Observable and Attention-Directing BDI Agents for Human-Autonomy Teaming

Blair Archibald Muffy Calder Michele Sevegnani Mengwei Xu
School of Computing Science, University of Glasgow, Glasgow, UK
{blair.archibald, muffy.calder, michele.sevegnani, mengwei.xu}@glasgow.ac.uk

Human-autonomy teaming (HAT) scenarios feature humans and autonomous agents collaborating to meet a shared goal. For effective collaboration, the agents must be transparent and able to share important information about their operation with human teammates. We address the challenge of transparency for Belief-Desire-Intention agents defined in the Conceptual Agent Notation (CAN) language. We extend the semantics to model agents that are *observable* (*i.e.* the internal state of tasks is available), and *attention-directing* (*i.e.* specific states can be flagged to users), and provide an executable semantics via an encoding in Milner's bigraphs. Using an example of unmanned aerial vehicles, the BigraphER tool, and PRISM, we show and verify how the extensions work in practice.

1 Introduction

Autonomous agents *e.g.* robots [27] are becoming increasingly. As the autonomy increases, so does the tasks that can be performed [28], and more emphasis is being placed on how agents can form teams, alongside humans, to achieve a common goal, a.k.a. human-autonomy teaming (HAT) [32, 19].

Transparency is especially important in HAT. While there is no single definition [38, 33], we draw on IEEE P7001 standard definition [15] that transparency is measurable, testable, and is "the transfer of information from an autonomous system or its designers to a stakeholder, which is honest, contains information relevant to the causes of some action, decision or behaviour and is presented at a level of abstraction and in a form meaningful to the stakeholder." Here, we focus on two transparency requirements: *observability* of internal states and *attention-directing* of a human teammate.

To illustrate transparency, consider a HAT scenario where a human and an Unmanned Aerial Vehicle (UAV) survey a site together. The UAV performs site surveys through imaging and retrieves objects if needed, while the human assembles and interprets geographically related information (obtained from the UAV). To coordinate an effective exploration, it is essential to have the appropriate transparency into what tasks either is performing and the progress of the relevant tasks. For example, the human needs to know if the UAV has finished surveying an area, or how much is left, before assembling analysis for the same area. Transparency must also be supported to allow attention-direction during autonomy failure (*e.g.* the engine is malfunctioning) and to highlight changes in the environment, *e.g.* when a specific land condition is detected by the UAV the human should be notified immediately for inspection.

We tackle the challenge of transparency by modelling and verifying agents that are *observable* [5], *i.e.* the status of tasks is known, and *attention-directing*, *i.e.* specific states can be flagged to users. Both observability and attention-directing are recommended features in a practical HAT system engineering guide [17], though strictly this guide refers to the former as transparency and the latter as augmenting cognition, which we consider part of transparency.

To demonstrate these features, we extend a model of Belief-Desire-Intention (BDI) agents proposed in [2] that presents an executable semantics of the CAN agent language [40] based on Milner's Bigraphs [18]. BDI agents [6, 22] consist of, (B)eliefs: what the agent knows; (D)esires: what the agent

M. Farrell & M. Luckcuck (Eds.):
Third Workshop on Formal Methods
for Autonomous Systems (FMAS2021)
EPTCS 348, 2021, pp. 167–175, doi:10.4204/EPTCS.348.12

wants to bring about; and (I)ntentions: the desires the agent is currently acting upon. BDI agents are chosen as they are represented in a declarative fashion (easy to log) with some self-explanatory concepts *e.g.* the beliefs of the agent. CAN is chosen as it features a high-level agent programming language that captures the essence of BDI concepts without describing implementation details such as data structures. As a superset of AgentSpeak [21], CAN includes advanced BDI agent behaviours such as reasoning with *declarative goals*, *concurrency*, and *failure recovery*, which are necessary for our UAV example modelled in Section 3. Importantly, although we focus on the CAN, the language features are similar to those of other mainstream BDI languages and the same modelling techniques would apply to other BDI programming languages. Besides the work of [7] that focuses on transparent ethical reasoning (*i.e.* why a decision has been made by an agent), we believe this is the first formal analysis of transparency applied to mainstream BDI agents.

