout the network and follow their routes according both to the car-following and to the lane-changing rules. The resulting travel conditions of a certain day i and the costs experienced by drivers are re-entered into their individual knowledge bases. Costs are perceived in terms of travel time experienced on a link-by-link basis, and drivers have a limited capacity to remember past experiences. Individuals forget experiences that are older than a certain number of days. This is defined as a global parameter, which means that drivers have the same ability to perceive the environment. Their internal model will affect the demand for the next period, i.e., day $i + 1$. This process continues for a pre-specified number of days. Fig. 1 depicts the basic DRACULA framework.

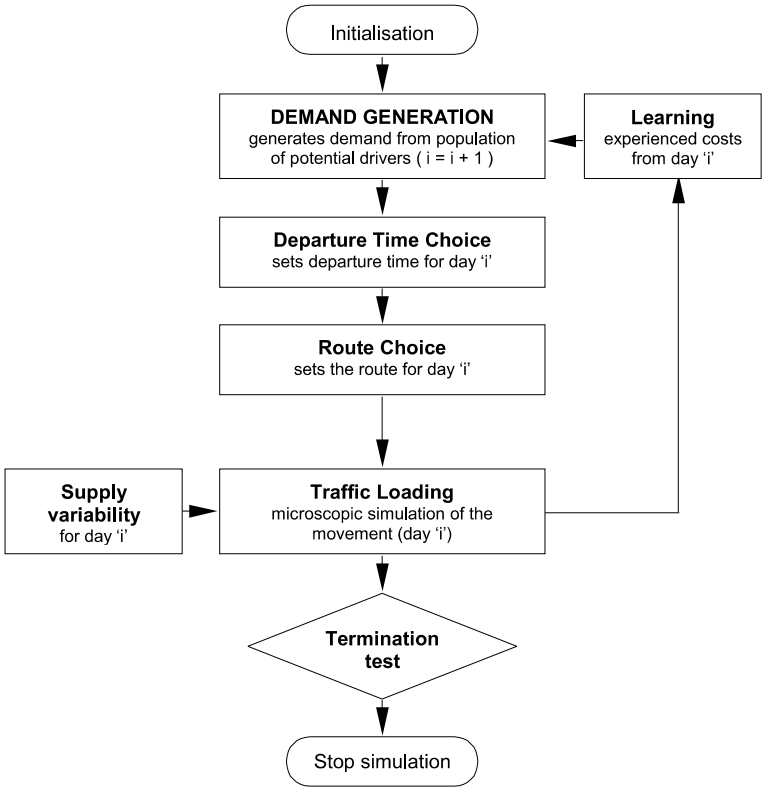

Fig. 1. The simulation process of the DRACULA framework.

When explicitly representing driver behaviour in microscopic simulation models, two aspects of human reasoning deserve special attention, namely the learning mechanism and the decision-making process. Drivers definitely need to make decisions regarding departure time, which route to take, trip purpose, and so forth, in order to perform a journey. Also, it is quite intuitive that drivers make use of remembered past experiences to aid their decisions, as they become more familiar with the network. Traditionally, decision-making processes have been approached in a centralised way, which means that a module is responsible for assigning the values for each variable of the drivers' decision.

In DRACULA, there are basically two ways used to assign departure time choice (Liu et al., 1995). The first and simplest method is to randomly assign a desired departure time for each potential driver in the modelled population. When drivers choose to travel on a certain day, they will depart at that desired departure time. Such decision does not depend on either route choices or any previous experiences. In this case, the departure time profile could be flat or distributed according to some user-specified distribution.

The second method incorporated into DRACULA, and considerably more complex, implements the choice in response to travellers' experience. In such a method, departure time is assigned prior to every journey. This process is based both on travellers' preferred arrival time and on experiences gathered from previous days. The costs experienced are used to build a knowledge base, which helps drivers to forecast the future state of the network. If the difference between the desired arrival time plus a scheduled delay and the actual arrival time is bigger than nil, the driver should adjust the departure time. Otherwise, the current departure time will be assumed.

The process of choosing routes is also based on past experiences. After each journey, individuals use the cost experienced from each link (the travel time on that link) along the chosen route to update their information about the network conditions. This is achieved by means of providing individuals with memory. Hence, the perceived cost will be an average of the remembered experiences for each link. The model adopted in DRACULA assumes drivers will take their usual daily route unless the cost of travelling through the minimum cost route is significantly better (Liu et al., 1995; Mahmassani and Jayakrishnan, 1991). Route choices are taken prior to the journey and drivers are not able to make en-route diversions.

With regard to the representation of vehicle movements, both the car-following and lane-changing models are implemented. From this perspective, DRACULA may also be seen as a reactive multi-agent system, as most driving behaviour reflects a mapping from perceived events to corresponding actions. On each day, and after every parameter has been set, drivers are launched onto the network at their origins and will follow their routes to respective destinations. Whenever performing a journey, drivers react to the presence of other drivers and to the environment as well.

3.2. An agent-based demand model

The DRACULA framework is extended to deal with the demand side by means of a population of driver agents and ITS technologies are featured in the framework to aid drivers' decision making (see Fig. 2). Drivers are dealt with as cognitive entities through the use of a BDI approach (Rao and Georgeff, 1991), where the internal model of each agent is essentially

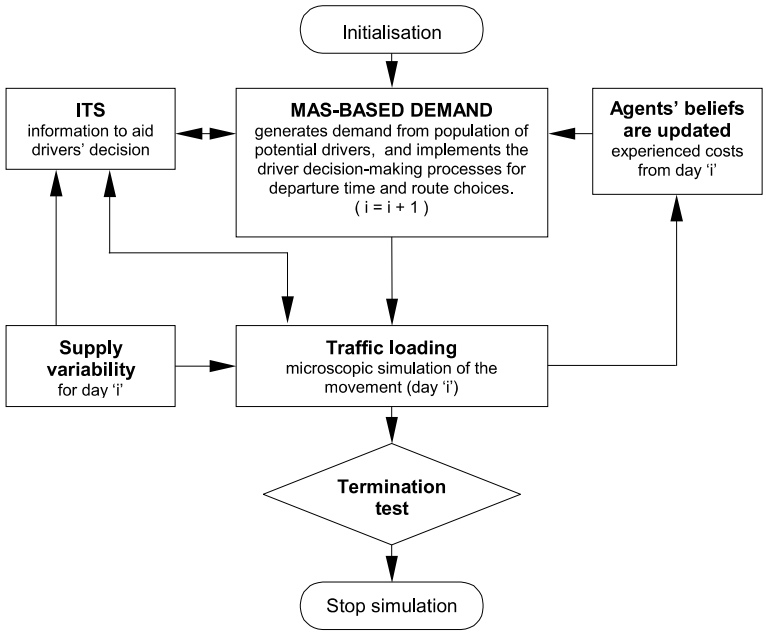

Fig. 2. The proposed extension to the DRACULA framework.

represented by sets of beliefs, goals, and intentions (as seen in following sections). Contrary to the original DRACULA framework, the decision-making process is made on a decentralised basis, which means that it is the driver's own responsibility to identify its needs, to manage its resources, and to take its decisions. The presence of communication facilities allows drivers to interact with different ITS technologies, e.g., ATIS. These technologies are also modelled as agents and are assumed to have a global representation of the world. The *supply variability* module, in Fig. 2, prepares the traffic environment conditions at the beginning of each day and updates the ITS technologies' internal model, which is used to provide drivers with information about the system conditions.

