Towards Verifying GOAL Agents in Isabelle/HOL

Alexander Birch Jensen[a]

DTU Compute, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, Building 324, DK-2800 Kongens Lyngby, Denmark

Keywords: Agent Programming, Formal Verification, Proof Assistants.

Abstract: The need to ensure reliability of agent systems increases with the applications of multi-agent technology. As we continue to develop tools that make verification more accessible to industrial applications, it becomes an even more critical requirement for the tools themselves to be reliable. We suggest that this reliability ought not be based on empirical evidence such as testing procedures. Instead we propose using an interactive theorem prover to ensure the reliability of the verification process. Our work aims to verify agent systems by emdedding a verification framework in the interactive theorem prover Isabelle/HOL.

1 INTRODUCTION

A key issue in deployment of multi-agent systems is to ensure its reliability (Dix et al., 2019). We observe that testing does not translate well from traditional software to multi-agent technology: we often have agents with complex behavior and the number of possible system configurations may be intractable.

Two leading paradigms in ensuring reliability are formal verification and testing. The state-of-the-art approach for formal verification is model-checking (Bordini et al., 2006) where one seeks to verify desired properties over a (generalized) finite state space. However, the process of formal verification is often hard and reserved for specialists. As such, we find that testing has a lower entry-level for ensuring reliable behavior in critical configurations. Metrics such as code coverage help solidify that the system is thoroughly tested. Even so, reliability of testing stands or falls by the knowledge and creativity of the designer.

In our work, we focus on formal verification. We lay out the building blocks for a solid foundation of a verification framework (de Boer et al., 2007) for the GOAL agent programming language (Hindriks, 2009) by formalizing it in Isabelle/HOL (Nipkow et al., 2002) (a proof assistant based on higher-order logic). All proofs developed in Isabelle are verified by a small logical kernel that is trusted. This ensures that Isabelle itself as well as any proof developments are trustworthy.

The paper is structured as follows. Section 2 considers related work in the literature. Section 3 introduces the semantics of GOAL and a verification framework for GOAL agents. Section 4 goes into details with our work on formalizing GOAL in Isabelle/HOL. Section 5 discusses some of the future challenges. Finally, Section 6 makes concluding remarks.

2 RELATED WORK

The work in this paper relates to our work on mechanizing a transformation of GOAL program code to an agent logic (Jensen, 2021). Unlike this paper, it does not consider a theorem proving approach. It focuses on closing the gap between program code and logic which is an essential step towards improving the practicality of the approach. In (Jensen et al., 2021), we argue that a theorem proving approach is interesting to pursue further for cognitive agent-oriented programming, and consequently we ask ourselves why it has played only a miniature role in the literature so far.

(Alechina et al., 2010) apply theorem-proving techniques to successfully verify correctness properties of agents for simpleAPL (a simplified version of 3APL (Hindriks et al., 1999)).

(Dennis and Fisher, 2009) develop techniques for analysis of implemented, BDI-based multi-agent system platforms. Model-checking techniques for AgentSpeak (Bordini et al., 2006) are used as a starting point and then extended to other languages includ-

[a] https://orcid.org/0000-0002-7708-667X

Jensen, A.
Towards Verifying GOAL Agents in Isabelle/HOL.
DOI: 10.5220/0010268503450352
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 1*, pages 345-352
ISBN: 978-989-758-484-8

ing the GOAL programming language.

Tools for generating test cases and debugging multi-agent systems have been proposed by (Poutakidis et al., 2009), where the test cases are used to monitor (and debug) the system behavior while running. Another take on the testing approach is by (Cossentino et al., 2008) which simulates a simplified version of the system by executing it in a virtual environment. This especially appeals to systems to be deployed in real-world agent environments where testing can be less accessible or of a high cost.

The verification framework theory of programs has some notational similarities with UNITY (Misra, 1994). However, while we verify an agent programming language, UNITY is for verification of general parallel programs. (Paulson, 2000) has worked at mechanizing the UNITY framework in Isabelle/HOL.

3 VERFICATION FRAMEWORK

In this section, we introduce GOAL, its formal semantics and a verification framework for agents.

3.1 GOAL Agents

Agents in GOAL are rational agents: they make decisions based on their current beliefs and goals. The agent's beliefs and goals continuously change to adapt to new situations. Thus, it is useful to describe the agent's by its current *mental state*: its current beliefs and goals.

Based on the agent's mental state, it may select to execute a *conditional action*: an action that may be executed given the truth of a condition on the current mental state (its enabledness).

GOAL agents are thus formally defined by an initial mental state and a set of conditional actions. We will get into more details with those concepts in the following.

3.1.1 Mental States

A mental state is the agent's beliefs and goals in a given state and consists of a belief and a goal base. The belief and goal base are sets of propositional formulas for which certain restrictions apply:

1. The set of formulas in the belief base is consistent,
2. for every formula ϕ in the goal base:
   (a) ϕ is not entailed by the agent's beliefs,
   (b) ϕ is satisfiable,
   (c) If ϕ entails ϕ', and ϕ' is not entailed by the agent's beliefs, then ϕ' is also a goal.

