Trust Modelling and Verification Using Event-B

Asieh Salehi Fathabadi
University of Southampton, United Kingdom
a-salehi-fathabadi@soton.ac.uk

Vahid Yazdanpanah
University of Southampton, United Kingdom
v.yazdanpanah@soton.ac.uk

Trust is a crucial component in collaborative multiagent systems (MAS) involving humans and autonomous AI agents. Rather than assuming trust based on past system behaviours, it is important to formally verify trust by modelling the current state and capabilities of agents. We argue for verifying actual trust relations based on agents' abilities to deliver intended outcomes in specific contexts. To enable reasoning about different notions of trust, we propose using the refinement-based formal method Event-B. Refinement allows progressively introducing new aspects of trust - from abstract to concrete models incorporating knowledge and runtime states. We demonstrate modelling three trust concepts and verifying associated trust properties in MAS. The formal, correctness-by-construction approach allows to deduce guarantees about trustworthy autonomy in human-AI partnerships. Overall, our contribution facilitates rigorous verification of trust in multiagent systems.

1 Introduction

Trust is a crucial contextual concept in multiagent systems (MAS), representing the cognitive state of a trustor towards a trustee [3]. While there are various accounts of trust, in this work, we focus on trust with respect to accomplishing tasks. While trust modelling in MAS has historically relied on reasoning about past behaviours[10], recent work emphasises integrating current context rather than fully depending on history. This involves verifying what agents can actually deliver based on their present capabilities, beyond reputations. We argue for complementing offline safety assurances with online trust verification for autonomous systems. Consider an autonomous delivery vehicle (ADV) tasked with transporting goods. Offline verification during design suffices for basic safety and whether the ADV is reliable in general (regardless of their current state and how they can perform in the context). However, assessing trust online for a particular delivery also requires checking the ADV's abilities given its current battery, payload etc. against user requirements.

Trust modelling in MAS, and what we introduce as "actual trust", entails representing different aspects like agents' abilities, knowledge and commitments. To model such a multidimensional notion, refinement techniques like Event-B [1] allow correct-by-construction modelling [7, 5]. Our key contribution is a refinement-based approach that supports formally verifying various trust concepts. We demonstrate formally modelling three trust notions relating to agent abilities, knowledge and commitments. The automated consistency guarantees complement offline assurance for trustworthy autonomy and human-AI partnerships [11, 12]. This work is an initial step on modelling trust using Event-B's refinement strategy that practically enables step-wise verification of actual trust between agents in autonomous systems.

2 Actual Trust: Power, Knowledge, and Commitments

In modelling and reasoning about trust, it is key to distinguish what an agent may rely on due to past behaviour of another agent and their *typical* behaviour from what in a given situation agents are *actually*

Submitted to:
FMAS 2023

able to deliver. While the former category of trusting has a retrospective view, and uses history to reason about trust [10], the latter form of trust is to reason about what the other agent can *actually* deliver and has basis in what is known in the theory of causality as *actual causality* [6]. In this work, we focus on the latter notion, refer to it as *actual trust* and understand it as a relational notion between two agents or agent groups i (as the trustor) and j (as the trustee) and say in a particular multiagent system M, i trusts j with respect to task t only if i is able to verify that j is able and committed to deliver t. To model and verify our notion of actual trust, as knowledge of another agents' ability and commitment to ensure a particular task, it is important to highlight how it relates to its key components conceptually.

Trusting for ability to materialise eventualities: In contrast to purely history-oriented perspectives to trust, that look at the history and trust an agent to behave similar to its past behaviour, we deem that trusting needs to be fine-tuned based on the current state of the system and actions agents are able to execute and what agents intend to deliver. For instance, even if an ADV was successful in former deliveries, it may be suffering from a low battery now and unable to deliver tasks. So, one should fine tune trust in the agent's power to deliver based on the current situation.

Trust as an epistemic state: We understand actual trust as an epistemic notion meaning that it is essentially about knowledge of the trustor on how another agent relates to a particular event. Recalling the running example, the user needs to reason about abilities of an ADV, consider its publicly announced intentions, and verify if the ADV can be trusted for a particular delivery. This form of trust allows specification of trust in different contexts and for different types requirements and knowledge levels. For instance, a given ADV j may be seen as "trusted for delivering $5kg$ of groceries" but this trust may not extend when it comes to passenger pickup..

