CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORT SCIENCES

JAN MACEK

AGENT-BASED MODELING OF ITS SYSTEMS IN VEHICLE SIMULATOR

DIPLOMA THESIS

2022

Abstract

TODO

Contents

1 Introduction

In a world in which vehicle transport plays an inseparable role to the society, even with an increasing demand, it is important to analyze and study driver behaviour and inherently interactions and relationships between drivers and their surroundings. This is supported by the fact that, even though substantial advancements in autonomous driving are being made, for the near-future, humans will still have to control vehicles themselves and therefore be exposed to a substantial risk of danger, as data shows that up to 90 % of traffic accidents are a result of human error. [citace] Each traffic accident has got a tremendous effect on socio-economic growth. A study be the European Union states that reduction of X % of traffic accidents could save X euros. [citace] A rather non-cynical point of view is that each life lost is a failure in society itself and effort should be made to diminish fatal accidents.

A method that has proven to be effective at studying driver behaviour and traffic safety is by using interactive vehicle simulators (IVS), which allow to undertake experiments in a safe, controlled and reproducible way. Because driving simulator is basically a digital twin of a real vehicle, it is naturally reasonable to make the interaction between the driver and IVS as close to reality as possible, which inherently improves data quality of the simulation and potentionally also the range of IVS application. An IVS has got a broad spectrum of employment. It is not only used as a tool to research driver behaviour, but also used in development and testing of advanced driver-assistance systems, extending the simulator to a hardware-in-the-loop or a vehicle-in-the-loop system, which enables to test real hardware and simulate full road testing.

Simulating the traffic environment is a complex problem, mainly because of its highly dynamic characteristics. All vehicles need to interact with each other and act upon other drivers' actions. Because agent-based simulations have proven to model complex behaviour well, this modelling technique seems like a suitable solution for achieving a realistic traffic environment for IVS.

The goal of this thesis is to investigate multi-agent systems, their characteristics and evaluate usages of these systems, including applications in the IVS research. Because agent-based systems are also suitable from modelling communication between entities, research regarding application of the agent-based systems in relation to ITS systems, which also share the characteristic of communication and interoperability between individual units within the system, will also be done.

After investigation of ABM in the IVS and ITS field, an experimental work is presented, which shows an implementation of a traffic (ITS) system using ABM, with self-proposed methodology and architecture, which will be described and implemented into an existing IVS. The proposed simulation system will then be examined and evaluated, deciding if the implementation was successful, if the systems adequately represent behaviour observed in real world and possibly performance of the system.

2 Multi-agent systems

Multi-agent systems is a broad paradigm and/or research topic. It is a subfield of Distributed problem solving. A multi-agent systems can be, in a short way, described as a group of autonomous agents that act towards their objectives in an environment to achieve a common goal [1]. The agents can be defined as independent units in an environment, forming a system. They are able to act independently, possess knowledge and communicate with each other (i.e. share the knowledge). The agents should be working towards some form of a common goal, which could be achieved either by cooperating or competing. The agents usually have a perception through which they can gain knowledge from their environment.

The basic definition of MAS suggests that they should be used to model/represent a system that is not-centralized and rather distributed, where autonomous, intelligent units perform actions independently. Multi-agent systems have gained popularity in the recent years, as they offer high flexibility of modelling highly non-linear systems and offering abstraction levels that make it more natural to deal with scale and complexity in these systems [2]. It is apparent that MAS excel in modeling social environments involving humans. MAS share a lot of features with human societies, as these societies also revolve around atomic units - persons, that act independently - each person makes decisions and acts upon their own beliefs. Persons also interact with each other, be it communication, cooperation or competition, while working towards some goal. There are numerous real-life examples that strongly resemble this MAS definition, for example sport teams, where each player has his own role, like a defender and attacker. Although all players have the same intention (i.e. win the game), their specific action differ depending on the state of their environment, each of them acting upon their own beliefs, which makes the team resemble a distributed system. From more practical perspective, MAS paradigm can be used when building air traffic management, for example.

2.1 Agent

Agents are the fundamental building blocks of a multi-agent system. While they can have many features and characteristics specific for their use case and are not possible to generalize, there are several elementary characteristics [1] that define them in scope of MAS.

Situatedness - Agents are designed so that they interact with the environment through sensors, resulting in actions using actuators. The agent should be able to directly interact with its environment using actuators.