In the commuter scenario, driver agents should have autonomy to decide whether to travel at certain moment on the day accounting for their knowledge about the most preferable routes and the dynamics of the traffic system. The capacity to learn from previous experiences affects driver's beliefs, enhances its knowledge, and improves the decision-making process. As commuters know about constraints posed by non-flexible arrival time at destinations, they are expected to evaluate and decide whether to change route or departure time. The driver internal knowledge may also be enriched with information provided by systems such as ATIS. An important factor that influences the driver's beliefs on the current route conditions is the information from exogenous sources. Hence, the user should be able to perceive the environment state either by accessing some sort of information system or by forecasting it from its knowledge.

With regard to the decision-making process, some models such as DRACULA implement a quantitative approach. The metrics associated to those models considers that drivers use the cost

of past experiences memorised in terms of travel time for each link. The parameters are processed and used for estimating the cost of future journeys.

In practice, cognitive reasoning of human beings is mostly associated to qualitative aspects of the environment rather than an exact numeric measure. Intuitively, a driver does not remember the travel time explicitly, neither for each link nor for the entire route. To the contrary, the elements of the network are associated to qualitative measures. For instance, when travelling through a link with an average speed at least equal to the desired speed, the link can be associated to a *free flow* state. If the average speed performed through a link is lower than the desired speed and higher than a certain value, which is identified as a stationary state, the state associate to that link is *normal*. Finally, a *jammed* state is identified when the average speed is equal or lower than a certain value that would characterise a stationary state, bellow 20 km per hour for instance. These thresholds could be different for each driver and could also depend on the characteristics of links, such as their capacity. With regard to routes, the average travel time could be identified by means of the difference between the arrival time at destination and the departure time. It is a more intuitive way to represent the reality albeit it is just a different manner to write the sum of all link costs within the route.

Nevertheless, it is possible to identify many other metrics to represent the human cognition in a commuter scenario. Another example, more adequate as en-route reasoning strategy though, is to compare the desired arrival time against the time needed to perform the remaining links of the chosen route. Hence, the driver is continuously evaluating the quality of its initial choice on the basis of the remaining estimated travel time with respect to the desired arrival time. The driver could try to adjust its desired travel speed as an attempt to meet the desired arrival time. In the model presented in this work, we adopt a qualitative approach by which drivers keep in their set of base beliefs the state associated to links of known routes. For every new link, the knowledge base is updated with the addition of a new belief. In the following sections, we briefly describe the BDI reasoning mechanism and present an AgentSpeak(L) specification of our commuter scenarios to assess variable demand.

4. The specification language: AgentSpeak(L)

AgentSpeak(L) is a language devised to bridge the gap between formal modelling and practical implementation as far as BDI agents are concerned. It basically reduces the task of modelling intentional agents to identifying base beliefs, goals and plans. Before going further on the specification of the commuter scenarios we shall present in this work, we think it is worthwhile presenting the syntax of AgentSpeak(L), so as to facilitate the understanding of the specification of our driver model. The following definitions giving the syntax of the language are taken from Rao (1996), where the author first specified it, and are presented here exactly as they were given in his original work.

Definition 1. If b is a predicate symbol, and $(t_1, \ldots, t_n)$ are terms then $b(t_1, \ldots, t_n)$ or $b(\boldsymbol{t})$ is a belief atom. If $b(\boldsymbol{t})$ and $c(\boldsymbol{s})$ are belief atoms, $b(\boldsymbol{t}) \wedge c(\boldsymbol{s})$, and $\neg b(\boldsymbol{t})$ are *beliefs*. A belief atom or its negation will be referred to as a *belief literal*. A ground belief atom will be called a *base belief*.

Definition 2. If g is a predicate symbol, and $(t_1, \ldots, t_n)$ are terms then $!g(t_1, \ldots, t_n)$ (or $!g(\boldsymbol{t})$) and $?g(t_1, \ldots, t_n)$ (or $?g(\boldsymbol{t})$) are *goals*.

Note that both beliefs and goals are predicate symbols (as are actions, as seen later). A predicate symbol is a goal if it is preceded by the operators '!' or '?' (see Definition 2). Thus, an agent a may have the *achievement goal* of being in location Y in a future state, expressed by $!location(a, Y)$. It may also check what its present position is, by a *test goal* such as $?location(a, X)$, given its set of base beliefs (updated through perception of the environment).

Definition 3. If $b(\boldsymbol{t})$ is a belief atom, $!g(\boldsymbol{t})$ and $?g(\boldsymbol{t})$ are goals, then $+b(\boldsymbol{t})$, $-b(\boldsymbol{t})$, $+!g(\boldsymbol{t})$, $+?g(\boldsymbol{t})$, $-!g(\boldsymbol{t})$, $-?g(\boldsymbol{t})$ are *triggering events*.

Agents go through repeated cycles where they observe the environment and, based on their observations and goals, they execute certain actions that may change the state of the environment. This may influence the agents' beliefs as well, which need to be revised.

Definition 4. If a is an action symbol and $t_1, \ldots, t_n$ are first-order terms, then $a(t_1, \ldots, t_n)$ or $a(\boldsymbol{t})$ is an *action*.

Definition 5. If e is a triggering event, $b_1, \ldots, b_m$ are belief literals, and $h_1, \ldots, h_n$ are goals or actions then $e : b_1 \wedge \ldots \wedge b_m \leftarrow h_1; \ldots; h_n$ is a *plan*. The expression to the left of the arrow is referred to as the *head* of the plan and the expression to the right of the arrow is referred to as the *body* of the plan. The expression to the right of the colon in the head of a plan is referred to as the *context*. For convenience, an empty body is rewritten with the expression *true*.

Rao (1996) further mentions that a plan specifies the means by which an agent should satisfy an end. However, in none of the known work concerning AgentSpeak(L) (Rao, 1996; d'Inverno and Luck, 1998) the authors approach the issue of how beliefs and intentions are updated during the execution of a plan. [2]

Definition 6. An *agent* is given by a tuple $\langle E, B, P, I, A, S_{\mathrm{E}}, S_{\mathrm{O}}, S_{\mathrm{I}} \rangle$, where E is a set of events, B is a set of base beliefs, P is a set of plans, I is a set of intentions, and A is a set of actions. The selection function S_{E} selects an event from the set E; the selection function S_{O} selects an option or an applicable plan [3] from a set of applicable plans; and S_{I} selects an intention from the set I.

[2] We refer here to the updating of beliefs and deletion of intentions directly from the execution of a plan, not (in the case of beliefs) through changing the environment by means of actions and subsequent perception of the environment and the ensuing belief update, or (in the case of intentions) executing sub-plans. Although, in some of the examples given in the papers it appears that this is possible, formally a plan is only formed of goals and actions, not triggering events (i.e., addition and deletion of beliefs).

[3] Rao (1996) presents *applicable plans* in Definition 10 in his original work.

Definition 7. The set I is a set of intentions. Each *intention* is a stack of *partially instantiated plans*, i.e., plans where some of the variables have been instantiated. An intention is denoted by $[p_1 \ddagger \ldots \ddagger p_z]$, where p_1 is the bottom and p_z is the top of the stack. The elements of the stack are separated by '‡'. For convenience, Rao (1996) refers to the intention $[+!\texttt{true} : \texttt{true} \leftarrow \texttt{true}]$ as the *true intention*, and denotes it by T.