Public commitments as a proxy to intentions: When we are dealing with autonomous AI agents, we need to consider that being able to deliver a task fundamentally differs from delivering the task. Imagine that an ADV v with a full battery and ability to deliver some goods is located relatively close to an agent i with a delivery tasks t. In this case, i can't simply assume that v can be trusted for delivering t as it may be already committed to deliver tasks other than t or is in the middle of other plan executions. To handle this, we use notion of publicly-announced commitments as a proxy to model what agents intend to bring about.[1]

3 Refinement-Based Trust Formal Modelling and Verification [2]

Background knowledge: Event-B [1] is a refinement-based formal method for system development. The mathematical language of Event-B is based on set theory and first order logic[a]. An Event-B model consists of two parts: *contexts* for static data and *machines* for dynamic behaviour. An Event-B model is constructed by making progressive refinements starting from an initial abstract model which may have more general behaviours and gradually introducing more detail that constrains the behaviour towards the desired system. Each refinement step is verified to be a valid refinement of the previous step.

[a] Please refer to Event-B Language user manual `https://wiki.event-b.org/index.php/Event-B_Language` for extra support to understand the presented model.

[1] Note that assuming full access to agents' intentions is against separation of concerns, privacy, and encapsulation as key design principles in safe and responsible AI and software development.

[2] Note that because of space limitation, the Event-B model of trust is not fully presented here. And for simplicity, in purpose of demonstrating the vision idea, we model trust in its simplest definition.

Benefiting from refinement technique in building Event-B formal model, instead of one single-layer complex design model of system, we propose to gradually introduce different concepts of actual trust through refinement steps. Figure 1 presents our vision idea of applying refinement-based development to model actual trust in autonomous systems. Left side illustrates the trust relationship between trustor and trustee, while right side presents the structure of our proposed Event-B formal model, including three levels of refinements: machines ($M0$, $M1$ and $M2$) and associated contexts ($cntx0$, $cntx1$ and $cntx2$).

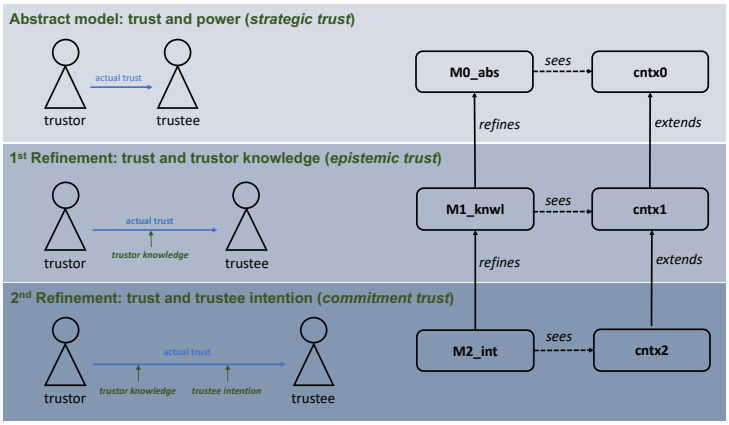

Figure 1: Trust Modelling and the Refinement Strategy

In line with trust key components and first-order building blocks presented in Section 2, starting from the top level, trust is first modelled as an abstract relationship between two agents, trustor and trustee, *M0_abs*; followed by first refinement level where *trustor knowledge* is introduced, *M1_knwl*. Then *trustee intention* is introduced in a further refinement level, *M2_int*.

3.1 Modelling trust in Event-B

Agents and tasks are defined as a set in the context *cntx0*, which is partitioned to two sub-sets: trustors and trustees:

Background knowledge: An Event-B contexts contains carrier sets s, constants c, and axioms A(c) that constrain the carrier sets and constants.

```
CONTEXT cntx0
SETS AGENTS TASKS CONSTANTS trustors trustees
AXIOMS axm1 : trustors ⊆ AGENTS // Definition: Subset ⊆
       axm2 : trustees ⊆ AGENTS
       axm3 : partition (AGENTS, trustors, trustees)
```

Definition 1. Abstract (strategic) trust: agent i weakly (abstraction) trusts j regarding task t if j has an action or a sequence of actions available to it to ensure e. We are operating in a cooperative setting, hence assuming that agents share information and have perfect knowledge of themselves as well as other agents' abilities. We define trust in terms of the ability to deliver t.