A consistent belief base means that it does not entail a contradiction which in turn ensures that we cannot derive everything (any formula follows from falsity). The goals are declarative and describe a state the agent desires to reach. As such, it does not make sense to have a goal that is already believed to be achieved. Furthermore, we should not allow goals that can never be achieved. The following captures some of the temporal features of goals: consider an agent that has as its goal to "*be at home* $\wedge$ *watch a movie*". Then both "*be at home*" and "*watch a movie*" are subgoals. On the contrary, from individual goals to "*be at home*" and "*watch a movie*", we cannot conclude that it is a goal to "*be at home* $\wedge$ *watch a movie*".

Any sort of knowledge we want the agent to have can be packed into the belief base as a logic formula. As we are modelling agents for propositional logic, the expressive power is of course limited to propositional symbols, but this is more of a practical limitation than a theoretical one.

3.1.2 Mental State Formulas

To reason about mental states of agents, we introduce a language for mental state formulas. These formulas are built from the usual logical connectives and the special belief modality $\mathsf{B}\Phi$ and goal modality $\mathsf{G}\Phi$, where Φ is a formula of propositional logic.

The semantics of mental state formulas is defined as usual for the logical connectives. For the belief and goal modality, we have:

- $(\Sigma,\Gamma) \models_M \mathsf{B}\Phi \iff \Sigma \models_C \Phi$,
- $(\Sigma,\Gamma) \models_M \mathsf{G}\Phi \iff \Phi \in \Gamma$,

where (Σ,Γ) is a mental state with belief base Σ and goal base Γ and $\models_C$ is classical semantic consequence for propositional logic.

We can think of the modalities as queries to the belief and goal base:

- $\mathsf{B}\Phi$: do my current beliefs entail Φ?
- $\mathsf{G}\Phi$: do I have a goal to achieve Φ?

Below we state a number of important properties of the goal modality:

- $\not\models_M \mathsf{G}(\phi \longrightarrow \psi) \longrightarrow (\mathsf{G}\phi \longrightarrow \mathsf{G}\psi)$,
- $\not\models_M \mathsf{G}(\phi \wedge (\phi \longrightarrow \psi)) \longrightarrow \mathsf{G}\psi$,
- $\not\models_M \mathsf{G}(\phi \wedge \psi) \longrightarrow (\mathsf{G}\phi \wedge \mathsf{G}\psi)$,
- $\models_C (\varphi \longleftrightarrow \gamma) \implies \not\models_M (\mathsf{G}\varphi \longleftrightarrow \mathsf{G}\gamma)$,

where $\not\models_M$ stated without a mental state is to be understood as universally quantified over all mental states. The properties show that G does not distribute over implication, subgoals cannot be combined to form a

larger goal, and lastly that equivalent formulas are also equivalent goals.

We further define syntactic consequence $\vdash_M$ for mental state formulas as given by the rules R1-R2 and axioms A1-A5 in Tables 1 and 2. These rules are as given in (de Boer et al., 2007). We assume the existence of $\vdash_C$: a proof system for propositional logic.

Mental states are useful for stating the agent's mental state at a given point in time. The idea is that the mental state changes over time, either due to perceived changes in the environment or due to actions taken by the agent. We will assume that any changes in the environment are solely caused by the actions of agents. This allows us to model changes to mental states as effects of performing actions.

3.1.3 Actions

Agents take actions to change the state of affairs, and thus their mental states. Hopefully, a sequence of actions will lead to the agents having a mental state where it believes its goals are achieved (believed). In general terms, the idea behind our work is that the existence of such a sequence is provable.

In order for the agent to make reasonable choices, actions have conditions that state when they can (and should) be performed. The pair of action and condition form a conditional action. The conditional action $\varphi \triangleright do(a)$ of an agent states a condition φ for when the agent may execute action a. We may write $enabled(a)$ when referring to the condition of action a. The conditions are mental state formulas.

We now need to specify what it means to execute an action. For this, we need to associate the execution of an action with updates to the belief base. In practical cases, this is given by an action specification stated in a more general language (i.e. using variables). Informally, the result of executing an action is thus to update the belief base of the mental state and to remove any goals that the agent now believes to be achieved.

Note that we do not allow for direct manipulation of the goal base. We have an initial goal base that is only changed as a result of action execution. This is not an inherent limitation of GOAL, but is a useful abstraction for the purpose of this paper.

A trace is an infinite sequence of interleaved mental states and action executions. As such, a trace describes an order of execution. In case the agents have choices, multiple traces may exist for a given program. We assume that we have only fair traces: each action is scheduled infinitely often. This means that there is always a future point in which an action is scheduled (note that a scheduled action need not be enabled).

3.2 Proof Theory

Hoare triples are used to specify actions: $\{\varphi\}\ \rho \triangleright do(a)\ \{\psi\}$ specifies the conditional action a with mental state formulas φ and ψ that state the pre- and postcondition, respectively. These will be the axioms of our proof system and will vary depending on the context, i.e. available actions.