Definition 8. The set E consists of events. Each event is a tuple $\langle e, i \rangle$, where e is a triggering event and i is an intention. If the intention i is the *true intention*, the event is called an *external event*; otherwise it is an *internal event* (and i is the intention that has generated the event e).

We now give a brief overview about the operation cycle of a BDI agent in AgentSpeak(L). As described by d'Inverno and Luck (1998), there are two basic models of operation, which essentially involve either updating the intention set, reflected by responding to an event, or actually executing intentions. When updating the intention set, the agent selects an event e, from the set of events E, and generates all the plans whose invocation conditions, identified by the triggering event at the head of the plan, match this event. These plans are the *relevant plans*. Then, if the context part of a relevant plan is a logical consequence of the set of base beliefs, it will be selected as an *applicable plan* and will form a plan instance that is the agent's *intended means*. Ending the cycle, the agent updates its set of intentions I. If the event selected, which started the cycle, is an external event, a new intention is generated and inserted into the intention set. Otherwise, the event is internal and the plan instance is added to the head of the intention that posted the event.

In the second model, the agent selects an intention from the intention set I. The plan at the top of the selected intention is now the *executing plan*, and the next formula in the body of the plan is the *executing formula*. Depending on the selected intention and the executing formula of the executing plan, the agent starts one of the possible courses of action. If the executing formula is an achieving goal, a new goal event is generated and posted to the event set E, and the intention is suspended until the goal has been achieved. In case the executing formula is a query goal, the information retrieved is used to instantiate the corresponding terms in the executing plan. Finally, if the executing formula is an action, the action is posted to the action set A, awaiting execution. In the last two situations, the executing formula is removed from the executing plan. If there is no further formula in the executing plan, the agent starts the execution of the next plan in the selected intention. If there is no next plan, then the intention has succeeded and can be removed from the intention set.

Note that an AgentSpeak(L) agent is specified simply as a set of base beliefs and a set of plans. Intentions are generated automatically from triggering events. This process is detailed in (Rao, 1996; d'Inverno and Luck, 1998). Here we have focused only on those elements that are necessary to characterise the BDI agents we present next. In the following section we use AgentSpeak(L) as an specification language to model commuter scenarios.

5. A BDI driver model

The main goal of this work is to present a framework such that it can be used to model and implement commuter scenarios. The traffic system is abstracted and modelled as a multi-agent

world where the environment is the network, basically described in terms of links connected to one another, and drivers are dealt with as decision-making entities. In our model, we shall consider that there exists a population of potential drivers who are able to perform journeys throughout the road network, as suggested by Liu et al. (1995). Thus, drivers from the population, who have decided to make a trip on a certain day, constitute the demand for travel on that day.

Nonetheless, another component plays an important role in modern traffic and transportation engineering: the ITS. One of the key objectives of ITS is to directly influence users' behaviour pattern as an alternative to cope with the limited capacity of the network infrastructure. ATIS and Advanced Traffic Management Systems (ATMS) are some examples among others. The interested reader is referred to Chatterjee et al. (1999) and Garrett (1998) for more details on those technologies. We shall give special attention to information systems, which are important exogenous information sources for drivers. Two agents are identified as the main protagonists in our multi-agent traffic system, namely the driver agent and the information system agent, hereafter referred to as the *its-agent*. So, our multi-agent traffic system is given by the tuple $\langle N, A, ITS\rangle$, where N is a set of road links, A is a set of driver agents, and ITS is a set of intelligent transportation technologies.

The links of the network N are assumed to be oriented. In other words, if the upstream node of a link, say l_1, is connected to the downstream node of another link, say l_2, then l_1 and l_2 are said to be adjacent in this order. Hence a driver is able to infer that it may move along l_1 towards l_2. We represent this in terms of belief predicates. For instance, if $connected(upstream(l_1), downstream(l_2))$ then $adjacent(l_1, l_2)$.

A journey performed throughout the network starts at an *origin* and ends at a *destination*, usually represented as traffic zones. In this work, we consider that both origin and destination are also links in the road network. Thus, *route* is a belief predicate defined as $route(zone_o, zone_d, time_r, [link_1, \ldots, link_i, link_{i+1}, \ldots, link_n])$, where $zone_o$ is the origin and $zone_d$ is the destination of the route. The term $time_r$ represents the travel time whereas terms between square brackets are a list of consecutive links from $zone_o$ to $zone_d$, hence $adjacent(link_i, link_{i+1})$.

Yet, with regard to link characteristics, we could have some notion associated to the quality of service. The belief predicate $state(l, s)$ represents the state of a link, where $l \in N$ and s is the qualitative representation of the demand over l, which may be one of $\{free, normal, jammed\}$.

As already mentioned, a driver agent is represented as a reasoning and intentional entity in the system. Its behaviour is represented in terms of mental attitudes such as beliefs, desires, and intentions, which represent informational, motivational, and intentional states, respectively. Then, a driver cognitive structure is given by the tuple $\langle E, B, P, I, A, S_E, S_O, S_I\rangle$, as presented in Definition 6, for BDI agents.

Albeit intelligent transportation technologies are also modelled as agents, only drivers have a BDI representation. An *its-agent* is a simple structure that has a global representation of the environment, with both topologic and dynamic models, to provide updated and useful information to drivers. Considering that drivers will constitute the demand for travel competing for limited resources, i.e., the limited capacity of the road infrastructure, the *its-agent* will play a role of mediating conflicts in order to optimise the use of such resources. Examples of environments where a mediator agent tries to sort out conflicting situations can be found elsewhere (Sycara,

1989; Tedesco and Self, 2000). In this work, we are especially concerned with the cognitive structure of a driver agent. Nonetheless, ITS technologies are reaching high levels of intelligence, autonomy, and pro-activity, and modelling such systems as BDI agents will also be considered in detail in following works.

Time is another important notion that captures the dynamic essence intrinsic in this application domain. In this work, we use a sequence of external stimuli to represent how time evolves. With this respect, two elements are present in our system, namely day and time. The former represents the actual day on which a driver is supposed to make decisions concerning the trip and is given in terms of the predicate $today(day)$. The latter represents an instant within the period of time being evaluated on that day and is written as $timeNow(time)$.

In our commuter scenario, the very first decision an agent makes on each day is whether to travel. This decision is represented in terms of a trip purpose. So, a driver may decide to go shopping, to go to work, and so forth, or even to stay at home. In the latter case, it has decided not to travel and will not take part in the demand for that day. Another important information related to the purpose of a trip is the day on which the agent intends to perform certain activities. $purpose(day, p)$ is the predicate used to represent this idea, where day is an integer representing the day on which the activity must be undertaken and p is the actual purpose of the trip, which can be any from a set such as $\{working, shopping, leisure, \ldots\}$. The belief predicate $destination(p, zone_{\mathrm{dst}})$ is used to associate a place to the purpose. Thus, p represents the purpose whereas $zone_{\mathrm{dst}}$ represents the destination zone for that activity, which is fixed at the beginning of the simulation.

Basically, when planning a journey, the driver needs to make decisions about the purpose for the trip, the destination it intends to reach, the desired arrival time, what time it needs to depart, and a route to its destination. Thus, a trip is given by the tuple $\langle p, z_{\mathrm{org}}, z_{\mathrm{dst}}, t_{\mathrm{arv}}, t_{\mathrm{dpt}}, r\rangle$, where p represents the purpose, z_{org} is the origin zone, z_{dst} is the destination zone, t_{arv} is the desired arrival time at destination, t_{dpt} is the departure time, and r is the route. The belief predicate $location(l)$ is used to represent the location of a driver and is also used both as an achievement and as a test goal. So, $!location(l)$ represents that the driver is willing to reach link l, whereas $?location(l)$ represents the goal to check which its present location l is. An example of a possible set of base beliefs for our BDI driver could include the clauses presented in Fig. 3.

