A Verification Framework for GOAL Agents

Jensen, Alexander Birch

Published in:
Pre-Proceedings of 8th International Workshop on Engineering Multi-Agent Systems

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Pre-Proceedings of 8th International Workshop on Engineering Multi-Agent Systems
(EMAS 2020 @ AAMAS), 8-9 May 2020, Auckland, New Zealand.
Eds. Cristina Baroglio, Jomi F. Hübner, and Michael Winikoff.

A Verification Framework for GOAL Agents

Alexander Birch Jensen

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

Abstract. We describe a project about formal verification of GOAL agents. We explain how to mechanically transform GOAL code into agent logic under a set of assumptions. The framework allows us to prove correctness of an implemented GOAL agent. We focus on a Blocks World for Teams problem and show the first step of a correctness proof. Finally, we sketch future steps for the project by identifying the many challenges.

Keywords: Agent programming · Agent verification · Formal logic

1 Introduction

Current approaches to programming agents are inspired by the agent-oriented programming paradigm [12], and most common frameworks are based on the Belief-Desire-Intention (BDI) model [10]. Two popular examples are JADE for the Java-platform, and GOAL which is closely tied to Prolog and logic programming [1,7,5]. One of the current great challenges of the multi-agent system research field is the quest to verify their correctness [6]. Research efforts are being put into the model checking approach where one verifies desired properties in the possible states of the system [3].

In this paper, we suggest a different approach based on an agent logic to verify agents. This is a continuation of work by de Boer, Hindriks, van der Hoek and Meyer on a framework for verifying GOAL agents [2]. The framework is a complete programming theory for GOAL – it provides a formal, operational semantics for the GOAL language and a temporal logic proof theory. The framework aims to close the gap between agent logics and agent programming. Sections 3 and 4 briefly summarizes this work informally.

The research project covered here applies said verification framework to an instance of a Blocks World for Teams (BW4T) problem [8]. We consider BW4T to be an ideal scenario to focus this research around.

The first main contribution of this paper is describing a transformation method from GOAL program code to agent logic. As a second main contribution, we sketch the future steps of the project by identifying the many challenges in extending the verification framework. We only show a first step of the correctness proof for our BW4T agent, and we do not have the full proof yet. Thus, we consider the shown proof step a minor contribution.

2 The BW4T Environment

In the BW4T environment the goal is to collect blocks, located in different rooms, in a particular sequence of colors. Pathfinding between rooms is handled by the environment server. For now, we consider a single-agent case. The relevant percepts are: *in(RoomId)*: agent is in room *RoomId*, *atBlock(BlockId)*: agent is at block *BlockId*, *holding(BlockId)*: agent is holding block *BlockId*, *color(BlockId, Color, RoomId)*:[1] block *BlockId* has color *Color* and is in room *RoomId*. The only action considered is *goTo(RoomId)*: agent moves to room *RoomId*.

Following the Russel & Norvig categorization of environments [11], BW4T can (for our configuration) be categorized as a single-agent, partially observable, deterministic environment. This categorization plays an essential part in allowing the environment to be modelled by the verification framework. We will assume the environment to be fully observable which in turn is ensured by running on a predefined map.

3 GOAL Agent Programming

Mental States We think of agents as having mental states. A mental state consists of a belief base and a goal base of mental state formulas. The language is that of propositional formulas. Mental state formulas extend the propositional language with the operators B and G. the formula $\mathsf{B}\phi$ can be thought of as a query to the belief base of the mental state of the agent. The formula is true if ϕ follows from the current state of beliefs in the agent. Likewise, $\mathsf{G}\phi$ can be thought of as a query to the goal base of the mental state of the agent. The formula is true if ϕ is a current goal.

Actions Consider first two special actions *adopt*(ϕ) and *drop*(ϕ) that are used to adopt or drop goals, respectively. These actions are always part of the agent's capabilities. The environment makes available a number of actions to the agent.

Intuitively, a conditional action extends the notion of a basic action, or capability, by giving a condition on the mental state for it to be executable. We do not cover the formal distinction between basic and conditional actions in details. A conditional action may not be enabled in a given mental state, and we denote the condition of a conditional action a as *enabled(a)*.

GOAL Agent A GOAL agent consists of a set of conditional actions and the agent's initial mental state. The mental state is updated due to the agent executing an action. This plays into our assumption of a single-agent, deterministic environment. How an action updates the mental state is defined by a partial function (mental state transformer) that associates agent capabilities with updates to the belief base. Note that the environment is not part of this model.

See [2] for details about formalizing the GOAL programming language.

4 Proof Theory for GOAL Agents

Temporal Logic We construct a temporal logic on top of GOAL. It differs from regular definitions of temporal logic by having its semantics derived from the semantics of GOAL and by incorporating the belief and goal operators. Hoare triples are used to specify actions – the Hoare triple $\{\varphi\}\ \rho\ \triangleright\ do(a)\ \{\psi\}$ specifies a conditional action a with precondition φ and postcondition ψ where φ and ψ are mental state formulas.