The provable Hoare triples are given by the Hoare system in Table 3. The rule for infeasible actions states the non-effect of actions when not enabled. The rule for conditional actions gives means to reason about the enabledness of a conditional action. The conjunction and disjunction rules allow for combining proofs of Hoare triples. Lastly, with the consequence rule we can strengthen the precondition and weaken the postcondition.

3.3 Proving Properties of Agents

On top of Hoare triples, a temporal logic is used to verify properties of GOAL agents. We extend the language of mental state formulas with the temporal operator **until**. The formula φ **until** ψ means that ψ eventually comes true and until then φ is true, or that φ never becomes true and ψ remains true forever.

The temporal logic formulas are evaluated with respect to a given trace and some time point on the trace. This means that conditions are checked in the current mental state as the program is executed.

Proving that a property of the agent is ensured by the execution of the program is done by proving a number of **ensures** formulas:

$$\varphi \,\mathbf{ensures}\, \psi := (\varphi \longrightarrow (\varphi \,\mathbf{until}\, \psi)) \wedge (\varphi \longrightarrow \Diamond \psi)$$

The formula φ **ensures** ψ states that φ guarantees the realization of ψ. We can show that ψ_n is realized from ψ_1 in a finite number of steps:

ψ_1 **ensures** ψ_2,
ψ_2 **ensures** ψ_3,
$\ldots$,
ψ_{n-1} **ensures** ψ_n

The existence of such a sequence is stated by the operator $\varphi \mapsto \psi$:

$$\frac{\varphi \,\mathbf{ensures}\, \psi}{\varphi \mapsto \psi} \qquad \frac{\varphi \mapsto \chi \;\; \chi \mapsto \psi}{\varphi \mapsto \psi}$$

$$\frac{\varphi_1 \mapsto \psi \;\ldots\; \varphi_n \mapsto \psi}{(\varphi_1 \vee \ldots \vee \varphi_n) \mapsto \psi}$$

As opposed to the **ensures** operator, here φ is not required to remain true until ψ is realized. To briefly lay out how to prove a $\mapsto$ property, φ **ensures** ψ is proved by showing that every action a satisfies the

Table 1: $\vdash_M$: Properties of beliefs.

R1	if φ is an instance of a classical tautology, then $\vdash_M \varphi$
R2	$\vdash_C \phi \Longrightarrow \vdash_M \mathsf{B}\phi$
A1	$\vdash_M \mathsf{B}(\phi \longrightarrow \psi) \longrightarrow \mathsf{B}\phi \longrightarrow \mathsf{B}\psi$
A2	$\vdash_M \neg\mathsf{B}\bot$

Table 2: $\vdash_M$: Properties of goals.

A3	$\vdash_M \neg\mathsf{G}\bot$
A4	$\vdash_M \mathsf{B}\phi \longrightarrow \neg\mathsf{G}\phi$
A5	$\vdash_C \phi \longrightarrow \psi \Longrightarrow \vdash_M \neg\mathsf{B}\psi \longrightarrow (\mathsf{G}\phi \longrightarrow \mathsf{G}\psi)$

Table 3: The proof rules of our Hoare system.

Infeasible actions:	$\dfrac{\varphi \longrightarrow \neg \mathit{enabled}(a)}{\{\varphi\}\ a\ \{\varphi\}}$
Rule for conditional actions:	$\dfrac{\{\varphi \wedge \psi\}\ a\ \{\varphi'\} \qquad (\varphi \wedge \neg\psi) \longrightarrow \varphi'}{\{\varphi\}\ \psi \triangleright \mathit{do}(a)\ \{\varphi'\}}$
Consequence rule:	$\dfrac{\varphi' \longrightarrow \varphi \qquad \{\varphi\}\ a\ \{\psi\} \qquad \psi \longrightarrow \psi'}{\{\varphi'\}\ a\ \{\psi'\}}$
Conjunction rule:	$\dfrac{\{\varphi_1\}\ a\ \{\psi_1\} \qquad \{\varphi_2\}\ a\ \{\psi_2\}}{\{\varphi_1 \wedge \varphi_2\}\ a\ \{\psi_1 \wedge \psi_2\}}$
Disjunction rule:	$\dfrac{\{\varphi_1\}\ a\ \{\psi\} \qquad \{\varphi_2\}\ a\ \{\psi\}}{\{\varphi_1 \vee \varphi_2\}\ a\ \{\psi\}}$

Hoare triple $\{\varphi \wedge \neg\psi\}\ a\ \{\varphi \vee \psi\}$ and that the Hoare triple $\{\varphi \wedge \neg\psi\}\ a'\ \{\psi\}$ is satisfied by at least one action a'.

4 ISABELLE FORMALIZATION

This section describes the Isabelle/HOL formalization. So far we have formalized propositional logic, mental states, formulas over mental states and a proof system for mental state formulas. In Section 5, we will discuss future work on the formalization of the verification framework.

The Isabelle files are publicly available online:

https://people.compute.dtu.dk/aleje/public/

The file *Gvf_PL.thy* is a formalization of propositional logic. The file *Gvf_GOAL.thy* is the presented part of the formalization of GOAL and the verification framework.