We make the following research contributions: (1) an extension of CAN language semantics to support transparency, (2) an extension of the bigraph based executable semantics framework, (3) an evaluation based on Unmanned Aerial Vehicles (UAVs) to illustrate the framework.

2 Framework

In this section, we provide transparency mechanisms for *observability*, *i.e.* documenting the status of tasks and their progress toward completion, and *attention-directing*, *i.e.* flagging information about specific states—of the situation or agent itself—to a user. We start with some CAN language background.

2.1 CAN Background

The CAN language formalises a classical BDI agent consisting of a belief base $\mathscr{B}$ and a plan library Π. The belief base $\mathscr{B}$ is a set of formulas encoding the current beliefs and has belief operators for entailment (*i.e.* $\mathscr{B} \models \varphi$), and belief atom addition (resp. deletion) $\mathscr{B} \cup \{b\}$ (resp. $\mathscr{B} \setminus \{b\}$) and any logic over the belief base $\mathscr{B}$ is allowed providing entailment is supported. A plan library Π is a collection of plans of the form $e : \varphi \leftarrow P$ with e the triggering event, φ the context condition, and P the plan-body. Events can be either be external (*i.e.* from the environment in which the agent is operating) or internal (*i.e.* sub-events that the agent itself tries to accomplish). In the plan-body, we use $P_1;P_2$ for sequence and $goal(\varphi_s, P, \varphi_f)$ for the declarative goal failing if φ_f holds and exiting successfully if φ_s holds (see [25]).

A basic configuration $\langle \mathscr{B}, P \rangle$, where P is the plan-body being executed (*i.e.* the current intention), is used in rules that define the execution of a single intention. The agent configuration is defined as $\langle E^e, \mathscr{B}, \Gamma \rangle$ where E^e denotes the a set of pending external events and Γ the current set of intentions (*i.e.* partially executed plan-body programs). The agent-level evolution is specified by the transitions over the agent configuration. For example, the *agent-level* transition to progressing intention which is progessable ($\langle \mathscr{B}, P \rangle \rightarrow \langle \mathscr{B}', P' \rangle$) or dropping *any* unprogressable intention ($\langle \mathscr{B}, P \rangle \nrightarrow$) can be given as follows:

$$\frac{P \in \Gamma \quad \langle \mathscr{B}, P \rangle \rightarrow \langle \mathscr{B}', P' \rangle}{\langle E^e, \mathscr{B}, \Gamma \rangle \Rightarrow \langle E^e, \mathscr{B}', (\Gamma \setminus \{P\}) \cup \{P'\} \rangle} A_{step} \qquad \frac{P \in \Gamma \quad \langle \mathscr{B}, P \rangle \nrightarrow}{\langle E^e, \mathscr{B}, \Gamma \rangle \Rightarrow \langle E^e, \mathscr{B}, \Gamma \setminus \{P\} \rangle} A_{update}$$

We refer the reader to [40, 25] for a full overview of the semantics of CAN.

2.2 Observability

Observability captures *what* an agent is doing so that human teammates can coordinate their tasks for effective collaboration. As intention represents tasks, we focus on the *status* and *progress* of intentions.

$$\frac{e \in E^e}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e \setminus \{e\}, \mathscr{B}, \Gamma \cup \{e\}\rangle} A_{event}$$

$$\frac{\langle e, I, \texttt{pending}\rangle \in E^e}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e \setminus \{\langle e, I, \texttt{pending}\rangle\} \cup \{\langle e, I, \texttt{active}\rangle\}, \mathscr{B}, \Gamma \cup \{\langle e, I\rangle\}\rangle} A^{new}_{event}$$

$$\frac{P \in \Gamma \quad \langle \mathscr{B}, P\rangle \rightarrow \langle \mathscr{B}', P'\rangle}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e, \mathscr{B}', (\Gamma \setminus \{P\}) \cup \{P'\}\rangle} A_{step}$$

$$\frac{\langle P, I\rangle \in \Gamma \quad \langle \mathscr{B}, \langle P, I\rangle\rangle \rightarrow \langle \mathscr{B}', \langle P', I\rangle\rangle}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e, \mathscr{B}', (\Gamma \setminus \{\langle P, I\rangle\}) \cup \{\langle P', I\rangle\}\rangle} A^{new}_{step}$$