Trust is modelled as a three-dimension relation variable between a trustor, a set of trustees and a task, in the abstract machine. In *M0_abs*, an invariant, $inv1$, is specifying the *agent_task* variable as a function between a subset of trustees and a single task, indicating the task that can be delivered by a subset of

trustee agents. And *inv*2 is specifying the *trustor_trustee_task* variable indicating the relation between a trustor and a pair of *agent_task*.

Background knowledge: An Event-B machine contains variables v, invariant predicates I(v) that constrain the variables, and events. In Event-B, a machine corresponds to a transition system where *variables* represent the states and *events* specify the transitions.

@inv1 : agent_task ∈ ℙ(trustees) ⇸ tasks // specifies which agents are able to deliver which task
// Definition: Powerset: ℙ(S) = {s | s ⊆ S}
// Definition: A function (agent_task) is a relation with the restriction that each element of the domain (ℙ(trustees)) is related to a unique element in the range (tasks); a many to one mapping
// Definition: Set membership ∈
@inv2 : trustor_trustee_task ∈ trustors ⇸ agent_task // specifies set of triples i ↦ (j ↦ t), when agent i can trust a set of agents j to deliver task t

The event *trust* is adding a new triple to the *trustor_trustee_task* variable, *act*1. While *grd*1 − 3 is checking the type of the event parameters, *grd*4 is indicating the above definition, ensuring j is able to deliver task t; guards *grd*5 − 8 are described later.

Background knowledge: An event in a machine, comprises a guard denoting its enabling-condition and an action describing how the variables are modified when the event is executed. In general, an event e has the following form, where t are the event parameters, G(t, v) is the guard of the event, and v := E(t, v) is the action of the event: e == **any** t **where** G(t,v) **then** v := E(t,v) **end**

event trust **any** i j t
where @grd1 : i ∈ trustors
@grd2 : j ∈ ℙ(trustees)
@grd3 : t ∈ tasks
@grd4 : t ∈ agent_task[{j}] // j is able to deliver task t
// Definition: relational image: r[S] = {y | ∃x·x ∈ S ∧ x ↦ y ∈ r} where S is a set
@grd5 : i ∉ j // to preserve inv3
// Definition: Set non−membership ∉
@grd6 : j ≠ ∅ // abstract guard to preserve inv4
@grd7 : j ⊆ knowledge[{i}] // refining guard to preserve inv4
@grd8 : commitments[i ↦ (j ↦ t)] = {TRUE} // refining guard to preserve inv4
then @act1 : trustor_trustee_task := trustor_trustee_task ∪ {i ↦ (j ↦ t)}
// Definition: Union ∪

Running example: For instance, for an agent i and an ADV j and task of "delivering 5kg of groceries", i can trust j only if "delivering 5kg of groceries" is within the allocated tasks to j: *grd*3. Then, *act*1 will add a new triple of (i, j, t) to the variable set *trustor_trustee_task*.

3.2 Modelling verifiable trust properties

To propose the idea of formal verification of properties of trust in autonomous systems, here we presents two invariants, specifying two fundamental trust properties. *inv3* is specifying that an agent i would not trust itself to deliver a specific task t. And *inv4* is specifying avoiding trust deadlock, that for each trustor i and task t, there is always a non-empty subset of trustees j that can deliver t.

@inv3: $\forall i, j \cdot i \in trustors \land j \in \mathbb{P}(trustees) \land i \in dom(trustor\_trustee\_task) \Rightarrow i \notin j$
// Definition: Conjunction $\land$, Universal quantification $\forall$, Implication $\Rightarrow$
// Definition: Domain: dom(r) $\forall r \cdot r \in S \leftrightarrow T \Rightarrow dom(r) = \{x \cdot (\exists y \cdot x \mapsto y \in r)\}$ where S and T are sets
@inv4: $\forall i, t \cdot i \in trustors \land t \in tasks \Rightarrow (\exists j \cdot j \in \mathbb{P}(trustees) \land j \neq \varnothing)$
// Definition: Existential quantification $\exists$

3.3 Verifying trust properties

Background knowledge: Event-B is supported by the Rodin tool set [2], an extensible open source toolkit which includes facilities for modelling, verifying the consistency of models using theorem proving and model checking techniques, and validating models with simulation-based approaches.