Autonomy - Agent is able to choose its actions without other agents' interference on the network.

Inferential capability - Agent is able to work on an abstract goal specifications, identifying and utilizing relevant information it gets from observations.

Responsiveness - Agent is able to respond to a perceived condition of environment in a timely fashion.

Social behaviour - Agent must be able to interact with external sources when the need arises, e.g. cooperating and sharing knowledge.

2.2 Agent type and architecture

As has been already stated, in order to model complex applications, the agent characteristics as well as their internal control architecture will differ between use cases. In an effort to standardize MAS development, several architectures that describe how agents work have been proposed. Generally, there are three classses of agent modelling architectures that are defined based on interaction complexity with external sources:

- Reactive agents
- Deliberative agents
- Interacting agents

An overview of the class characteristics and examples of agent architectures utilizing the respective classes will be given.

2.2.1 Reactive agent architectures

Having first emerged in behaviorist psychology, the concept of reactive agents id founded by agents that make their decisions based on limited information, with simple situation-action rules. The agents usually make decisions directly based on the input from their sensors. This type of agents is mostly suited for application where agent resilience and robustness is the most important factor, instead of optimal behaviour. An example of a reactive agent architecture is the Subsumption architecture.

Subsumption architecture
This architecture described by Rodney Brooks in 1986 [3] and decomposes the agent into hierarchical levels that operate in a bottom-up fashion, meaning that bottom layers that control elementary behaviour are activated by the upper layers that define more complex action or define goals for the agent. It is important to note that the behavioral modules map sensations directly to actions and can only define what the agent does, not being able to change its desires. An example of a subsumption architecture is depicted in fig. (1) with example modules from each hierarchical level, where the bottom-level module is `Avoid Objects`, which is a needed action in order to complete the `Wander Around` action, which can be induced by the general goal on the top layer `Explore World`.

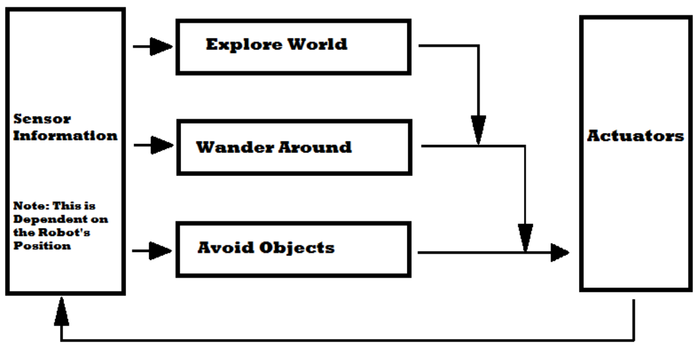


Figure 1: Example of a Subsumption architecture [4]

2.2.2 Deliberative agent architectures

Compared to a reactive agent, deliberative agents have a more complex structure and is closer to a human-like, rational behaviour. A deliberative agent is defined as one that possesses an explicitly represented, symbolic model fo the world, and in which decisions are made via symbolic reasoning [5]. In other words, the agent maintains its internal representation of the external environment and thus capable to plan its actions, while being in an explicit mental state which can dynamically change. The Beliefs, Desires and Intentions (BDI) architecture is the most widely known modelling approach of deliberative agents.

BDI architecture

The BDI paradigm has been used in various applications, such as simulating impacts of climate change on agricultural land use and production [6] or to improve internet network resilience by creating BDI agents that combats DDoS attacks [7]. The main idea behind this architecture is the emphasis on practical reasoning - the process of figuring out what to do. There are three logic components that characterize an agent:

Beliefs - The internal knowledge about the surrounding environment, which is being constantly updated by agent's perception.

Desires - What the agent wants to accomplish. An agent can have multiple desires, which can be hierarchically structured or have different priority.

Intentions - Intentions are formed when an agent commits to a plan in order to achieve a chosen goal. The plans are pre-defined within an agent, formally called a *plan library.* The plan that an agent has set to carry out can dynamically change based on updated beliefs or desires.

These components together define an agent's *reasoning engine* (fig. (2)), which drives the agent's (deliberative) behaviour.