Basic Action Theory A basic action theory specifies the effects of agent capabilities by associating Hoare triples with effect axioms. By means of an *enabled* predicate for each action the conditions on actions are also specified. Additionally, the theory specifies what does not change following execution of actions by associating Hoare triples with frame axioms. The effect and frame axioms are not part of the GOAL language and should be specified by the user – in our case this is partly derived from inspection of the program code. Axioms for the special actions `drop` and `adopt` are static and provided by the GOAL language.

Proof System A complete Hoare system for GOAL is given by the proof in Tables 4-10 in [2]. We note that readers should study the rule for conditional actions and also the structural rules as we do not have the space to display or explain them here.

5 Transformation from GOAL Agent Code

In order to make the link between GOAL code and the framework, we describe how to perform a mechanical transformation from GOAL code to the agent logic. To adhere to current limitations of the framework, we assume the following:

(1) Single agent assumption: only the agent executes actions.
(2) The environment is deterministic: an action has a deterministic outcome.
(3) The starting configuration is known (part of the initial mental state).
(4) Action execution is instant (no durative actions).

We further impose restriction on the structure of the GOAL code, notably:

(1) The main module should only have rules of the form `if` φ `then` a where φ is a query to the mental state and a is an action in the environment.
(2) The preconditions of an action should specify the minimal condition under which the action is enabled (action specification).
(3) Annotate environment interaction code with an action name (event module).

BW4T Transformation Hoare triples are derived from the GOAL code by transforming the relevant parts of the action specification and the event and main module. We explain the method by means of a running example where we transform code for the `goTo` action.

Consider first the action specification for `goTo`:

```
define goTo(X) with pre { true } post { true }
```

Perhaps counter-intuitively, with regards to the verification framework, the effects of actions are perceived through changes in the environment and are left empty (simply *true*). The precondition specifies the minimal conditions for action execution – it is always possible to go to another room. We will continuously work on our Hoare triple until the final triple is derived. For now, this leaves with the rather simple Hoare triple (note that *enabled* is not yet specified):

$$\{\, true \,\}\ \ enabled(goTo(X))\ \triangleright\ do(goTo(X))\ \ \{\, true \,\}$$

Percept handling and environment interactions is a general issue in verification frameworks that has not been addressed effectively. Since the agent has full control, we can model the changes to the environment brought about by actions by mapping code in the event module to pre- and postconditions of actions. Below is the code (annotated with the *goTo* action name) from the event module:

```
if bel(in(X)), not(percept(in(X))) then delete(in(X)).
if percept(in(X)), not(bel(in(X))) then insert(in(X)).
```

While not apparent from the code, the room X in the first line will be different from that of the second line (given the assumption that the agent should move to a different room). This is a first example of the need for human interaction (intelligence) in the transformation. We further specify our Hoare triple:

$$\{\, \mathsf{B}(in(Y) \wedge \neg in(X)) \,\}\ \ enabled(goTo(X))\ \triangleright\ do(goTo(X))\ \ \{\, \mathsf{B}(in(X) \wedge \neg in(Y)) \,\}$$

Since $\mathsf{B}(in(X) \wedge \neg in(X))$ is never true, we implicitly ensure that $X \neq Y$ holds. The final step of deriving the Hoare triple, in its general form, is to transform code from the main module to the specification for $enabled(goTo(\mathrm{X}))$. That is, the condition under which the action should be enabled for the agent.

```
if goal(collect(C)), bel(not(state(traveling)),
        not(holding(_)), color(_, C, X)),
        not(bel(in(Y), color(_, C, Y))) then goTo(X).
```

We use `_` for variables we do not care about. Since this is not possible in the verification framework, we instead introduce a variable:

$$\{\, \mathsf{B}(in(Y) \wedge \neg in(X)) \,\}\ \ \mathsf{G}(collect(C)) \wedge \mathsf{B}(\neg holding(U) \wedge color(V, C, X)))$$
$$\wedge\, \neg \mathsf{B}(in(Y) \wedge color(W, C, Y))\ \triangleright\ do(goTo(X))\ \ \{\, \mathsf{B}(in(X) \wedge \neg in(Y)) \,\}$$

Also in this step it is not clear from the code which variables from the pre- and postcondition that match those of the action condition. Thus, this is the second example in which we need to apply intelligence to transform the code properly.

The verification framework is propositional in nature. Thus the Hoare triple above should be instantiated with propositional symbols. To keep things simple, consider the following initial state (capturing a starting configuration):

$$\mathsf{B}color(b_a, red, r_1) \wedge \mathsf{B}in(r_0) \wedge \mathsf{G}collect(red)$$

In the above, r_0 is some initial position of the agent. Note that this example only has a single execution trace. This is due to the simplicity of the given example – a larger example could have multiple traces.