$$\frac{P \in \Gamma \quad \langle \mathscr{B}, P\rangle \nrightarrow}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e, \mathscr{B}, \Gamma \setminus \{P\}\rangle} A_{update}$$

$$\frac{\langle P, I\rangle \in \Gamma \quad \langle e, I, \texttt{active}\rangle \in E^e \quad \langle \mathscr{B}, \langle P, I\rangle\rangle \nrightarrow \quad P = nil}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e \setminus \{\langle e, I, \texttt{active}\rangle\} \cup \{\langle e, I, \texttt{success}\rangle\}, \mathscr{B}, \Gamma \setminus \{\langle P, I\rangle\}\rangle} A^{new}_{update\_suc}$$

$$\frac{\langle P, I\rangle \in \Gamma \quad \langle e, I, \texttt{active}\rangle \in E^e \quad \langle \mathscr{B}, \langle P, I\rangle\rangle \nrightarrow \quad P \neq nil}{\langle E^e, \mathscr{B}, \Gamma\rangle \Rightarrow \langle E^e \setminus \{\langle e, I, \texttt{active}\rangle\} \cup \{\langle e, I, \texttt{failure}\rangle\}, \mathscr{B}, \Gamma \setminus \{\langle P, I\rangle\}\rangle} A^{new}_{update\_fail}$$

Figure 1: Derivation rules for agent configuration.

2.2.1 Intention Status

As a high-level planning language, CAN is agnostic to many practical issues – including transparency. One issue of CAN is the inability to tell the status of an intention. For example, the rule A_{update} in Fig. 1 discards an intention that cannot do any more intention-level transitions: both if it has already succeeded, or if it failed, and there is no way to determine which was the case. In practice, we may find it helpful, particularly in HAT, to have precise knowledge on the success or failure status of an intention.

We extend CAN semantics to allow intention status. Following work [13], we introduce four status values for an external event (that ultimately gives rise to an intention): `pending`, `active`, `success`, and `failure` along with an unique identifier I. These values indicate when an event is not addressed yet (`pending`), currently being addressed (`active`), successfully addressed (`success`), and addressed with a failure (`failure`). Unique identifiers enable the agent to track the means-end relations between the events and the related intentions, and differentiate intentions from others.

Figure 1 presents the original rules and our status-enabled rules where $\langle E^e, \mathscr{B}, \Gamma\rangle$ consists of a set of external events E^e that the agent is required to respond, a belief set $\mathscr{B}$, and intention base Γ is a set of partially executed plan-bodies P that the agent has committed to. While the original rule A_{event} deletes events once they have turned into intentions, the new rule A^{new}_{event} instead switches the status of the event to `active`. When intentions progress, the status of related external events remains `active`, until the intention is removed either with `sucess` (if we reached a *nil*) via $A^{new}_{update\_suc}$ or `failure` via $A^{new}_{update\_fail}$.

2.2.2 Intention Progress

Besides the intention status, it is also useful to estimate the progress of intentions. For example, the human teammate may want to know how long the UAV needs to finish surveying to plan their next tasks.

In BDI agents, goal-plan trees (shown in Fig. 2) are a canonical representation of intentions [34]. The root of the tree is an external event, and its children are plans that can handle this event. Plans may contain sub-events, giving rise to a tree structure that represents all possible ways of achieving a task. Traditionally, for simplicity, goal-plan trees often do not represent actions and contain only

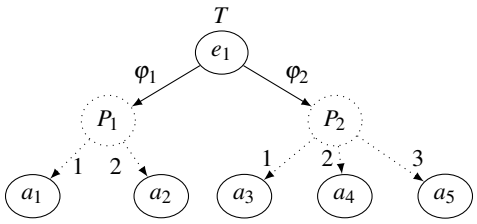

Figure 2: A goal-plan tree.

goals/subgoals and plans *e.g.* in [34]. However, we need to include the actions—a crucial part in agent execution—to perform faithful intention progress estimation. Finally, executing a BDI program gives *one* execution trace[1] [41], *i.e.* a path through the tree.