This definition can be made clearer with a simple example scenario - a waiter in a restau-

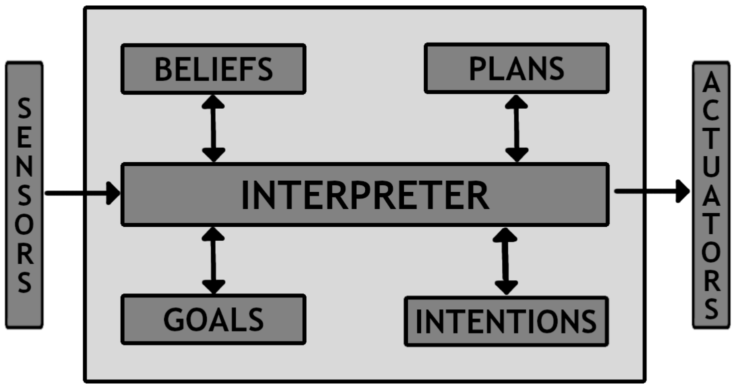


Figure 2: BDI architecture schematic

rant. The waiter's *beliefs* are the tables with customers and information about state of each table (i.e. choosing menu, waiting for meal, willing to pay etc.). The waiter's *desires* are to serve customers, for example accept order from a customer. The waiter carries out his desires by making a plan of *intentions* (e.g. go to table and ask if the customer wants a drink).

This architecture has its advantage in the fact that the functional decomposition of the system is clear and intuitive. However, with this architecture, there is a commitment-reconsideration tradeoff that needs to be optimized [8]. With too much commitment, there is a risk of agent overcommitment, where an agent might be trying to achieve a goal that is not longer valid. On the other hand, if an agent reconsiders too often, there is a risk that the agent will not achieve any goal because it will switch between intentions too quickly.

2.2.3 Hybrid approaches

Hybrid architectures try to utilize the best of both worlds of agent modelling. Purely reactive agents might lack the ability to solve complex tasks, whereas deliberative architectures are challenging to successfully implement on a concrete problem [9]. Pre-compiling a plan library for every possible scenario that can happen in environment with vast amount of complexities is simply not feasible and even impossible, due to uncertainties of following effects when agents affect the environment.

The underlying concept of hybrid architectures is structuring agent functionalities into layers that interact with each other, which provides several advantages, namely *modularization*, which decomposes the agent's functionalities into distinct modules that have determined interfaces, which facilitates design complexity. Furthermore, having distinct layers enables them to run in parallel, thus increasing an agent's computational ability and also reactivity [10].

Generally, amongst the most widely used hybrid architectures, a *controller* layer can be found, which handles reactive tasks therefore is also connected to the sensor readings, and is hierarchically on the lowest level. Then, there are *planning* layers that handle the logic-based, deliberative tasks and often interact with the controller layer. In-between them, there is usually a *sequencing* layer that can suppress output from the reactive layer. An example of a well-known hybrid architecture is the $_3$T architecture, whose abstract model can be seen on the figure (3) below.

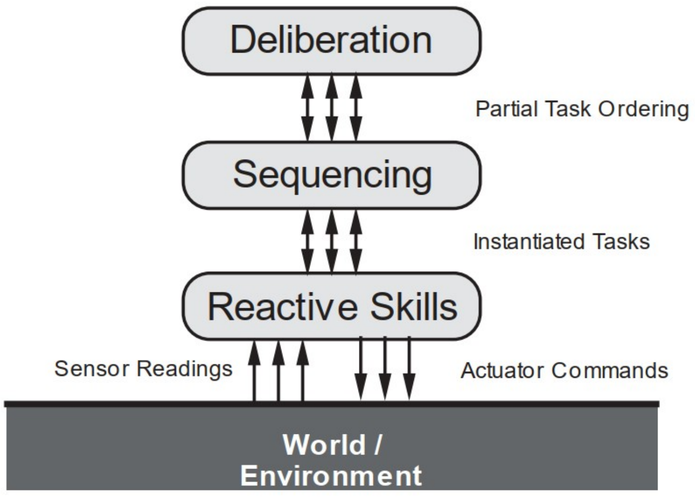


Figure 3: An architecture model of a $_3$T hybrid architecture [11]

$_3$T architecture

This architecture builds upon a predecessor architecture called RAPs (Reactive Action Packages) [12]. A RAP is essentially a process or a description of how to complete a task using discrete steps. Note that it has got no planning abilities, i.e. it's action is only based on current perceived environment state and not on an anticipated state. When a RAP is executed, it should finish only when it satisfied its outcome or else it will produce a failure state. This ensures that the agent can self-diagnose a failure and implement some fail-safe mechanisms. Individual RAPs are queued by the interpreter, in the case of the $_3$T architecture it is called a *sequencer*, which is the intermediate layer between the *reactive skills* and *deliberation* layer.