With this very first picture of the application domain, it is now possible to start representing the dynamics of the reasoning mechanism of drivers by means of a BDI architecture.

Commuters are basically involved with executing repetitive trips, such as standard home–work–home journeys. On a certain day, say i, it is possible for someone to either decide to stay at home or to go to work (we aim here at a flexible framework so that we could represent drivers with different distributions of working days and also capable of having different activity purposes for the period of time considered in the simulation). In this way, assuming the driver is willing to make the trip on day i, it should decide on which route to take and what time to depart. In today's traffic systems, exogenous information sources are very likely to affect such decisions in many respects. Next we shall define three commuter scenarios ranging from the simplest one, Scenario 1, to more complex situations, as in Scenarios 2 and 3, where we consider the presence of information systems.

```
adjacent(h, link1).        adjacent(link1, link2).
adjacent(link2, w).        adjacent(h, link3).
adjacent(link3, link4).    adjacent(link4, link7).
adjacent(link7, w).        adjacent(h, link5).
adjacent(link5, link6).    adjacent(link6, link7).

route(h, w, 30, [link1, link2]).
route(h, w, 45, [link3, link4, link7]).
route(h, w, 35, [link5, link6, link7]).

state(link5, jammed).      purpose(4, work).
state(link3, free).        purpose(5, home).
state(link7, jammed).      purpose(6, leisure).

today(4).                  timeNow(0800).

preTripInformationSystem(user).
enRouteInformationSytem(non_user).
acceptanceWillingness(20).

perceivedArrivalCost(10, work).
usualDepartureTime(h, w, 0815).
expectedTravelTime(h, w, 999).

tripRoute(h, w, [link1, link2], usual).
tripDepartureTime(0815, [link1, link2]).
```

Fig. 3. Example of possible clauses in a set of base beliefs.

5.1. The commuter scenarios

5.1.1. Scenario 1

In the first scenario, drivers are supposed to make every decision with respect to the trip before starting the journey. Afterwards, diversions are not allowed. We assume that no information is provided to individuals, neither before nor during the trip. Therefore, drivers make their decisions based solely on their set of base beliefs. Note that AgentSpeak(L) uses similar convention as in Prolog, namely variables start with capitals whereas constants and predicates begin with lowercase letters. Thus, as plans are partially instantiated, we use capitals to indicate variable terms. Also, we have extended the language to allow for integer constants, relational operators, and list notation to be used. These simple extensions allow for more clear specifications.

Plan 1. Starting the day. Perceiving that a new day starts is an external stimulus, which is represented by the event $+today(day)$. The set of base beliefs is updated, then. When the driver perceives the beginning of another day it becomes committed to an activity purpose for that day.

```
01. +today(Day) : true
02.    <-  ?purpose(Day, P);
03.        ?destination(P, ZDst);
04.        !location(ZDst).
```

With respect to the route selection, we use the achievement goals !*anyRoute*($zone_{org}, zone_{dst}$, [*links*]), !*bestRoute*($zone_{org}, zone_{dst}$, [*links*]), and !*usualRoute*($zone_{org}, zone_{dst}$, [*links*]), to represent the desire of a driver to choose any route, the best route, or the usual route, respectively, among the *route*($zone_{org}, zone_{dst}, t$, [*links*]) clauses in the set of its base beliefs. For the sake of readability and to ease exchanging parameters among plans, we use a variable term, say *R*, to capture an instance of the link list of a route *route*($zone_{org}, zone_{dst}, t$, [*links*]). So, when *R* is instantiated, it is assigned the content of [*links*].

The belief predicate *tripRoute*($zone_{org}, zone_{dst}$, [*links*], *strategy*) is used to represent the particular route chosen by the driver. The additional term *strategy* recalls the selection method used and can be one of {*any*, *best*, *usual*, *system*}. Thus, whenever a route is selected, the set of base beliefs is updated and the belief *tripRoute* is added to represent the driver's selection.

Plan 2a. Changing location by selecting any route. In this case the driver opts for the strategy of getting any route, despite the cost. After choosing the route, the driver proceeds with setting the departure time given the selected itinerary. This is written in terms of the achievement goal !*chooseDepartureTime*([*links*]), in line 3.

```
01. +!location(Z_Dst):location(Z_Org) & (not(Z_Dst = Z_Org))
02.    <- !anyRoute(Z_Org,Z_Dst,R);
03.       !chooseDepartureTime(R).
```

Plan 2b. Changing location by selecting the usual route. The strategy adopted in this plan is to choose the usual route.

```
01. +!location(Z_Dst):location(Z_Org) & (not(Z_Dst = Z_Org))
02.    <- !usualRoute(Z_Org,Z_Dst,R);
03.       !chooseDepartureTime(R).
```

Plan 2c. Changing location by selecting the best route. In this plan, the driver attempts to find the best route among the *route* ($zone_{org}, zone_{dst}, t$, [*links*]) clauses the set of its base beliefs.

```
01. +!location(Z_Dst):location(Z_Org) & (not(Z_Dst = Z_Org))
02.    <- !bestRoute(Z_Org,Z_Dst,R);
03.       !chooseDepartureTime(R).
```

Plan 2d. Getting to the intended location (end of the recursion). Basically, the context part of this plan identifies two situations. In the first case, the driver decides to stay at home, for example. So, the intended location is already the present location. In the other situation, the driver has moved throughout the chosen route and finally arrived at the destination. In both cases, as the driver perceives that it is at the desired location, it then deduces that there is nothing else to do.

```
01. +!location(Z_Dst):location(Z_Dst)<- true.
```

Plan 3. Choosing any route. To confirm the selection, an update to the set of base beliefs occurs in line 2, and the trip route is added to the base beliefs.

```
01. +!anyRoute(Z_Org, Z_Dst, R) : route(Z_Org, Z_Dst, T, R)
02.     <-  +tripRoute(Z_Org, Z_Dst, R, any).
```

Plan 4. Choosing the usual route. In our commuter world model, we consider that the driver already has in its base beliefs a belief of having a preferred route, which is the usual route. In this case, the driver will keep such a route.

```
01. +!usualRoute(Z_Org, Z_Dst, R) : tripRoute(Z_Org, Z_Dst, R, usual)
02.     <-true.
```

The belief predicate *expectedTravelTime* ($zone_{org}$, $zone_{dst}$, *time*) is used to represent the expected travel time from $zone_{org}$ to $zone_{dst}$. No route is considered, though. In other words, a driver may have an estimation of the necessary time to reach a certain destination without considering any route, at first instant. The driver does not have any guarantee that it will experience the same travel time *T* for route *R*. In the context part of Plan 5a, it evaluates whether the travel time for the route it is considering is lower than the travel time it expected to experience. In a first execution of this sub-plan, before considering any alternative route, the driver has a ''pessimistic behaviour'' and sets a very high value for its expected travel time (so as to set reasonable expected times after whatever first attempt is made)—refer again to Fig. 3, this is the reason for the high value as parameter. Note that the context part of the plan will not be satisfied once the best route has been found, thus finishing the recursion (see next plan).