To illustrate goal-plan trees and execution traces, consider the following two plans as follows:

$$P_1 = e_1 : \varphi_1 \leftarrow a_1; a_2; \quad P_2 = e_1 : \varphi_2 \leftarrow a_3; a_4; a_5.$$

These two plans are visualised as the goal-plan tree T given in Fig. 2. The goal-plan tree T expresses that, given an event e_1, it has two plan choices, namely plan P_1 and P_2, where the edges between the event e_1 and plan P_1 and P_2 are labelled with the relevant context condition φ_1 and φ_2. If plan P_1 is selected (given φ_1 holds), it has two sequenced actions a_1 and a_2 to execute where the edge between the plan to its plan-body is labelled with the position number. As such, we can obtain one execution trace $e_1; P_1; a_1; a_2$ and the other one (due to the selection of plan P_2) $e_1; P_2; a_3; a_4; a_5$.

For intention progress estimation, we use a compile-time process to obtain all possible execution traces. For each trace, we record the maximum length of the execution trace. During execution, the agent tracks its current execution trace so that after an agent-step we can determine the maximum length of the full trace this current trace belongs to[2]. The progress of an intention is then estimated as the ratio of the position of the last element in the current trace and the length of the full trace this current trace belongs to. For example, given a trace $e_1; P_1; a_1; a_2$, its maximum length is 4 and the position for each element in this trace is 1 for e_1, 2 for P_1, 3 for a_1, and 4 for a_2. If the current execution trace is $e_1; P_1; a_1$ then the progress estimation is $3/4 = 75\%$ (as a_1 is the last element). In the case of failure recovery, for example, if action a_1 failed, the agent backtracked to event e_1 and subsequently selected the plan P_2. Then the current trace becomes $e_1; P_2$ (against the full trace $e_1; P_2; a_3; a_4; a_5$) with the progress estimation $2/5 = 40\%$. The execution trace is maintained separately for each intention.

2.3 Directing Attention

Agents should be able to direct the attention of teammates when particular states become relevant, *e.g.* when adversarial situations are encountered. In BDI programming languages such as AgentSpeak, a change in the belief base generates an event, which subsequently requires an agent to select a plan. For example, if a belief atom b is true, an event in the form of $+b$ is added to the desires. While useful, this one-to-one mapping between belief and event is also limiting, especially in the context of directing attention *e.g.* it prohibits the generation of multiple events. Further, multiple events generated by a single belief change may require adjustment throughout the operational life of the agent.

[1] For practicality, we do not allow recursive plans as this can lead to infinite traces.

[2] To ensure each trace is uniquely identifiable, the same action occurring in a different plan is treated as a different action.

```
1  // Initial beliefs
2  ¬sensor_malfunc, ¬engine_malfunc
3  // External events
4  ⟨e_retrv, identifier1⟩
5  // Motivation library events
6  parked <- ⟨e_parked, identifier2⟩
7  // Plan library
8   e_retrv :  φ <- take_off; goal(at_destination, e_path1, fc); retrieve
9   e_retrv :  φ <- take_off; goal(at_destination, e_path2, fc); retrieve
10  e_retrv :  φ <- take_off; goal(at_destination, e_path3, fc); retrieve
11  e_retrv :  sensor_malfunc <- return_base
12  e_retrv :  engine_malfunc <- activate_parking
13  e_path1 :  true <- navigate_path_1
14  e_path2 :  true <- navigate_path_2
15  e_path3 :  true <- navigate_path_3
16 e_parked :  true <- send_GPS
where φ = ¬sensor_malfunc ∧ ¬engine_malfunc,
fc = sensor_malfunc ∨ engine_malfunc
```

Figure 3: UAV BDI agent design.