One of the generated proof obligations (PO) for an Event-B model, is "invariant preservation": e / v / INV (where *e* is the event name, and *v* is the invariant name)

INV PO ensures that the property specified in the invariant INV is preserved by event e. To preserve the trust properties defined in *inv*3 and *inv*4, the event *trust* is guarded by *grd*5 and *grd*6, see above. Two POs trust/inv3/INV and trust/inv4/INV are generated and automatically discharged by Rodin tool.

3.4 Refining trust

Next, we introduced the refined notion of epistemic trust in which agents' knowledge is integrated.

Definition 2. Refined (epistemic) trust: for a stronger notion of trust we require a variable of *knowledge* specifying the knowledge relationship between two agents i, j, indicating whether i is fully aware of j's abilities.

Refining model *M1_knwl* introduces the *knowledge* variable to model the knowledge of trustors about trustees:

@inv1: $knowledge \in trustors \leftrightarrow trustees$
// Definition: A relation (knowledge) is a set of ordered pairs; a many to many mapping.

Running example: For instance, a given ADV j may be seen as "trusted for delivering $5kg$ of groceries" by an agent i who is fully aware of j's abilities but not by agent v who is not aware of j and that j has the capacity to ensure t.

Definition 3. Refined (commitment) trust: for a stronger notion of trust we require a variable of *commitments* specifying a function that takes a trust triple (i, j, t) and determines whether agent j is committed to deliver task t for agent i. We refine the trust model, not only in terms of the ability, but also the commitment to deliver t.

And refining model *M2_int* introduces the *commitment* variable to model the intention of trustees to deliver the associate task (for simplicity in this paper, we model commitment as a Boolean indicating whether an agent(s) as trustee intends to deliver the associated task or not):

@inv1: $commitments \in trustor\_trustee\_task \rightarrow BOOL$

*inv*4 is refined to include the knowledge property in *M1_knwl* and commitment specification in *M2_int*:

@inv4: $\forall i, t \cdot i \in trustors \land t \in tasks \Rightarrow$
$(\exists j \cdot j \in \mathbb{P}(trustees) \land j \neq \varnothing \land (j \mapsto t) \in agent\_task \land$
$j \subseteq knowledge[\{i\}] \land commitments[i \mapsto (j \mapsto t)] = \{TRUE\} \land$
$i \mapsto (j \mapsto t) \in trustor\_trustee\_task)$

And refining event *trust* includes extra guards *grd*7 and *grd*8 to preserve *inv*4, see above. Not providing these guards results in failed generated INV POs.

Running example: For instance, for an agent *i* and an ADV *j* and task of "delivering 5kg of groceries", *i* can trust *j* only if "delivering 5kg of groceries" is within the allocated tasks to *j*: *grd*3 (verified in the abstract machine), and *i* is fully aware of j's abilities: *grd*7 (verified in the machine *M1_knwl*) and *j* is committed to deliver 5kg of groceries to *i*: *grd*8 (verified in the machine *M2_int*).

Model checking trust properties: The presented Event-B model can be model checked by instantiating the context elements, for example for the ADV system. Also the scenario checker integrated in Rodin can demonstrate difference scenarios of the desire system. Due to the concise nature of this work and space limitation, we are unable to include the modelling checking experience.

4 Concluding Remarks and Future Directions

The step-wise refinement approach presented in this paper, demonstrates three notions of actual trust, and two verifiable trust properties. The model can simply refined to include more notions and properties. This paper elaborates on how the autonomous system research can benefit from refinement-based formal methods in terms of modelling trust. The abstraction technique aids the modelling and verification process in step-wise manner, where instead of a single complex model, the formal model is gradually built through refinement levels, hence easier to be understood and proved. Also the Event-B formal method provides the verification techniques (theorem proving and ProB model checking [8]) in each refinement level, to ensure the trustworthiness of autonomous systems.