Plan 5a. Choosing the best route.

```
01. +!bestRoute(Z_Org, Z_Dst, R) : route(Z_Org, Z_Dst, T, R)
02.     & expectedTravelTime(Z_Org, Z_Dst, T_Exptd) & (T < T_Exptd)
03.     <-  +tripRoute(Z_Org, Z_Dst, R, best);
04.         +expectedTravelTime(Z_Org, Z_Dst, T);
05.         !bestRoute(Z_Org, Z_Dst, R).
```

Plan 5b. Keeping the instantiation for the best route. This plan finally keeps the instantiations of the appropriate parameters for *bestRoute*, when the recursion in Plan 5a finishes (when all attempts to satisfy the context part of Plan 5a fail, the best route then instantiated is the one to be chosen).

```
01. +!bestRoute(Z_Org, Z_Dst, R) : true<- true.
```

Plan 6a. Choosing departure time in the case of having any route. Once drivers have chosen the route, they can set a departure time for the trip, given by the addition of the belief *tripDepartureTime*(*time*, [*links*]), in line 6.

```
01. +!chooseDepartureTime(R) : tripRoute(Z_Org, Z_Dst, R, any)
02.     <- ?today(Day);
03.        ?purpose(P, Day);
04.        ?arrivalTime(P, T_Arv);
05.        ?route(Z_Org, Z_Dst, T, R);
06.        +tripDepartureTime(T_Arv − T, R).
```

Plan 6b. The same is considered in the case of having the best route.

```
01. +!chooseDepartureTime(R) : tripRoute(Z_Org, Z_Dst, R, best)
02.     <- ?today(Day);
03.        ?purpose(P, Day);
04.        ?arrivalTime(P, T_Arv);
05.        ?route(Z_Org, Z_Dst, T, R);
06.        +tripDepartureTime(T_Arv − T, R).
```

As for the expected travel time, the belief predicate *usualDeparture* $(zone_{org}, zone_{dst}, time_{usl})$ is used to represent the departure time, $time_{usl}$, usually adopted when travelling from $zone_{org}$ to $zone_{dst}$, without considering any itinerary. We use the belief *perceivedArrivalCost* $(time_{dly}, zone_{dst})$ to represent the cost for being late, in terms of time $time_{dly}$, experienced in a previous journey to a certain destination, $zone_{dst}$. As an extension for this belief, we could also consider a schedule delay associated to the purpose of the trip as well as the cost associated to the fact of arriving earlier.

Plan 6c. Contrary to Plans 6a and 6b, a different approach is used when setting the departure time for the usual route.

```
01. +!chooseDepartureTime(R) : tripRoute(Z_Org, Z_Dst, R, usual)
02.     <- ?usualDeparture(Z_Org, Z_Dst, T_Usl);
03.        ?perceivedArrivalCost(T_Dly, Z_Dst);
04.        +tripDepartureTime(T_Usl T_Dly, R).
```

The predicate *moveAlong*(*route*) is a basic action, i.e., the agent can execute it directly on the environment. *route* is the term representing the itinerary the agent opted to follow. Contrary to the basic action *move*, which we shall present for the third scenario, *moveAlong* is executed over the whole route. This action does not consider link by link, as drivers are not allowed to divert the chosen itinerary during the course of a journey. So, the driver's location is updated to the destination as a result of such an action.

Plan 7. This is a plan for effectively moving along the chosen route. Once a driver is committed to achieve certain destination and perceives that the desired departure time holds at that moment, it can execute its journey, then.

```
01. +timeNow(T) : tripDepartureTime(T, R)
02.     <- moveAlong(R).
```

Plan 8. When the driver reaches its destination, it compares the present time with the desired arrival time. The belief *perceivedArrivalCost* is updated to reflect the experienced delay in future journeys.

```
01. +location(Z_Dst) : true
02.    <-  ?today(Day);
03.        ?purpose(Day, P);
04.        ?arrivalTime(P, T_Arv);
05.        ?timeNow(T);
06.        +perceivedArrivalCost(T − T_Arv, Z_Dst).
```

5.1.2. Scenario 2

For the second scenario we shall consider that drivers are allowed to ask for information before starting a trip, which could help their decision-making process. No en-route diversion is possible, though. In this way, we shall make some considerations with regard to communication in our multi-agent traffic system. Before going on with further plan specifications, we mention that, in our approach, inter-agent communication is accomplished through message passing. We assume that sending a message is a basic action, whereas receiving a message causes, as with perception, the addition of a belief in the agent's belief base, which may in turn trigger a plan execution. Some trip planner applications on the Internet are examples of such information sources.

Let us consider that $b(t_1, \ldots, t_n)$ is a belief predicate as presented in Definition 1. Thus, *communicate* (*ag*, "$b(t_1, \ldots, t_n)$"), *request*(*ag*, "$b(t_1, \ldots, t_n)$"), and *broadcast*("$b(t_1, \ldots, t_n)$") are special cases of basic action predicates. The term *ag* is used to identify the agent to which the message is addressed, whereas "$b(t_1, \ldots, t_n)$" represents the propositional content of the message. The *communicate* predicate is used to send *ag* the belief $b(t_1, \ldots, t_n)$. The *request* predicate asks agent *ag* for $b(t_1, \ldots, t_n)$. In this case, we assume that the agent *ag* (e.g., an information system) presents a "benevolent" behaviour and always replies to the request made (e.g., by a driver). Note that there is no agent addressed in the *broadcast* predicate. In such a case, the content is sent to all of the agents in the multi-agent system.

Imagine that an agent, say a_1, sends a message $b(\mathtt{t})$ to a driver, say agent a_2. Then, a_1 executes *communicate*(a_2, "$b(\mathtt{t})$") (or *broadcast*("$b(\mathtt{t})$") to all drivers in the system). When a_2 receives the message, an event $+b(\mathtt{t})$ occurs and $b(\mathtt{t})$ is added to the set of its base beliefs, as with perception. So, a_2 cannot distinguish whether $+b(\mathtt{t})$ is a simple perception or a message passed through communication. In such a situation, we suppose that the *belief revision function* checks whether a_1 is trustworthy. If a_1 is considered to be trustworthy, besides adding $b(\mathtt{t})$ to the belief base, the function adds another belief predicate, *informed*(a_1, $b(\mathtt{t})$), indicating who has informed a_2 about $b(\mathtt{t})$. Thus, it is possible to have plans associated both to the content of the message and to the informing agent, if considering the sender is relevant.

With respect to interactive information sources, we identify two major groups of drivers. The first group is formed with those drivers who are eligible to use the information system, either because they are subscribers or because they are equipped to receive the information. Those drivers who are not users of information systems form the second group. In order to consider such a relation in the model, we use the belief predicates *preTripInformationSystem*(*category*) and

enRouteInformationSytem(*category*) to identify whether drivers are users of pre-trip and en-route information systems, respectively. The *category* term identifies between users and non-users, and can be one of $\{user, non\_user\}$. The second predicate anticipates what we shall present in the third scenario when we talk about en-route diversion.

However, saying whether the driver is user or non-user of information systems is not enough to guarantee that the information provided will be considered in the decision-making process. In order to make this criterion complete, we use the belief predicate *acceptanceWillingness*(*value*) to represent whether the driver is willing to accept and to use the information provided. The term *value* is a random number used to capture the probability of a driver to accept an advice. By comparing this value with a certain threshold, we can identify from the population of drivers how many percent of them will use the information. For instance, one could want to know the impact of information systems when 20% of the population is very likely to use the information provided. This *value* is generated by the *belief revision function* and is updated at every agent cycle, which means it may be different from day to day. A complete description and detailed formalization of such a function is subject of following work.