4.1 Propositional Logic

Before we can get started on the more interesting parts of our formalization, we need a basis of propositional logic to build on. We formalize the language of propositional formulas, its semantics and a sequent calculus proof system.

We introduce natural numbers as a type for propositional symbols:

type-synonym *id* = *nat*

In line with usual textbook presentations it would have been straight-forward to use string symbols. In a theorem prover setting, working with natural numbers can be a lot more smooth, however.

4.1.1 Syntax

We define the formulas of propositional logic as a datatype with constructors for propositional symbols and the operators $\neg$, $\longrightarrow$, $\vee$ and $\wedge$:

datatype Φ_L =
Prop id |
Neg Φ_L (⟨$\neg_L$⟩) |
Imp Φ_L Φ_L (**infixr** ⟨$\longrightarrow_L$⟩ *60*) |
Dis Φ_L Φ_L (**infixl** ⟨$\vee_L$⟩ *70*) |
Con Φ_L Φ_L (**infixl** ⟨$\wedge_L$⟩ *80*)

We introduce infix notation for the logical operators with their usual precedence and $\longrightarrow$ being right-associative. A subscript is used to avoid conflicts with the built-in Isabelle logical operators.

4.1.2 Semantics

The semantics of formulas is evaluated from a model. Since the language is that of propositional logic, the model is merely an assignment of truth values to propositional symbols:

type-synonym *model* = ⟨*id* ⇒ *bool*⟩

We assume this assignment to be a function over propositional symbols returning a Boolean value.

Given a model and a formula, propositional symbols are simply looked up in the model f. For the operators, we recursively decompose the formula and use Isabelle's built-in logical operators to compute the truth value:

primrec sem_L :: ⟨*model* ⇒ Φ_L ⇒ *bool*⟩ **where**
⟨$sem_L\ f\ (Prop\ x) = f\ x$⟩ |
⟨$sem_L\ f\ (\neg_L\ p) = (\neg sem_L\ f\ p)$⟩ |
⟨$sem_L\ f\ (p \longrightarrow_L q) = (sem_L\ f\ p \longrightarrow sem_L\ f\ q)$⟩ |
⟨$sem_L\ f\ (p \vee_L q) = (sem_L\ f\ p \vee sem_L\ f\ q)$⟩ |
⟨$sem_L\ f\ (p \wedge_L q) = (sem_L\ f\ p \wedge sem_L\ f\ q)$⟩

We introduce the notion of entailment ($\vdash$) as an infix operator that takes two multisets of formulas:

abbreviation *entails* :: ⟨Φ_L *set* ⇒ Φ_L *set* ⇒ *bool*⟩
(**infix** ⟨$\models_C\#$⟩ *50*) **where**
⟨$\Gamma \models_C \# \Delta \equiv$
$(\forall f.\ (\forall p \in \Gamma.\ sem_L\ f\ p) \longrightarrow (\exists p \in \Delta.\ sem_L\ f\ p))$⟩

The abbreviation above encodes the usual understanding of entailment: for all models, at least one formula in the succedent should be true if all the formulas in the antecedent are true.

We introduce shorthand syntax for the special case where the right-hand side is a singleton set:

abbreviation *entails-singleton* :: ⟨Φ_L *set* ⇒ Φ_L ⇒ *bool*⟩
(**infix** ⟨$\models_C$⟩ *50*) **where**
⟨$\Gamma \models_C \Phi \equiv \Gamma \models_C \# \{\ \Phi\ \}$⟩

4.1.3 Sequent Calculus

We define a sequent calculus to derive tautologies. The inductive definition below is based on a standard sequent calculus for propositional logic with multisets on both sides of the turnstile:

inductive *seq* :: ⟨Φ_L *multiset* ⇒ Φ_L *multiset* ⇒ *bool*⟩
(**infix** ⟨$\vdash_C\#$⟩ *50*) **where**
⟨$\{\#\ p\ \#\} + \Gamma \vdash_C \# \Delta + \{\#\ p\ \#\}$⟩ |
⟨$\Gamma \vdash_C \# \Delta + \{\#\ p\ \#\} \Longrightarrow \Gamma + \{\#\ \neg_L\ p\ \#\} \vdash_C \# \Delta$⟩ |
⟨$\Gamma + \{\#\ p\ \#\} \vdash_C \# \Delta \Longrightarrow \Gamma \vdash_C \# \Delta + \{\#\ \neg_L\ p\ \#\}$⟩ |
⟨$\Gamma + \{\#\ p\ \#\} \vdash_C \# \Delta + \{\#\ q\ \#\} \Longrightarrow$
$\Gamma \vdash_C \# \Delta + \{\#\ p \longrightarrow_L q\ \#\}$⟩ |
⟨$\Gamma \vdash_C \# \Delta + \{\#\ p, q\ \#\} \Longrightarrow \Gamma \vdash_C \# \Delta + \{\#\ p \vee_L q\ \#\}$⟩ |
⟨$\Gamma + \{\#\ p, q\ \#\} \vdash_C \# \Delta \Longrightarrow \Gamma + \{\#\ p \wedge_L q\ \#\} \vdash_C \# \Delta$⟩ |
⟨$\Gamma \vdash_C \# \Delta + \{\#\ p\ \#\} \Longrightarrow \Gamma \vdash_C \# \Delta + \{\#\ q\ \#\} \Longrightarrow$
$\Gamma \vdash_C \# \Delta + \{\#\ p \wedge_L q\ \#\}$⟩ |
⟨$\Gamma + \{\#\ p\ \#\} \vdash_C \# \Delta \Longrightarrow \Gamma + \{\#\ q\ \#\} \vdash_C \# \Delta \Longrightarrow$
$\Gamma + \{\#\ p \vee_L q\ \#\} \vdash_C \# \Delta$⟩ |
⟨$\Gamma \vdash_C \# \Delta + \{\#\ p\ \#\} \Longrightarrow \Gamma + \{\#\ q\ \#\} \vdash_C \# \Delta \Longrightarrow$
$\Gamma + \{\#\ p \longrightarrow_L q\ \#\} \vdash_C \# \Delta$⟩