The skills layer is a collection of so-called *skills*. Skills provide an interface of an agent to its environment. They are its abilities that allow the agent to transform or maintain a particular state in the environment. Each skill has got an expected input and output, which allows to route them together.

Finally, the deliberation layer is responsible for planning on a high level of abstraction, in order to make its problem space small. Routine sequences of tasks should not be specified or dealt with at this layer.

2.2.4 Conclusion

The different architecture types that were presented can each have a different application where they excel. It is also important to note that a line between reactive and deliberative agents is not necessarily strict, hence the existence of hybrid architecture types.

In my opinion, choosing the optimal agent architecture will depend on the scope of agent goals, environment and action space definition. With respect to the topic of this thesis, which is to implement ITS systems, such as cooperative vehicles or traffic lights and such, it is quite clear that fast response time and predictable behaviour in a dynamic environment would be suited more for *reactive*, possibly *hybrid* architectures, as deliberative agents tend to perform worse in highly dynamic environments where frequent re-planning is required.

It safe to say that an agent-based ITS system will require some form of agent interaction and thus communication between individual agents, because it is more than likely that all agents won't be competing with each other, but rather trying to cooperate to achieve a common goal.

2.3 Interaction between agents

Interacting agents are able to interact with other agents within the environment. This concept extends an agent with an interface dedicated to communication, which makes them able to directly communicate and thus cooperate in a decentralized fashion. The concept of cooperation between agents is important in the ITS modelling context, as these systems facilitate sharing information in-between drivers and also from the road traffic environment to drivers. Therefore, interaction is a strong component when considering modelling ITS and road traffic in general. However, this interface also adds to the system's complexity.

One should adhere to some communication principles to facilitate cooperation between agents. As such, agents should communicate according to Gricean maxims (see the table below) [13]. This way, the communication and performance overhead is minimized and system is more stable.

Table 1: Gricean maxims

Quantity	Say not more nor less than it is required
Relevance	Stay relevant to the topic of discussion
Manner	Avoid obscurity and ambiguity
Quality	Do not give false or unsupported information

It needs to be said that the agents developed for an ITS system will most likely not be competing amongst each other, as the agent reward should be higher the more the agents cooperate. For example, when designing cooperative intersections system, the reward function for the individual intersections (i.e. agents) should take into account not only the throughput on their own intersection but also throughput of the whole system. However, that doesn't eliminate the need to negotiate. Consider a situation where there are multiple traffic light intersections, each deciding on which phases to enable, based on the incoming traffic intensity. An optimal option that minimizes cost (e.g. travel time) for one agent could have a significant negative effect on other intersections. Therefore it is important to achieve an equilibrium that minimizes the overall cost.

2.3.1 Organizational decomposition

Before defining the communication protocol, which is arguably the most important aspect of agent interaction, it is important to think about the the overall system topology and inter-agent relationships. There are two possibilities when designing agent structure.

Hieararchical organization

In hierarchical organization [14], agents are organized in a tree structure, where each its level has got a different level of autonomy. The flow of control is from top to bottom, i.e. agents in lower level of hierarchy conform to decisions from higher levels.

A simple form of hierarchical organization ensures that there is a low number of conflicts and the system can be operating by relatively simpler sequential processes where the control flow is straightforward, however, this also decreases the robustness of the system, because the control and autonomy not as distributed, e.g. when a failure of a single agent with at a high hierarchical level causes the whole system to fail. A subtype of this organization - a uniform hierarchy, the authority is more distributed among the agents. This makes the system more fault tolerant and perform graceful degradation in scenarios where one ore more parts of the system fail [1]. Here, a special attention needs to be given to conflict resolution, which is not always as straightforward, as mentioned earlier.