Plan 9. In this case, the driver opts to ask for advice. This plan is similar to Plans 2a, 2b, 2c. The driver is aimed at getting a route suggested by the system, though. In line 4, the term *sysAg* identifies the information system to which the message is addressed. Note that we use *sysAg* in the plan code, as it is an instantiated term, not a variable. The belief predicate *sysRoute*($zone_{org}, zone_{dst}, t, [links]$) is the route suggestion expected from *sysAg*. Contrary to Plans 2a, 2b, 2c, the driver cannot proceed with the selection of a departure time. It is necessary to wait until variable *R* has been instantiated.

```
01. +!location(Z_Dst) : location(Z_Org) & (not(Z_Dst = Z_Org))
02.     & preTripInformationSystem(user) & acceptanceWillingness(V)
03.     & (V <= 20)
04.     <-  request(sysAg, sysRoute(Z_Org, Z_Dst, T, R)).
```

Plan 10a. An event occurs indicating that the request has been responded (or a broadcast has been sent). The decision on whether to accept it is made as soon as the message arrives. The driver accepts the suggestion and *tripRoute* is instantiated. In line 2, the *strategy* term of the predicate *tripRoute* is *system*, indicating the route was suggested by an exogenous source. In line 3, note that only after *tripRoute* has been instantiated the driver can proceed with the selection of a departure time.

```
01. +sysRoute(Z_Org, Z_Dst, T, R) : acceptanceWillingness(V) & (V <= 20)
02.     <-  +tripRoute(Z_Org, Z_Dst, R, system);
03.         !chooseDepartureTime(R).
```

Plan 10b. In this case, the event occurs indicating the request has been responded (or a broadcast has been sent) but the driver ignores it and opts for the usual route instead. Other strategies could be used as alternatives to choosing the usual route, in line 2, such as any route.

```
01. +sysRoute(Z_Org, Z_Dst, T, R) : acceptanceWillingness(V) & (V > 20)
02.     <- !usualRoute(Z_Org, Z_Dst, Q);
03.        !chooseDepartureTime(R).
```

Plan 11. In the case of adopting the suggested route, setting the departure time follows the same approach as in Plans 6a and 6b.

```
01. +!chooseDepartureTime(R) : tripRoute(Z_Org, Z_Dst, R, system)
02.     & sysRoute(Z_Org, Z_Dst, T, R)
03.     <- ?today(Day);
04.        ?purpose(P, Day);
05.        ?arrivalTime(P, T_Arv);
06.        +tripDepartureTime(T_Arv − T, R).
```

5.1.2.1. Requesting link state

Plan 12. When the driver is not very confident of its choice, it may evaluate the state of links within the route and possibly use such information to decide on another route. The following plan illustrates this situation after an *anyRoute* has been chosen. In this situation, the driver decides to re-evaluate the state of a certain link believed to be jammed as perceived from previous journeys.

```
01. +tripRoute(Z_Org, Z_Dst, R, any) : preTripInformationSystem(user)
02.     & state(L, jammed) & acceptanceWillingness(V) & (V <= 20)
03.     <- !member(L, R);
04.        request(sysAg, state(L, S)).
```

In line 1 of Plan 12, note that an event indicating that *tripRoute* has been instantiated occurs, and the driver can infer anything about the selected route, then. The context of this plan identifies all the links in itinerary *R* that the driver believes to be in a jammed state. Such a belief is acquired from past experiences and now the driver attempts to check the actual state of the link. In line 3, we use an achievement goal !*member*(*Link*, *Route*) to check whether the link with which the variable *Link* is instantiated is a member of the list of links with which the variable *Route* is instantiated. It would not be practical to keep this information as beliefs for all links in all routes. The plan for the goal *member* is not given here, but it is as simple as in Prolog.

Plan 13. The answer from the information system is perceived. The driver's acceptance results in an attempt at assuming the usual route instead. After adjusting the route, the departure time needs to be re-evaluated.

```
01. +state(L, jammed) : tripRoute(Z_Org, Z_Dst, R, Str)
02.     & acceptanceWillingness(V) & (V <= 20)
03.     <- !member(L, R);
04.        !usualRoute(Z_Org, Z_Dst, K);
05.        !chooseDepartureTime(K).
```

5.1.3. Scenario 3

In the third scenario, drivers can receive information from exogenous sources both before starting a trip and during the course of a journey. VMS and DRGS are examples of exogenous sources that can be used during a trip.

Contrary to the basic action predicate *moveAlong*(*route*), used for the previous scenarios, the *move*(*link*) predicate represents the movement on a link-by-link basis. Every time the *move* action is executed, it changes the present driver's location to the following link in the link list, and generates a new *location*(*link*) belief. The set of base beliefs is updated through the perception of the event +*location*(*link*). In this way, drivers are able to re-evaluate the quality of the trip during the journey.

Plan 14. Starting the trip. Once the driver has perceived the time to depart has arrived, it immediately starts moving through the links in the list R. Z_{Org} and Z_{Dst} are said to be dummy links in the sense they are used just to connect origin and destination to the network, respectively. Therefore, drivers will actually move only through links in the link list. As the driver starts moving through L, the list of remaining links in the selected route is updated. This is done to represent the fact that drivers keep attention on the links to come, whereas passed links are left aside.

```
01. +timeNow(T) : tripDepartureTime(T,R) & tripRoute(Z_Org, Z_Dst, R, Str)
02.     & (R = [L|Links]) & location(Z_Org)
03.     <- move(L);
04.         +tripRoute(Z_Org, Z_Dst, Links, Str).
```

Note that we use the Prolog notation to manipulate lists. Thus, the formula $R = [L|Links]$ in the context part of Plan 14 is used to instantiate the variables L and *Links*, which are the first and the remaining links of R, respectively. Hence, the *tripRoute* is updated in line 4.

Plan 15a. Keep moving as other links within the route are reached. Whenever the driver finishes moving through a link, its location changes to that link and it can start moving through the next one in the list, which is updated then. So, when the list gets empty the destination is reached.

```
01. +location(L) : tripRoute(Z_Org, Z_Dst, R, Str) & (R = [L_1|Links])
02.     <- move(L_1);
03.         +tripRoute(Z_Org, Z_Dst, Links, Str).
```

Plan 15b. Making a request during the course of a trip. The following plan is used whenever the driver decides to ask for help, for instance, to DRGS. At the moment, we are only considering alternative routes from the location being L to destination Z_{Dst}. Such information is provided by the system under request from the driver. This is the significant difference between this and the previous scenarios: drivers can divert and take a different route during the course of a journey.

```
01. +location(L) : tripRoute(Z_Org, Z_Dst, R, Str) & (R = [L_1|Links])
02.     & enRouteInformationSytem(user)
03.     & acceptanceWillingness(V) & (V <= 20)
```

```
04.     <-  request(sysAg, sysRoute(L_l, Z_Dst, T, Q));
05.         move(L_l).
```

Plan 15c. Reaching destination. The driver reaches the destination and the set of base beliefs is updated.

```
01. +location(L) : tripRoute(Z_Org, L, [ ], Str) <- true.
```