Again, we introduce a shorthand for a singleton set on the right-hand side:

abbreviation *seq-st-rhs* :: ⟨Φ_L *multiset* ⇒ Φ_L ⇒ *bool*⟩
(**infix** ⟨$\vdash_C$⟩ *50*) **where**
⟨$\Gamma \vdash_C \Phi \equiv \Gamma \vdash_C \# \{\#\ \Phi\ \#\}$⟩

4.1.4 Soundness

The soundness theorem for our sequent calculus can be stated nicely with entailment for multisets:

theorem *seq-sound*:
⟨$\Gamma \vdash_C \# \Delta \Longrightarrow$ *set-mset* $\Gamma \models_C \#$ *set-mset* Δ⟩
by (*induct rule*: *seq.induct*) (*auto*)

The function *set-mset* turns a multiset into a set. The proof is by induction of the sequent calculus rules and all subgoals (one for each rule) can be solved automatically without further specification.

We skip proving completeness. Firstly, it is much harder to prove and, more importantly, it does not serve a critical purpose in our further work. The soundness theorem is a strong result ensuring that we only prove valid formulas.

4.2 Mental States

We introduce a type for mental states. A mental state is a pair of sets of propositional logic formulas:

type-synonym *mst* = ⟨(Φ_L *set* × Φ_L *set*)⟩

Mental states have properties that constrain which sets of formulas qualify as belief or goal bases. Baking these properties into the type is rather complicated. We instead introduce them in a separate definition:

definition *is-mst* :: ⟨*mst* ⇒ *bool*⟩ (⟨∇⟩) **where**
⟨$\nabla\ x \equiv$ *let* $(\Sigma, \Gamma) = x$ *in*
$\Sigma\ \neg\models_C \bot_L \wedge (\forall \gamma \in \Gamma.\ \Sigma\ \neg\models_C \gamma \wedge \{\}\ \neg\models_C \neg_L\ \gamma)$⟩

Note that one important property of mental states, namely that subgoals of goals should also be goals, is left our of this definition. We will instead bake this into the semantics.

We can use the definition as an assumption when needed. We should keep in mind that if we manipulate a mental state, it will also be required to prove that mental state properties are preserved.

4.3 Mental State Formulas

The language of formulas over mental states are somewhat similar to the language for propositional formulas. However, instead of propositional symbols we may have belief or goal modalities.

4.3.1 Syntax

The syntax for mental state formulas is defined as a new datatype:

datatype Φ_M =
$B\ \Phi_L$ |
$G\ \Phi_L$ |

Neg Φ_M (⟨$\neg_M$⟩) |
Imp Φ_M Φ_M (**infixr** ⟨$\longrightarrow_M$⟩ *60*) |
Dis Φ_M Φ_M (**infixl** ⟨$\vee_M$⟩ *70*) |
Con Φ_M Φ_M (**infixl** ⟨$\wedge_M$⟩ *80*)

While the well-known logical operators are as before, the language of mental state formulas is defined on a level above the propositional language.

4.3.2 Semantics

We define the semantics of mental state formulas as a recursive function taking a mental state and a mental state formula:

primrec *semantics* :: ⟨*mst* ⇒ Φ_M ⇒ *bool*⟩ (**infix** ⟨$\models_M$⟩ *50*)
where
⟨$M \models_M (B\ \Phi) = (let\ (\Sigma, \text{-}) = M\ in\ \Sigma \models_C \Phi)$⟩ |
⟨$M \models_M (G\ \Phi) = (let\ (\Sigma, \Gamma) = M\ in$
$\Phi \in \Gamma \vee \Sigma \neg\models_C \Phi \wedge (\exists \gamma \in \Gamma.\ \{\} \models_C \gamma \longrightarrow_L \Phi))$⟩ |
⟨$M \models_M (\neg_M\ \Phi) = (\neg\ M \models_M \Phi)$⟩ |
⟨$M \models_M (\Phi_1 \longrightarrow_M \Phi_2) = (M \models_M \Phi_1 \longrightarrow M \models_M \Phi_2)$⟩ |
⟨$M \models_M (\Phi_1 \vee_M \Phi_2) = (M \models_M \Phi_1 \vee M \models_M \Phi_2)$⟩ |
⟨$M \models_M (\Phi_1 \wedge_M \Phi_2) = (M \models_M \Phi_1 \wedge M \models_M \Phi_2)$⟩