We take an approach to directing attention that employs the motivation library of [26]. This allows new (and multiple) intentions (*e.g.* to communicate with a human) to be created dynamically when a particular state is recognised (*i.e.* the correct beliefs hold). To be precise, a motivation library (specified by the agent designers) $\mathscr{M}$ is a set of rules of the form: $\psi \rightsquigarrow \langle e, I\rangle$ where ψ is a world state, e an event, and I an identifier. The semantics of motivation execution is specified as follows:

$$\frac{\psi \rightsquigarrow \langle e,I\rangle \in \mathscr{M} \quad \mathscr{B} \models \psi \quad \langle e,I\rangle \notin \Gamma}{\langle E^e, \mathscr{B}, \Gamma\rangle \rightarrow \langle E^e \cup \langle e, I, \texttt{active}\rangle, \mathscr{B}, \Gamma \cup \{\langle e,I\rangle\}\rangle} \; A_{motive}$$

Informally, if the agent believes ψ, it should adopt the event $\langle e,I\rangle$ (if it has not adopted it before). As such, the programmers can specify (and revise) multiple rules $\psi \rightsquigarrow \langle e_1, I_1\rangle, \ldots, \psi \rightsquigarrow \langle e_n, I_n\rangle$ to ensure a correct set of events to be generated at any time when ψ holds. From our experience, this approach also benefits from the modularity principle by separating the dynamics of desires and design of plan library.

3 UAV Example

To demonstrate our new transparency mechanisms, we give a simple Unmanned Aerial Vehicles (UAVs) example, where UAVs are used for object retrieval tasks, *e.g.* package delivery, and might be subject to engine or sensor malfunction. The agent design is in Fig. 3 where we heavily rely on declarative goal and failure recovery features in CAN. There is one main retrieve task, initiated by external event `e_retrv` (line 4), which can be handled by five relevant plans (lines 8–12). The first 3 plans provide different flight paths after take-off in which they all have a declarative goal. For example, the declarative goal

`goal(at_destination, e_path1, fc)` says that it is achieved if it believes `at_destination` holds and failed if `fc` holds. When `fc` holds, the other two plans (lines 11 and 12) perform safe recovery in the event of engine or sensor malfunction. In the case of engine malfunction, instead of returning to base (when sensor is faulty), the plan in line 12 instructs the UAV to land and park itself. Once the UAV is landed and parked, the human teammate should be notified of the situation and respond accordingly. Therefore, for directing attention, the motivation library is given in line 6, so that if the belief that the UAV parked itself (*i.e.* `parked`) holds after the action `activate_parking`, the agent should adopt the event `e_parked` to send GPS coordinates (line 16) to a human teammate to retrieve the UAV. For succinct presentation, we do not show the encoding of actions such as `take_off` and `retrieve` (which can found in the online model[3]).

3.1 Analysis

We encoded the CAN semantics in bigraphs [18, 2], which has been used previously to encode the semantics of process calculi [8, 30]. Our bigraph encoding permits execution and symbolic analysis and we employ BigraphER [29]—an open-source language and toolkit for bigraphs—to generate (and export) a transition system for analysis with model checking tools *e.g.* PRISM [16]. The example above is available in BigraphER format in the online model. For reasoning, states are labelled with *bigraph patterns* [4], predicates that indicate if there is match of a bigraph in that state. Temporal properties are expressed using linear or branching time temporal logics *e.g.* Computation Tree logic (CTL) [10].

For observability, we want to check 1) if an intention is being progressed, its status should never be `pending`, 2) if an intention becomes a completed empty program, its related event will eventually succeed, and 3) if an intention becomes blocked, but is not an empty program, its related original event will eventually fail. For progress, we label each step in the execution trace with its true progress estimation and check that there always exists a path where these match. For attention-directing, we check that if the belief atom `parked` is believed, then eventually the event `e_parked` will be added to the agent desires.

As an example, consider observability and properties 2) and 3) for the event, namely $\langle$`e_retrv, identifier1`$\rangle$ given in Fig. 3, which we can express in CTL as follows:

$$\mathbf{A}[\varphi_1 \implies \mathbf{F}(\varphi_2 \wedge \neg\varphi_4)] \qquad \text{for property 2)}$$
$$\mathbf{A}[\varphi_3 \implies \mathbf{F}(\varphi_4 \wedge \neg\varphi_2)] \qquad \text{for property 3)}$$