5.1.3.1. Non-interactive information sources. Non-interactive information sources are those aimed at reaching most users. Contrary to what we have seen so far, as drivers need to make a request to interactive sources in order to get information, communication happens in one direction only. Thus, requests are not necessary to receive information. Good examples of this kind of sources are the mass information systems, such as radio and TV. Traffic signs and, most recently, VMS, also have the ability to reach most drivers travelling throughout the network. In our commuter world, we consider that information provided by this kind of systems is mapped to link states. In practical terms, drivers will associate the content of information, such as messages, to possible states for links. We are not considering routes, though. Nonetheless, such situations are plausible, for instance, in the case alternative diversions to road disruptions are explicitly suggested. In order to represent such situation in the commuter agent model, this one-way communication is identified as an update of the belief set, in the same way as with messages received in response to driver's requests. Hence, whenever a belief *state*(*l*, *s*) is updated, an event is generated to indicate that a message has been received; in this case it is just a reminder that some link *l* is in the state *s*, not a response to a request made. As mentioned before, it is the *belief revision function*that checks the trustworthiness of the information source, and an event of the type $+informed(ag, b(\mathtt{t}))$ is also added to the base beliefs. At that moment, the driver may either accept or ignore it, depending on its acceptance willingness. Some situations are presented in the following plans.

Plan 16a. In this case, the driver receives the information before starting the journey. Although the route has already been selected, the driver may attempt to choose a better solution.

```
01. +state(L, jammed) : tripRoute(Z_Org, Z_Dst, R, Str) & tripDepartureTime(T, R)
02.     & timeNow(T_N) & (T < T_N) & acceptanceWillingness(V) & (V <= 20)
03.     <-  !member(L, R);
04.         !bestRoute(Z_Org, Z_Dst, Q);
05.         !chooseDepartureTime(Q).
```

Basically, there are two situations in which the message received, with respect to a certain link *L*, will have no relevance to the driver. In the first case, the driver is already at link *L*. In the second situation the driver is at any other link, but the link *L* is not a member of the route adopted, either because it has already been passed or because it has never been part of route *tripRoute*. Both situations are not written in form of plans as the achievement goal `!member(L, R)` will have failed.

Plan 16b. A message is received and the event is perceived. In this case, the content is still in context, the driver accepts it and tries to find the best route from the actual location (in case the trip has been started already) to the destination.

```
01. +state(L_S, jammed) : tripRoute(Z_Org, Z_Dst, R, Str) & location(L)
02.     & (not(L_S = L)) & acceptanceWillingness(V) & (V <= 20)
03.     <- !member(L_S, R);
04.        !bestRoute(L, Z_Dst, Q).
```

An alternative situation to Plan 16b in which a message has been received and its content can still be considered for use is when the driver is not willing to change its choices. So, the context part of Plan 16b will never match.

5.2. A framework to implement BDI drivers in commuter scenarios

This section aims at describing a framework to implement BDI drivers in commuter scenarios. It is presented as an aid to the assessment of the impact that intelligent transportation technologies, such as ATIS, may have on drivers' decision-making. Thus, the BDI cognitive structure previously presented is used as the basis for representing drivers within the simulation framework depicted in Fig. 4. On the *Demand* side, the variable demand is formed of drivers that have decided to travel on a certain day. The decision-making results from the mental attitudes of each of our BDI driver agents. SIM_Speak (Machado and Bordini, 2001, 2002) allows AgentSpeak(L) specifications to be implemented under the SIM_AGENT toolkit, which is a cognitive agent framework (Sloman and Logan, 1999; Sloman and Poli, 1996). On the *Supply* side, the movement is individually represented on a microscopic basis in the DRACULA environment (Liu et al., 1995), which implements both car-following and lane-changing models. *Dracprep* prepares the traffic environment conditions on each day whereas *Dracsim* simulates drivers moving throughout the network.

In order to illustrate the potential application of such a framework, let us consider a specific *its-agent*: the ATIS agent. The main aim of the *ATIS agent* is to provide drivers with useful information on the existing traffic condition. For instance, subscribers of web-based traffic information applications could try to figure out traffic state before starting a journey.

The *MA Initialisation* module synthesises the *population* for the experiment from an OD matrix and route alternatives are assigned to each driver from a list of possible routes for each origin and destination pair. The initial set of base beliefs for each driver agent in the population is generated after a first run of the supply side. This is necessary to estimate the usual desired arrival time. The file resulting from this initialisation contains the belief set for each agent in a format that follows the specification of the SIM_Speak framework. The *Input MA* file gathers drivers' decisions on route and departure time, so that they can be launched onto the network to perform their journeys at selected departure times. Such decisions may have been influenced by the information provided by the *atis agent*, that keeps a global model of the traffic environment condition. On the other hand, the *Output MA* file returns the travel costs experienced by each driver in terms of realised travel time, in other words the perceptions of drivers during the course of the journey simulated in DRACULA; there is an updating of the base belief sets. On the following day, the driver uses its

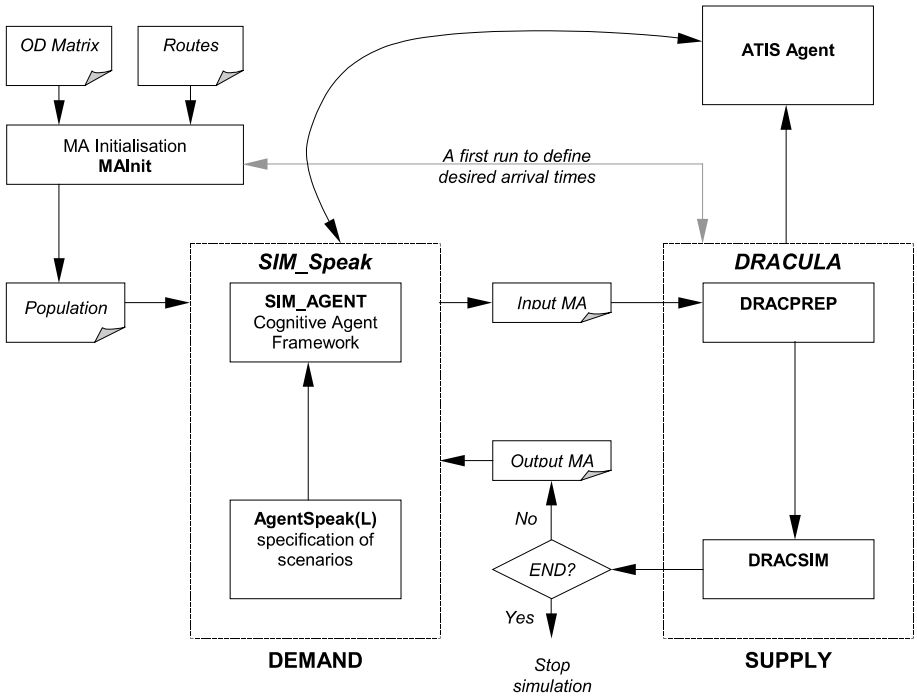

Fig. 4. A framework to implement BDI drivers in commuter scenarios.

updated beliefs to make its decision and this process is repeated over a specified number of days, which is defined at the beginning of the simulation.

Users of exogenous information sources may rely on the information provided in order to improve the quality of their decisions. Obviously, even making a request for information or receiving it through broadcasting, users are able to either accept or ignore its content. So, this framework contributes to assessing the impact that exogenous information may have in the overall performance of the traffic system, the quality of information provided, and the quality of decisions made by users. Therefore, the use of a cognitive representation is imperative as mental attitudes are fundamental parts of decision-making mechanisms.