The belief modality for a given propositional formula is true if the formula is entailed by the belief base. For the goal modality, we require that the goal is either directly in the goal base or that it is a subgoal not entailed by the belief base. The remaining cases for logical operators are trivial. Let us elaborate on the choice of baking the subgoal property into the semantics. For any given goal, there are an infinite number of subgoals. If we require that these subgoals be part of the goal base set, no finite set will ever constitute a valid goal base. Working with finite goal bases is more convenient.

4.3.3 Properties of the Goal Modality

We state a lemma with the properties of the goal modality from 3.1.2:

lemma *G-properties*:
shows
⟨$\neg (\forall \Sigma\ \Gamma\ \Phi\ \psi.\ \nabla\ (\Sigma, \Gamma) \longrightarrow (\Sigma, \Gamma) \models_M$
$G\ (\Phi \longrightarrow_L \psi) \longrightarrow_M G\ \Phi \longrightarrow_M G\ \psi)$⟩
and
⟨$\neg (\forall \Sigma\ \Gamma\ \Phi\ \psi.\ \nabla\ (\Sigma, \Gamma) \longrightarrow (\Sigma, \Gamma) \models_M$
$G\ (\Phi \wedge_L (\Phi \longrightarrow_L \psi)) \longrightarrow_M G\ \psi)$⟩
and
⟨$\neg (\forall \Sigma\ \Gamma\ \Phi\ \psi.\ \nabla\ (\Sigma, \Gamma) \longrightarrow (\Sigma, \Gamma) \models_M$
$G\ \Phi \wedge_M G\ \psi \longrightarrow_M G\ (\Phi \wedge_L \psi))$⟩
and
⟨$\{\} \models_C \Phi \longleftrightarrow_L \psi \longrightarrow \nabla\ (\Sigma, \Gamma) \longrightarrow (\Sigma, \Gamma) \models_M$
$G\ \Phi \longleftrightarrow_M G\ \psi$⟩

The first three show that G is a weak logical operator. Each of those are proved by providing a counter-example. Let us consider the first property: G does not distribute over implication. We assume the contrary to hold:

assume ∗:
⟨$\forall \Sigma\ \Gamma\ \Phi\ \psi.\ \nabla\ (\Sigma, \Gamma) \longrightarrow (\Sigma, \Gamma) \models_M$
$G\ (\Phi \longrightarrow_L \psi) \longrightarrow_M G\ \Phi \longrightarrow_M G\ \psi$⟩

We then come up with a belief and goal base to use as a counter-example:

let $?\Sigma$ = ⟨{}⟩ **and** $?\Gamma$ = ⟨{ $?\Phi \longrightarrow_L ?\psi$, $?\Phi$ }⟩

The automation easily proves that in our example G does not distribute over implication:

have
⟨$\neg\ (?\Sigma, ?\Gamma) \models_M G\ (?\Phi \longrightarrow_L ?\psi) \longrightarrow_M G\ ?\Phi \longrightarrow_M G\ ?\psi$⟩
by *auto*

We further prove that the example belief and goal base in fact constitute a mental state:

moreover have ⟨$\nabla\ (?\Sigma, ?\Gamma)$⟩
unfolding *is-mst-def*
by *auto*

Combining the last two facts with our assumption, we show a contradiction and the proof is done:

ultimately show *False*
using ∗ **by** *blast*

The two next subgoals are analogous. The remaining last property is easily proved by Isabelle's simplifier.

4.4 Tautologies of Propositional Logic

The next large part of the formalization concerns the belief property *R1* from Table 1. The property states that any instantiation of a classical tautology is a tautology in the logic for GOAL.

To automate the step from a propositional formula to a mental state formula instance, we design an algorithm to substitute propositional symbols for belief or goal modalities:

primrec *conv* :: ⟨(id, Φ_M) *map* ⇒ Φ_L ⇒ Φ_M⟩ **where**
⟨$conv\ T\ (Prop\ p) = the\ (T\ p)$⟩ |
⟨$conv\ T\ (\neg_L\ \Phi) = (\neg_M\ (conv\ T\ \Phi))$⟩ |
⟨$conv\ T\ (\Phi_1 \longrightarrow_L \Phi_2) = (conv\ T\ \Phi_1 \longrightarrow_M conv\ T\ \Phi_2)$⟩ |
⟨$conv\ T\ (\Phi_1 \vee_L \Phi_2) = (conv\ T\ \Phi_1 \vee_M conv\ T\ \Phi_2)$⟩ |
⟨$conv\ T\ (\Phi_1 \wedge_L \Phi_2) = (conv\ T\ \Phi_1 \wedge_M conv\ T\ \Phi_2)$⟩

The algorithm requires some mapping of propositional symbols to mental state formulas. For simplicity, we can assume that each formula is either a belief or goal modality without losing any expressivity.