The specification for property 2) checks that along all paths, if an intention is completed (*i.e.* φ_1 holds), this implies that eventually the original event will succeed (and not fail), *i.e.* $\mathbf{F}(\varphi_2 \wedge \neg\varphi_4)$. Similar logic applies when an intention is blocked for property 3). We do not give full details, but for the interested reader, the state formulae can be represented by the following bigraph patterns:

$$\varphi_1 \stackrel{\text{def}}{=} \mathsf{Intent_e}.(\mathsf{Identifier.Identifier1} \mid \mathsf{Nil} \mid \mathsf{id})$$
$$\varphi_2 \stackrel{\text{def}}{=} \mathsf{Event_e}.(\mathsf{Identifier.Identifier1} \mid \mathsf{Success} \mid \mathsf{id}),$$
$$\varphi_3 \stackrel{\text{def}}{=} \mathsf{Intent_e}.(\mathsf{Identifier.Identifier1} \mid \mathsf{ReduceF} \mid \mathsf{id})$$
$$\varphi_4 \stackrel{\text{def}}{=} \mathsf{Event_e}.(\mathsf{Identifier.Identifier1} \mid \mathsf{Failure} \mid \mathsf{id}),$$

where Identifier1 is the identifier of the event $\langle$`e_retrv, identifier1`$\rangle$, the symbol id (called a site in bigraphs) stands for the part of model that is abstracted away such as the execution trace, and the subscript e (called a link) maintains the mean-end relation between an original event and the related intention. The transition system for the agent in Fig. 3 has **44** states and **44** transitions[4] and all the above mentioned properties are shown to hold.

[3] `https://bitbucket.org/uog-bigraph/observable_attention-directing_bdi_model/src/master/`

[4] There are numerous internal states, that do not appear in the final transition system, but add to build time.

4 Related Work

We are not the first to label intention status [13, 14] (as pending/active), however like the original CAN semantics, these approaches do not make intention success/failure explicit. We believe having this information is much more important to human users than simply knowing an intention is dropped. Intention estimation has also been explored [35, 36], but in the context of scheduling where they wish to choose the most-completed intention first. The approach is based on *completeness* measures that use the resource consumption and effects of achieving intentions as an estimate, however this requires domain knowledge on the pre-conditions/post-effects of events that can be hard to obtain. For the attention directing feature, we adopted the motivation library approach from [26]. There are similar approaches such as conditional goals in [23, 24] and automatic events [39] (adopting a goal or an event that is conditionalised by beliefs).

Verifying BDI agents through model checking has been well explored, *e.g.* [12] verifies the decision making part (modelled in BDI agents) of a hybrid autonomous system, and [11] uses model checking to reason about agent programs written in Agent Programming Languages. Bremner *et al.* verify that BDI-based decision making (*e.g.* plan selection) in robotic systems adhere to ethical rules (*e.g.* save human) [7]. For transparency, they propose a recorder that produces human readable logs consisting of information such as belief base and the plans executed. Though we agree on the use of logging as a means of transparency, their aim is to provide logs for future forensic analysis whereas ours is to provide run-time logs for effective HAT.

5 Conclusion and Future Work

We presented an executable framework for verifiable BDI agents supporting *observability* (showing task status) and *attention-directing* (flagging relevant information), to facilitate effective human-autonomy teaming (HAT). Observability allows comprehending *what* an agent is doing and *how much* progress has been made, while attention directing ensures communication of critical information to human teammates.

Observability is implemented by enabling the agent to show the status of intentions (*e.g.* pending or active), and to provide a quantitative estimation of progress at each step. Attention-directing is achieved through a motivation library that allows agents to adopt multiple new events when a particular world state is believed to hold. Using a UAV example, we have demonstrated these features in practice.

Full scale observability and attention directing is beyond this initial framework. Future work includes an in-depth survey to determine what are the most important aspects to be made observable (to humans). This might require domain-dependent observability, for example, sometimes *why* an agent decides to do something is more important than progress estimation. Our current intention progress is a metric that measures the distance to the end of (a possible) execution trace. However, in practice actions do not take the same length of time and further domain knowledge annotation for agent programs may be required for more realistic progress estimation. Transparency encompasses more than observability and attention directing and determining: what to log, how to log, and how to use these logs, *e.g.* future forensic analysis or run-time decision making, will be crucial to tackle emerging HAT scenarios. Finally, a long-term goal is analysis of the accuracy of the intention progression predictions in different scenarios.

Acknowledgements This work is supported by the Engineering and Physical Sciences Research Council, under PETRAS SRF grant MAGIC (EP/S035362/1) and S4: Science of Sensor Systems Software (EP/N007565/1).