Contributions to Autonomous Systems (AS): In AS, replacing human decision-making with machine decision-making results in challenges associated with stakeholders' trust. Trustworthiness of an AS is key to its wide-spread adoption by society. To develop a trusted AS, it is important to understand how different stakeholders perceive an AS as trusted, and how the context of application affects their perceptions. The translation of trust issues into formalised solutions is challenging due to trust dynamics. In this work, we try to advance in this direction by utilising the ability of Event-B as a refinement-based formal method to manage the lack of information when modelling trust in multi-agent systems. High-level model aids to abstract away the uncertain/unknown trust specifications. We introduced the notion of actual trust versus statistical trust, toward trusting to an AS due to its safety checks, like, inherent uncertainties in the environment, diversity in the requirements and needs of different users and contexts of application. We formalised the notion of actual trust using Event-B formal modelling followed by verifying the safety properties of it. The actual trust notions is modelled and verified in three levels: strategic trust, epistemic trust and commitment trust.

Future directions: This formal modelling and verification approach for trust in autonomous systems can be extended in several directions. One avenue is via integrating Event-B models with Alternating-Time Temporal Logic [14] to allow more expressive temporal specifications and model checking of trust properties, e.g., in the context of connected mobility systems. Further research can also investigate gradation of trust (as a quantifiable notion) and formally relating trust and neighbouring notions in multiagent settings such as responsibility [13]. Quantifying trust based on strategy lengths and information-theoretic notions may also complement the approach pursued here. Overall, rigorous formal methods can provide significant assurances about trustworthy autonomy and human-AI partnerships, especially for safety-critical applications.

Acknowledgements: This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through a Turing AI Fellowship (EP/V022067/1) on Citizen-Centric AI Systems (`https://ccais.ac.uk/`) and the UKRI Trustworthy Autonomous Systems Hub (EP/V00784X/1).

References

[1] J-R. Abrial (2010): *Modeling in Event-B: System and Software Engineering*. Cambridge University Press.

[2] J-R Abrial, M. Butler, S. Hallerstede, T.S. Hoang, F. Mehta & L. Voisin (2010): *Rodin: An Open Toolset for Modelling and Reasoning in Event-B*. Software Tools for Technology Transfer 12(6), pp. 447–466.

[3] Cristiano Castelfranchi & Rino Falcone (2020): *Trust: Perspectives in cognitive science*. The Routledge Handbook of Trust and Philosophy, pp. 214–228.

[4] Mehdi Dastani & Vahid Yazdanpanah (2023): *Responsibility of AI systems*. Ai & Society 38(2), pp. 843–852.

[5] Hang-Jiang Gao, Zheng Qin, Lei Lu, Li-Ping Shao & Xing-Chen Heng (2007): *Formal specification and proof of multi-agent applications using event b*. Information Technology Journal 6(7), pp. 1181–1189.

[6] Joseph Y Halpern (2016): *Actual causality*. MiT Press.

[7] Arnaud Lanoix (2008): *Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles*. In: Second IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering, TASE 2008, June 17-19, 2008, Nanjing, China, IEEE Computer Society, pp. 297–304.

[8] Michael Leuschel & Michael Butler (2008): *ProB: An Automated Analysis Toolset for the B Method*. Software Tools for Technology Transfer (STTT) 10(2), pp. 185–203.

[9] Michael Leuschel & Michael Butler (2008): *ProB: An Automated Analysis Toolset for the B Method*. Software Tools for Technology Transfer (STTT) 10(2), pp. 185–203.

[10] Sarvapali D Ramchurn, Dong Huynh & Nicholas R Jennings (2004): *Trust in multi-agent systems*. The knowledge engineering review 19(1), pp. 1–25.

[11] Sarvapali D Ramchurn, Sebastian Stein & Nicholas R Jennings (2021): *Trustworthy human-AI partnerships*. Iscience 24(8), p. 102891.

[12] Sebastian Stein & Vahid Yazdanpanah (2023): *Citizen-Centric Multiagent Systems*. In: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, pp. 1802–1807.

[13] Vahid Yazdanpanah & Mehdi Dastani (2016): *Quantified degrees of group responsibility*. In: Coordination, Organizations, Institutions, and Norms in Agent Systems, Springer, pp. 418–436.

[14] Chenyang Zhu, Michael Butler, Corina Cirstea & Thai Son Hoang (2023): *A fairness-based refinement strategy to transform liveness properties in Event-B models*. Science of Computer Programming 225, p. 102907.