6. Conclusions

As contemporary traffic systems become more complex, considering driver behaviour as a key component toward the system optimisation is quite inevitable. In this scenario ITS have played an indispensable role, where the improvement of existing road infrastructure is one of the underlying ideas. Thus, modelling the uncertainty and variability of human behaviour is now essential to traffic analysis in order to assess, for example, the impact that such new technologies may cause. However, incorporating human factors into simulation models has not been a trivial task. Although many aspects related to driver behaviour can be viewed from a reactive perspective, much

decision-making necessitates more robust and expressive means to represent the cognitive reasoning inherent in human beings.

The extension proposed on the basis of the DRACULA model allows for the assessment of variable demand in a commuter world represented as a multi-agent system. Profiting from the scalability and modularity offered by agent-based techniques, the relation among all the components within the system, including different ITS technologies can be represented. It makes the extended DRACULA model an appropriate framework to assess new measures of today's traffic scenario. In this way, we give special emphasis to the impact of information from exogenous sources, such as VMS and DRGS, in driver decision-making. So, drivers are intentional agents endowed with mental attitudes and the reasoning mechanism is represented in terms of a BDI architecture. We chose three scenarios focusing route and departure time choices to support our approach. Also, we presented an AgentSpeak(L) specification of commuters as an alternative to traditional representations of drivers, which fail to model reasoning capabilities. AgentSpeak(L) has enabled the representation of BDI agents in a clear and straightforward manner. Another useful and very important factor to bear in mind is the ability to represent new reasoning strategies through the addition of new plans. Although we have not specified any in our model, different ITS technologies can also be represented in terms of BDI architectures. AgentSpeak(L) allows an easy representation of communication capabilities of agents, so that co-operative and collaborative interactions can be represented as well.

However, one may argue such an approach would not be satisfactory in terms of computational performance, as the population of BDI drivers would be very high for certain experiments. Nonetheless, considering the expressive advances in the Computer Science field, both in hardware and in software, it would not represent a barrier to the potential of BDI-based approaches to model application domains such as traffic systems. Distributed and multi-threaded programming could be used for implementation purpose. We find that further attention should be given to AgentSpeak(L) by the multi-agent systems community, both as a programming and as a specification language. Also, practitioners from other areas could see AgentSpeak(L) as an specification language for application domains formed by intentional entities.

Acknowledgements

We would like to thank Dr. Helena B.B. Cybis, from LASTRAN, Universidade Federal do Rio Grande do Sul, and Rodrigo Machado, from II, Universidade Federal do Rio Grande do Sul, for their invaluable suggestions and comments on this work. This work has been partially supported by the Brazilian agencies CAPES and CNPq.

References

Bazzan, A.L.C., 1997. An evolutionary game-theoretical approach for coordination of traffic signal agents. Ph.D. Thesis, Karlsruhe University, Karlsruhe.

Bazzan, A.L.C., Wahle, J., Klügl, F., 1999. Agents in traffic modelling: from reactive to social behaviour. In: Advances in Artificial Intelligence (KI-99), LNAI 1701. Spring-Verlag, Berlin, pp. 303–306.

Bouchefra, K., Reynaud, R., Maurin, T., 1995. IVHS viewed from a multiagent approach point of view. In: Proceedings of the IEEE Intelligent Vehicles Symposium, Detroit. IEEE, Piscataway, pp. 113–117.

Burmeister, B., Haddadi, A., Matylis, G., 1997. Application of multi-agent systems in traffic and transportation. IEE Proceedings on Software Engineering 144 (1), 51–60.

Cantarella, G.E., Cascetta, E., 1995. Dynamic process and equilibrium in transportation networks: towards a unifying theory. Transportation Science 29 (4), 305–329.

Chatterjee, K., McDonald, M., Paulley, N., Taylor, N.B., 1999. Modelling the impacts of transport telematics: current limitations and future developments. Transport Reviews 19 (1), 57–80.

d'Inverno, M., Luck, M., 1998. Engineering AgentSpeak(L): A formal computational model. Journal of Logic and Computation 8 (3), 233–260.

Garrett, A., 1998. Intelligent transport systems: potential benefits and immediate issues. Road and Transport Research 7 (2), 61–69.

Haugeneder, H., Steiner, D., 1993. A multi-agent approach to cooperation in urban traffic. In: CKBS-SIG Workshop on Cooperating Knowledge Based Systems. pp. 83–99.

Liu, R., Van Vliet, D., Watling, D.P., 1995. DRACULA: dynamic route assignment combining user learning and microsimulation. In: Proceedings of the 23rd European Transport Forum, PTRC, vol. E. pp. 143–152.

Machado, R., Bordini, R.H., 2002. Running AgentSpeak(L) agents on SIM_AGENT. In: J.-J. Meyer and M. Tambe (Eds.), Intelligent Agents VIII—Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), 1–3 August 2001, Seattle, WA. LNAI 2333, Springer-Verlag, Berlin, pp. 158–174.

Machado, R., Bordini, R.H., 2001. SIM_Speak [online]. Available from: http://www.inf.ufrgs.br/~bordini/SIM_Speak/ [14 May 2001].

Mahmassani, H.S., Jayakrishnan, R., 1991. System performance and user response under real-time information in a congested traffic corridor. Transportation Research 25A, 293–308.

Pires, M., Dias, J., Belo, O., 1997. Using multi-agent systems technology on cooperative traffic control simulation: a case study. In: Proceedings of the 16th IASTED International Conference, Austria. pp. 241–244.

Rao, A.S., 1996. AgentSpeak(L) BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J.W. (Eds.), Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, LNAI 1038. Springer-Verlag, Heidelberg, pp. 42–55.

Rao, A.S., Georgeff, M.P., 1991. Modeling rational agents within a BDI architecture. In: Guru, R. (Ed.), Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-91). Morgan Kaufmann, San Mateao, pp. 473–484.

Roozemond, D.A., 1999. Using autonomous intelligent agents for urban traffic control systems. In: Proceedings of the 6th World Congress on Intelligent Transport Systems, Toronto, 8–12 November, 1999.

Rossetti, R.J.F., Bampi, S., Liu, R., Van Vliet, D., Cybis, H.B.B., 2000. An agent-based framework for the assessment of drivers' decision-making. In: The 3rd Annual IEEE Conference on Intelligent Transportation Systems, Dearborn, 1–3 October 2000. IEEE, Piscataway, pp. 387–392.

Russell, S.J., Norvig, P., 1995. Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs.

Sloman, A., Logan, B., 1999. Building cognitively rich agents using the SIM_AGENT toolkit. Communications of the Association of Computing Machinery 43 (2), 71–77.

Sloman, A., Poli, R., 1996. SIM_AGENT: A toolkit for exploring agent designs. In: Wooldridge, M., Müller, J.P., Tambe, M. (Eds.), Intelligent Agents II: Proceedings of the 2nd International Workshop on Agents Theories, Architectures, and Languages (ATAL'95), LNAI 1037. Springer-Verlag, Berlin, pp. 392–407.

Sycara, K., 1989. Multiagent compromise via negotiation. In: Distributed Artificial Intelligence, London, vol. 2. pp.119–137.

Tedesco, P.A., Self, J.A., 2000. MArCO: using conflict mediation strategies to support group planning interactions. Technical Report 00/4. Leeds: CBL, University of Leeds.

Watling, D.P., 1994. Urban traffic network models and dynamic driver information systems. Transport Reviews 14 (3), 219–246.