References

[1] Victoria Alonso & Paloma de la Puente (2018): *System transparency in shared autonomy: A mini review*. *Frontiers in neurorobotics* 12, p. 83, doi:10.3389/fnbot.2018.00083.

[2] Blair Archibald et al. (2021): *Modelling and verifying BDI agents with bigraphs*. *arXiv preprint arXiv:2105.02578*.

[3] Blair Archibald et al. (2021): *Probablistic bigraphs*. *arXiv preprint arXiv:2105.02559*.

[4] Steve Benford et al. (2016): *On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces*. *ACM Transactions on Computer-Human Interaction (TOCHI)* 23(2), pp. 1–56, doi:10.1145/2882784.

[5] Adella Bhaskara et al. (2020): *Agent transparency: A review of current theory and evidence*. *IEEE Transactions on Human-Machine Systems* 50(3), pp. 215–224, doi:10.1109/THMS.2020.2965529.

[6] Michael Bratman (1987): *Intention, plans, and practical reason*. Harvard University Press, doi:10.2307/2185304.

[7] Paul Bremner et al. (2019): *On proactive, transparent, and verifiable ethical reasoning for robots*. *Proceedings of the IEEE* 107(3), pp. 541–561, doi:10.1109/JPROC.2019.2898267.

[8] Mikkel Bundgaard & Vladimiro Sassone (2006): *Typed polyadic pi-calculus in bigraphs*. In: *Proceedings of ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pp. 1–12, doi:10.1145/1140335.1140336.

[9] Filippo Cantucci & Rino Falcone (2020): *Towards trustworthiness and transparency in social human-robot interaction*. In: *Proceedings of 2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, IEEE, pp. 1–6, doi:10.1109/ICHMS49158.2020.9209397.

[10] Edmund M Clarke & E Allen Emerson (1981): *Design and synthesis of synchronization skeletons using branching time temporal logic*. In: *Proceedings of Workshop on Logic of Programs*, pp. 52–71, doi:10.1007/BFb0025774.

[11] Louise A Dennis et al. (2012): *Model checking agent programming languages*. *Automated software engineering* 19(1), pp. 5–63, doi:10.1007/s10515-011-0088-x.

[12] Louise A Dennis et al. (2016): *Practical verification of decision-making in agent-based autonomous systems*. *Automated Software Engineering* 23(3), pp. 305–359, doi:10.1007/s10515-014-0168-9.

[13] James Harland et al. (2014): *An operational semantics for the goal life-cycle in BDI agents*. *Autonomous agents and multi-agent systems* 28(4), pp. 682–719, doi:10.1007/s10458-013-9238-9.

[14] James Harland et al. (2017): *Aborting, suspending, and resuming goals and plans in BDI agents*. *Autonomous Agents and Multi-Agent Systems* 31(2), pp. 288–331, doi:10.1007/s10458-015-9322-4.

[15] IEEE (2020): *IEEE Draft Standard for Transparency of Autonomous Systems*. In: *IEEE P7001/D1*, Piscataway NJ: IEEE, pp. 1–76.

[16] Marta Kwiatkowska et al. (2011): *PRISM 4.0: Verification of Probabilistic Real-time Systems*. In: *Proceedings of Conference on Computer Aided Verification*, pp. 585–591, doi:10.1007/978-3-642-22110-1_47.

[17] Patricia McDermott et al. (2018): *Human-machine teaming systems engineering guide*. Technical Report, MITRE CORP.

[18] Robin Milner (2009): *The space and motion of communicating agents*. Cambridge University Press, doi:10.1017/CBO9780511626661.

[19] Thomas O'Neill et al. (2020): *Human–autonomy teaming: A review and analysis of the empirical literature*. *Human Factors*, doi:10.1177/0018720820960865.

[20] Raja Parasuraman & Victor Riley (1997): *Humans and automation: Use, misuse, disuse, abuse*. *Human factors* 39(2), pp. 230–253, doi:10.1518/001872097778543886.

[21] Anand S Rao (1996): *AgentSpeak (L): BDI agents speak out in a logical computable language*. In: *European workshop on modelling autonomous agents in a multi-agent world*, Springer, pp. 42–55, doi:10.1007/BFb0031845.