4.4.1 Automatic Substitution

For most practical applications, we will encounter top-down proofs starting from some mental state formula. In that case, the substitution is from belief and goal modalities to propositional symbols. To be able to determine if the mental state formula is an instantiation of a classical tautology, we need a one-to-one substitution. By this we mean that if and only

if modality operators and their contents are equal, then they map to the same propositional symbol. The choice of propositional symbol may be arbitrary. We design an algorithm to produce such a one-to-one mapping.

We exploit the built-in enumeration function in Isabelle:

abbreviation *bst* :: ⟨$\Phi_M \Rightarrow (id \times \Phi_M)$ *list*⟩ **where**
⟨*bst* $\varphi \equiv$ *enumerate 0* (*atoms* φ)⟩

The function *atoms* collect all belief and goal modalities, and *enumerate* builds a list of pairs of natural numbers and mental state formulas, starting from 0 and incrementing by 1 for each formula.

We substitute the modalities by the enumeration of propositional symbols:

definition *to-L* :: ⟨$\Phi_M \Rightarrow \Phi_L$⟩ **where**
⟨*to-L* $\varphi \equiv$ *to-L'* (*map-swap* (*bst* φ)) φ⟩

Here, *to-L'* is defined similarly to *conv* except it produces a propositional formula.

4.5 Proof System

The belief and goal properties are collected to form a proof system that is inductively defined:

inductive *derive* :: ⟨$\Phi_M \Rightarrow$ *bool*⟩ (⟨$\vdash_M$⟩) **where**
R1: ⟨*conv* (*map-of T*) $\varphi' = \varphi \Longrightarrow \{\#\} \vdash_C \varphi' \Longrightarrow \vdash_M \varphi$⟩ |
R2: ⟨$\{\#\} \vdash_C \Phi \Longrightarrow \vdash_M (B\ \Phi)$⟩ |
A1: ⟨$\vdash_M (B\ (\Phi \longrightarrow_L \psi) \longrightarrow_M (B\ \Phi \longrightarrow_M B\ \psi))$⟩ |
A2: ⟨$\vdash_M (\neg_M (B\ \bot_L))$⟩ |
A3: ⟨$\vdash_M (\neg_M (G\ \bot_L))$⟩ |
A4: ⟨$\vdash_M ((B\ \Phi) \longrightarrow_M (\neg_M (G\ \Phi)))$⟩ |
A5: ⟨$\{\#\} \vdash_C (\Phi \longrightarrow_L \psi) \Longrightarrow \vdash_M (\neg_M (B\ \psi) \longrightarrow_M (G\ \Phi) \longrightarrow_M (G\ \psi))$⟩

The definition of *R1* requires some explanation. If there exists a substitution from a propositional formula Φ' to a mental state formula Φ, and if Φ' is provable in the sequent calculus, then we are allowed to conclude Φ.

4.5.1 Correctness of Automatic Substitution

Before we start work on proving soundness for the proof system, let us first come back to our automatic substitution for top-down proofs. The following lemma states that if the generated propositional formula from Φ is provable, then Φ is also provable:

lemma *to-L-correctness*: ⟨$\{\#\} \vdash_C$ *to-L* $\varphi \longrightarrow \vdash_M \varphi$⟩

We will merely sketch the proof: we apply induction over the structure of Φ and prove it for each case. The base cases and negation are proved automatically. The branching cases require some extra work as the function *bst* will produce overlapping enumerations for subformulas. We assist the automation by providing a lemma showing that the generated mappings for subformulas can be replaced by that of the parent formula, e.g. $\Phi_1 \wedge \Phi_2$ for subformulas Φ_1 and Φ_2.

4.5.2 Soundness

The soundness theorem for the proof system is:

theorem *derive-soundness*:
assumes ⟨∇M⟩
shows ⟨$\vdash_M \Phi \Longrightarrow M \models_M \Phi$⟩

The proof is by induction over the rules:

proof (*induct rule*: *derive.induct*)

We will go into details with the case for *R1*. We get to assume that Φ' is provable, and that some mapping T exists to a mental state formula Φ:

case (*R1 T* φ' φ)

Due to the soundness result for the sequent calculus, we know that formula is true for any model. We get the idea to focus on a model in which the truth value of propositional symbols is derived directly from semantics of the belief or goal modality it maps to in the mental state:

then have ⟨sem_L (λx. *semantics M* (*the* (*map-of T x*))) φ'⟩
using *seq-sound* **by** *fastforce*

By induction over the structure of Φ', we show that this is equal to the semantics of the Φ for the given mental state:

moreover have
⟨sem_L (λx. *semantics M* (*the* (*map-of T x*))) $\varphi' =$
($M \models_M$ (*conv* (*map-of T*) φ'))⟩
by (*induct* φ') *simp-all*
ultimately show *?case*
using ⟨*conv* (*map-of T*) $\varphi' = \varphi$⟩ **by** *simp*

We invite the interested reader to study the available theory files for further details, although some prior experience using Isabelle may be required.