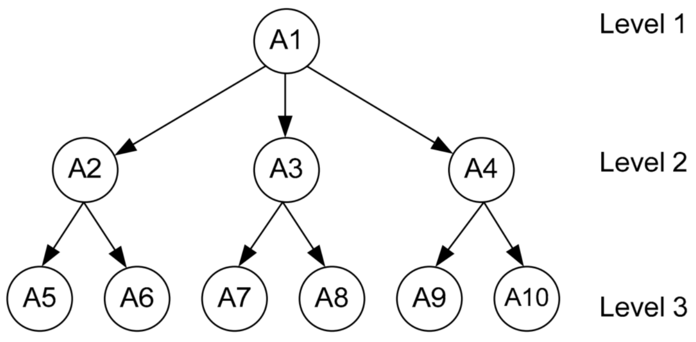


Figure 4: Hierarchical agent structure schematic [1]

Coalitions

Another useful agent organization is to organize into coalitions. In coalitions, a group of agents come together for a short time to increase the utility or performance of the individual agents in a group [1]. After the goal is reached or the coalition no longer becomes feasible, the coalition ceases to exist. The coalition can have a flat hierarchy, with the possibility to have one agent as the leader role for interactions outside the coalition. The use of coalitions allows to make a highly dynamic system, which also increases the system complexity, though.

There are more concepts on structuring a multi-agent system from organizational point of view, sych as teams and holons. However, the two mentioned approaches are the most well-suited for an ITS system development, as they have been used in more works for this purpose already (e.g. [15] and [16]).

In conclusion, the usage of the aforementioned architecture should ensure that the system complexity is kept as low as possible and the topology is transparent and uncluttered. The hierarchical architecture can be beneficial when applied to ITS systems such as urban traffic management, where conflicts of interests need to be resolved by a capable authority. Whereas, the coalition-based organization could be applied to create virtual clusters of so-called "connected" vehicles to apply C-ITS features, e.g. dynamic navigation or GLOSA.

2.3.2 Communication between agents

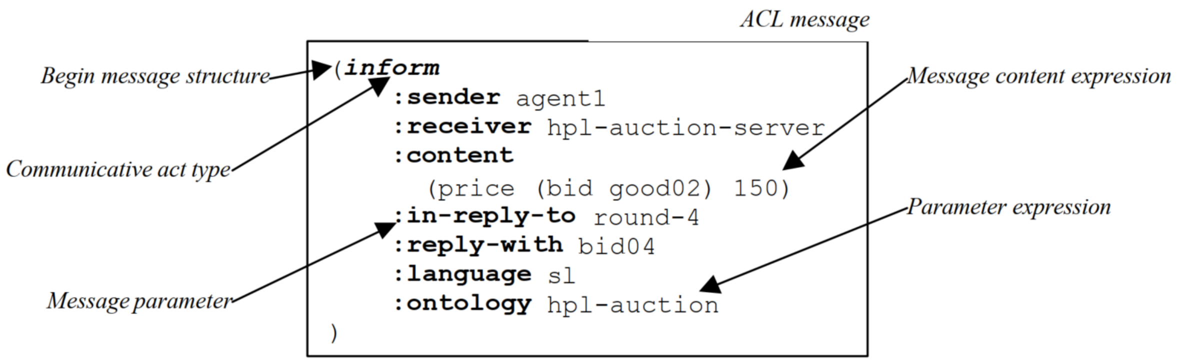
As has been stated before, agent communication is a crucial component of a multi-agent system. Communication makes for inter-agent interaction beyond cues that an agent receives from the environment through its sensors, as agents can either exchange or broadcast information which could not be obtainable for the agent otherwise. This allows for deeper and more complex decision making and behaviour design - agents can act upon the received information or update their beliefs about the world and provide distributed problem solving in general.

2.3.3 Agent Communication Language

Any language, including the one used by agents in an arbitrary system, should have defined its syntax and semantics to be an effective medium. Effectively, this means that a suitable communication interface for each agent should be defined, with a standardized way of expressing information. To satisfy these requirements a language developed specifically for MAS has been created, called *ACL - Agent Communication Language* [17]. This specification proposes a standard for agent communication. Most importantly, it defines so-called *communicative act (CA)* - a special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act. CAs are modelled on speech act theory. Pragmatically, CA's are performed by an agent sending a message to another agent, using a specified message format. For the index of the defined CAs and their classification, see the table (2).

Using all of the defined message types is not mandatory in order to use the ACL. However, the standard introduces ACL-compliant agent requirements that need to be fulfilled. The agent requirements are in the table (3).