[22] Anand S Rao et al. (1995): *BDI agents: From theory to practice.* In: *Proceedings of the First International Conference on Multiagent Systems*, pp. 312–319.

[23] M. Birna van Riemsdijk et al. (2004): *Dynamics of declarative goals in agent programming*. In: *Proceedings of International Workshop on Declarative Agent Languages and Technologies*, Springer, pp. 1–18, doi:10.1007/11493402_1.

[24] M. Birna van Riemsdijk et al. (2005): *Semantics of declarative goals in agent programming*. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 133–140, doi:10.1145/1082473.1082494.

[25] Sebastian Sardina & Lin Padgham (2007): *Goals in the context of BDI plan failure and planning*. In: *the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 16–23, doi:10.1145/1329125.1329134.

[26] Sebastian Sardina & Lin Padgham (2011): *A BDI agent programming language with failure handling, declarative goals, and planning*. *Autonomous Agents and Multi-Agent Systems* 23(1), pp. 18–70, doi:10.1007/s10458-010-9130-9.

[27] Allison Sauppé & Bilge Mutlu (2015): *The social impact of a robot co-worker in industrial settings*. In: *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 3613–3622, doi:10.1145/2702123.2702181.

[28] Isabella Seeber et al. (2020): *Machines as teammates: A research agenda on AI in team collaboration*. *Information & management*, doi:10.1016/j.im.2019.103174.

[29] Michele Sevegnani & Muffy Calder (2016): *BigraphER: Rewriting and Analysis Engine for Bigraphs*. In: *28th International Conference on Computer Aided Verification*, pp. 494–501, doi:10.1007/978-3-319-41540-6_27.

[30] Michele Sevegnani & Eloi Pereira (2014): *Towards a bigraphical encoding of actors*. In: *International Workshop on Meta Models for Process Languages*, doi:10.13140/RG.2.1.3681.9046.

[31] Michele Sevegnani et al. (2018): *Modelling and Verification of Large-Scale Sensor Network Infrastructures*. In: *23rd International Conference on Engineering of Complex Computer Systems, ICECCS*, pp. 71–81, doi:10.1109/ICECCS2018.2018.00016.

[32] R Jay Shively et al. (2017): *Why human-autonomy teaming?* In: *International conference on applied human factors and ergonomics*, Springer, pp. 3–11, doi:10.1007/978-3-319-60642-2_1.

[33] Anna Spagnolli et al. (2017): *Transparency as an ethical safeguard*. In: *International Workshop on Symbiotic Interaction*, Springer, pp. 1–6, doi:10.1007/978-3-319-91593-7_1.

[34] J. Thangarajah & L. Padgham (2011): *Computationally effective reasoning about goal interactions*. *Journal of Automated Reasoning* 47(1), pp. 17–56, doi:10.1007/s10817-010-9175-0.

[35] John Thangarajah et al. (2014): *Quantifying the completeness of goals in BDI agent systems*. In: *ECAI 2014*, IOS Press, pp. 879–884, doi:10.3233/978-1-61499-419-0-879.

[36] John Thangarajah et al. (2015): *Estimating the Progress of Maintenance Goals.* In: *AAMAS*, pp. 1645–1646.

[37] Christos Tsigkanos et al. (2020): *Scalable Multiple-View Analysis of Reactive Systems via Bidirectional Model Transformations*. In: *35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 993–1003, doi:10.1145/3324884.3416579.

[38] Alan FT Winfield et al. (2021): *IEEE P7001: A Proposed Standard on Transparency*. *Frontiers in Robotics and AI*, p. 225, doi:10.3389/frobt.2021.665729.

[39] Michael Winikoff (2005): *JACK intelligent agents: an industrial strength platform*. In: *Multi-Agent Programming*, Springer, pp. 175–193, doi:10.1007/0-387-26350-0_7.

[40] Michael Winikoff et al. (2002): *Declarative and procedural goals in intelligent agent systems*. In: *8th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 470–481.

[41] Mengwei Xu et al. (2019): *Intention interleaving via classical replanning*. In: *31st International Conference on Tools with Artificial Intelligence*, IEEE, pp. 85–92, doi:10.1109/ICTAI.2019.00021.