5 DISCUSSION

The work on the formalization still has a long way to go before we can start concluding on the process as a whole. So far our formalization covers the theory up the point of mental state formulas and a proof system. What remains is to formalize actions, Hoare triples and a temporal logic for GOAL, and how to prove temporal properties such as correctness.

We have successfully been able to verify some of the results in (de Boer et al., 2007). The progress shows promise and everything points towards it being feasible to verify all results of the paper.

We have not touched much on the limitations of the verification framework in this paper. In order to give the framework more appeal, we need to find

ways to deal with current limitations of the framework, namely:

- not being able to model or prove statements quantifying over mental states,
- not modelling the environment by assuming that the agent is the only actor,
- not accounting for multiple agents.

It is clear that neither of these limitations have an easy fix. It is not obvious how to quantify over mental states: we need to consider the temporal aspect of the agent system and prove that a property holds across several mental states. We have not encountered any proposals in the literature. Modelling the environment seems less of a challenge but potentially adds little immediate value beyond providing a better level of abstraction. Even if we alleviate assumptions of complete knowledge for agents, we still need to add some domain knowledge into the environment to be able to make any logical reasoning of interest. In a way, it simply pushes part of the problem into another structure, but could potentially be a step towards being able to state richer assumptions about interactions in the environment. Lastly, we want to discuss modelling of multiple agents and of communication. This is very much at the core of the issue in the field of verifying multi-agent systems. The nature of multiple parallel processes quickly makes every model explode in complexity. We acknowledge its importance but it is too early to share any thoughts on its role in our continued work, and how to resolve the problems that may arise.

6 CONCLUSIONS

We have argued that the use of theorem proving can be a valuable tool for developing formal verification techniques to ensure the reliability of multi-agent systems. The value comes from assuring that developed techniques work as intended and does not introduce new levels of possible errors. Such errors may compromise the reliability of the system even further by providing a false sense of security.

We have described a verification framework for the GOAL agent programming language that can be used to prove properties of agents. We have proved some of the results of the author's original work using the Isabelle/HOL proof assistant, and deemed it feasible to formalize all of the results.

Finally, we have discussed the many challenges ahead and given ideas to address them. Our plans are to continue work on the Isabelle formalization and simultaneously work on extending the theory.

ACKNOWLEDGEMENTS

I would like to thank Asta H. From for Isabelle related discussions and for comments on drafts of this paper. I would also like to thank Jrgen Villadsen for comments on drafts of this paper. I would also like to thank Koen V. Hindriks for helpful insights.

REFERENCES

Alechina, N., Dastani, M., Khan, A. F., Logan, B., and Meyer, J.-J. (2010). *Using Theorem Proving to Verify Properties of Agent Programs*, pages 1–33. Springer.

Bordini, R. H., Fisher, M., Visser, W., and Wooldridge, M. (2006). Verifying Multi-agent Programs by Model Checking. *Autonomous Agents and Multi-Agent Systems*, 12:239–256.

Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., and Russo, W. (2008). PASSIM: A simulation-based process for the development of multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 2:132–170.

de Boer, F. S., Hindriks, K. V., Hoek, W., and Meyer, J.-J. (2007). A verification framework for agent programming with declarative goals. *Journal of Applied Logic*, 5:277–302.

Dennis, L. A. and Fisher, M. (2009). Programming Verifiable Heterogeneous Agent Systems. In *Programming Multi-Agent Systems*, pages 40–55. Springer.

Dix, J., Logan, B., and Winikoff, M. (2019). Engineering Reliable Multiagent Systems (Dagstuhl Seminar 19112). *Dagstuhl Reports*, 9(3):52–63.

Hindriks, K. V. (2009). *Programming Rational Agents in GOAL*, pages 119–157. Springer US.

Hindriks, K. V., Boer, F., Hoek, W., and Meyer, J.-J. (1999). Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2:357–401.

Jensen, A. B. (2021). Towards Verifying a Blocks World for Teams GOAL Agent. In *ICAART 2021 - Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. SciTePress. To appear.

Jensen, A. B., Hindriks, K. V., and Villadsen, J. (2021). On Using Theorem Proving for Cognitive Agent-Oriented Programming. In *ICAART 2021 - Proceedings of the 13th International Conference on Agents and Artificial Intelligence*. SciTePress. To appear.

Misra, J. (1994). A Logic for Concurrent Programming. Technical report, Formal Aspects of Computing.

Nipkow, T., Paulson, L., and Wenzel, M. (2002). *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer.

Paulson, L. C. (2000). Mechanizing UNITY in Isabelle. *ACM Trans. Comput. Logic*, 1(1):332.

Poutakidis, D., Winikoff, M., Padgham, L., and Zhang, Z. (2009). *Debugging and Testing of Multi-Agent Systems using Design Artefacts*, pages 215–258. Springer.