Apart from the mentioned communicative acts, the ACL standard also defines message parameters, which make the structure of a message (table (4)). A sample composed message structure can be found in figure (5).



```
(inform
    :sender agent1
    :receiver hpl-auction-server
    :content
      (price (bid good02) 150)
    :in-reply-to round-4
    :reply-with bid04
    :language sl
    :ontology hpl-auction
)
```

Figure 5: Main structural components of ACL message [17]

It needs to be considered that there is plenty of other standard definitions that are suited for distributed system interoperability, such as W3C, CORBA, UML and many others, so the choice of the ACL needs to be argued. A lot of these applications are based on the so called CRUD set of communication primitives - Create, Read, Update and Delete. In contrast to this, ACL defines a lot of different communication primitives (i.e. communicative acts). This leads to more complex control [18]. However, The MAS developed in this thesis will be in a closed, virtual space, which will most likely not require integration with other services. This can be further leveraged to our advantage by stripping the language of some redundant message parameters like the `:transport protocol`, `:reply-by`, `:envelope` and other message types, which are not needed because communication will be abstraced away and virtualized within the simulation software. Also, [18] states that The ACL model naturally allows more semantic context to be included in messages. This can give applications more understandable information about unexpected events. In addition, the richer set of primitives can lead to more flexible interaction processes.

2.3.4 Conclusion

In this section, the ways how agents interact with each other were described, with respect to the application scope of this thesis, i.e. applying MAS on an ITS system simulation. The most important facts are that agent agent organization needs to be considered -

despite the fact that MAS are designed to be highly distributed, giving the agents hierarchical roles can often facilitate interaction between agents and can decrease system complexity while preserving functionality, especially during negotiation between agents. Agent grouping should, while also contributing to decreased system distribution, decrease computation and communication overhead. Next, the way how agents should communicated was described. The cornerstone of inter-agent communication was discussed, and a potentially suitable framework (for the thesis' use-case) of setting up communication between agents was investigated. Overall, it was found that designing how agents communicate is a crucial part of MAS development that can get complex, so it is important to keep the ways of sharing information as organized as possible, which can be done by defining the language, primarily its semantics and associated syntax. On the other hand, it was also considered that because the system will be designed and run in a virtual space, some intricacies can be abstracted away by the simulation engine.

Table 2: Categories of communicative acts [17]

Communicative act	**Information passing**	**Requesting information**	**Negotiation**	**Action** performing	**Error** handling
`accept-proposal`			■		
`agree`				■	
`cancel`				■	
`cfp`			■		
`confirm`	■				
`disconfirm`	■				
`failure`					■
`inform`	■				
`inform-if (macro act)`	■				
`inform-ref (macro act)`	■				
`not-understood`					■
`propose`			■		
`query-if`		■			
`query-ref`		■			
`refuse`				■	
`reject-proposal`			■		
`request`				■	
`request-when`				■	
`request-whenever`				■	
`subscribe`		■			

Table 3: The ACL-compliant agent requirements [17]

Requirement 1: Agents should send not-understood if they receive a message that they do not recognise or they are unable to process the content of the message. Agents must be prepared to receive and properly handle a not-understood
Requirement 2: An ACL compliant agent may choose to implement any subset (including all, though this is unlikely) of the predefined message types and protocols. The implementation of these messages must be correct with respect to the referenced act's semantic definition.
Requirement 3: An ACL compliant agent which uses the communicative acts whose names are defined in the specification must implement them correctly with respect to their definition.
Requirement 4: Agents may use communicative acts with other names, not defined in the specification document, and are responsible for ensuring that the receiving agent will understand the meaning of the act. However, agents should not define new acts with a meaning that matches a pre-defined standard act.
Requirement 5: An ACL compliant agent must be able to correctly generate a syntactically well formed message in the transport form that corresponds to the message it wishes to send. Symmetrically, it must be able to translate a character sequence that is well-formed in the transport syntax to the corresponding message.

Table 4: Pre-defined message parameters [17]

Message Parameter	**Meaning**
`:sender`	Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act.
`:receiver`	Denotes the identity of the intended recipient of the message.
`:content`	Denotes the content of the message; equivalently denotes the object of the action.
`:reply-with`	Introduces an expression which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously.
`:in-reply-to`	Denotes an expression that references an earlier action to which this message is a reply.
`:envelope`	Denotes an expression that provides useful information about the message as seen by the message transport service.
`:language`	Denotes the encoding scheme of the content of the action.
`:ontology`	Denotes the ontology which is used to give a meaning to the symbols in the content expression.
`:reply-by`	Denotes a time and/or date expression which indicates a guideline on the latest time by which the sending agent would like a reply.
`:protocol`	Introduces an identifier which denotes the protocol which the sending agent is employing.
`:conversation-id`	Introduces an expression which is used to identify an ongoing sequence of communicative acts which together form a conversation.

3 Intelligent Transport Systems

References

[1] Balaji Parasumanna Gokulan and D. Srinivasan. “An Introduction to Multi-Agent Systems”. In: vol. 310. July 2010, pp. 1–27. ISBN: 978-3-642-14434-9. DOI: 10.1007/978-3-642-14435-6_1.

[2] Birgit Burmeister, Afsaneh Haddadi, and Guido Matylis. “Application of multi-agent systems in traffic and transportation”. In: *IEE Proc. Softw. Eng.* 144 (1997), pp. 51–60.

[3] R. Brooks. “A robust layered control system for a mobile robot”. In: *IEEE Journal on Robotics and Automation* 2.1 (1986), pp. 14–23. DOI: 10.1109/JRA.1986.1087032.

[4] R.A. Brooks. *Cambrian Intelligence: The Early History of the New AI*. Bradford book. MIT Press, 1999. ISBN: 9780262024686. URL: https://books.google.cz/books?id=LZa3QgAACAAJ.

[5] M. Wooldridge. *What agents aren't: a discussion paper*. UNICOM Seminar on Agent Software, 1996. DOI: 10.1049/ic:19960648.

[6] Philippe Caillou et al. *A Simple-to-use BDI architecture for Agent-based Modeling and Simulation*. 2017. DOI: 10.1007/978-3-319-47253-9_2.

[7] Ingrid Nunes, Frederico Schardong, and Alberto Schaeffer-Filho. “BDI2DoS: An application using collaborating BDI agents to combat DDoS attacks”. In: *Journal of Network and Computer Applications* 84 (2017), pp. 14–24. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2017.01.035. URL: https://www.sciencedirect.com/science/article/pii/S1084804517300577.

[8] Michael Wooldridge and Simon Parsons. “Intention Reconsideration Reconsidered”. In: (1999), pp. 63–79. ISSN: 0302-9743. DOI: 10.1007/3-540-49057-4_5.

[9] Patricia Anthony et al. “Agent Architecture: An Overview”. In: *TRANSACTIONS ON SCIENCE AND TECHNOLOGY* (Jan. 2014), pp. 18–35.

[10] Jörg Müller. “Architectures and applications of intelligent agents: A survey”. In: *The Knowledge Engineering Review* 13 (Feb. 1999), pp. 353–380. DOI: 10.1017/S0269888998004020.

[11] R. Bonasso et al. “Experiences with an Architecture for Intelligent, Reactive Agents.” In: vol. 9. Jan. 1995, pp. 187–202. DOI: 10.1080/095281397147103.

[12] R. James Firby. “An Investigation into Reactive Planning in Complex Domains”. In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*. AAAI'87. Seattle, Washington: AAAI Press, 1987, pp. 202–206. ISBN: 0934613427.

[13] Yoav Shoham and Kevin Leyton-Brown. *Multiagent SystemsAlgorithmic, Game-Theoretic, and Logical Foundations*. DOI: 10.1017/cbo9780511811654.020.

[14] Ariuna Damba and Shigeyoshi Watanabe. “Hierarchical Control in a Multiagent System”. In: (2007). DOI: 10.1109/icicic.2007.334.

[15] P. G. Balaji et al. “Multi-agent System based Urban Traffic Management”. In: (2007). DOI: 10.1109/cec.2007.4424683.

[16] Michael Vijsel and John Anderson. “Coalition Formation in Multi-Agent Systems under Real-World Conditions”. In: *AAAI Workshop - Technical Report* (June 2004).

[17] Foundation for Intelligent Physical Agents. *Agent Communication Language Specification*. 2001. URL: http://www.fipa.org/specs/fipa00018/.

[18] Stefan Poslad. “Specifying Protocols for Multi-Agent Systems Interaction”. In: 2 (2007), p. 15. ISSN: 1556-4665. DOI: 10.1145/1293731.1293735.