Below is the relevant Hoare triple instantiation for going from r_0 to r_1:

$$\{\, \mathsf{B}(in(r_0) \wedge \neg in(r_1)) \,\} \quad \mathsf{G}(collect(red)) \wedge \mathsf{B}(\neg holding(b_a) \wedge color(b_a, red, r_1))) \\ \wedge \neg \mathsf{B}(in(r_0) \wedge color(b_a, red, r_0)) \ \triangleright\ do(goTo(r_1)) \quad \{\, \mathsf{B}(in(r_1) \wedge \neg in(r_0)) \,\}$$

In principle, the Hoare triples are instantiated with all possible combinations.

6 Proving Correctness of GOAL Agents

The proof of correctness consists of proofs for a number of **ensures** formulas. Together they prove that the agent reaches its goal in a finite number of steps. The operator is defined as φ **ensures** $\psi := (\varphi \rightarrow (\varphi$ **until** $\psi)) \wedge (\varphi \rightarrow \Diamond\psi)$. Informally, φ **ensures** ψ means that φ guarantees the realization of ψ.

The operator $\varphi \mapsto \psi$ is the transitive, disjunctive closure of **ensures**:

$$\frac{\varphi \textbf{ ensures } \psi}{\varphi \mapsto \psi} \qquad \frac{\varphi \mapsto \chi \ \ \chi \mapsto \psi}{\varphi \mapsto \psi} \qquad \frac{\varphi_1 \mapsto \psi \ \ldots \ \varphi_n \mapsto \psi}{(\varphi_1 \vee \ldots \vee \varphi_n) \mapsto \psi}$$

It differs from **ensures** as φ is not required to remain true until ψ is realized.

The proof of a formula φ **ensures** ψ requires that we show that every action a satisfies the Hoare triple $\{\, \varphi \wedge \neg\psi \,\}\ a\ \{\, \varphi \vee \psi \,\}$ and that there is at least one action a' which satisfies the Hoare triple $\{\, \varphi \wedge \neg\psi \,\}\ a'\ \{\, \psi \,\}$.

BW4T Agent Correctness Proof In the shown proof step we may use variables informally – these are to be thought of as instantiated with propositional symbols. For all actions a different from `goTo`, we supply the frame axiom $\{\, \mathsf{B}in(X) \,\}\ a\ \{\, \mathsf{B}in(X) \,\}$. The frame axiom states that only the action `goTo` can change the agent's belief about its current position.

The following invariant *inv-in*: $\mathsf{B}\,(\forall r,\, r'((in(r) \wedge r \neq r') \longrightarrow \neg in(r')))$ states that we can only be in one place at any given time. Invariants must be proved in the proof theory – we do not have the space to cover the proof in this paper.

It turns out the correctness property for the BW4T agent is the formula:

$$\mathsf{B}color(b_a, red, r_1) \wedge \mathsf{B}in(r_0) \wedge \mathsf{G}collect(red) \mapsto \mathsf{B}collect(red)$$

We show here only part of the first step in the proof. It reflects that the agent will first move from its initial position to the room containing the red block.

(1) $\mathsf{B}color(b_a, red, r_1) \wedge \mathsf{B}in(r_0) \wedge \neg\mathsf{B}holding(b_a) \wedge \mathsf{G}collect(red)$
ensures
$\mathsf{B}color(b_a, red, r_1) \wedge \underline{\mathsf{B}in(r_1)} \wedge \neg\mathsf{B}holding(b_a) \wedge \mathsf{G}collect(red)$

We have underlined the changes in the mental state as reflected by the formulas. For sketching the proof, we refer to the formula above as φ_{r_0} **ensures** ψ_{r_1}.

Effect of $goTo(r_1)$ We need to prove that $goTo(r_1)$ gives the desired state, i.e. the following Hoare triple must be satisfied:

$$\{\, \varphi_{r_0} \wedge \neg\psi_{r_1} \,\}\ enabled(goTo(r_1)) \ \triangleright\ do(goTo(r_1))\ \{\, \psi_{r_1} \,\}$$

By applying the rule for conditional actions we are required to prove the formula $(\varphi_{r_0} \wedge \neg enabled(goTo(r_1))) \rightarrow \psi_{r_1}$. By unfolding the formula, it is trivial

to show that it is a tautology. Furthermore, we need to prove the Hoare triple:

$$\{\, \varphi_{r_0} \wedge \neg\psi_{r_1} \wedge \mathit{enabled}(goTo(r_1)) \,\}\ \ \mathit{do}(goTo(r_1))\ \ \{\, \psi_{r_1} \,\}$$

We will merely sketch the remaining steps of the subproof: By the structural rules, we strengthen the precondition and weaken the postcondition (using the invariant *inv-in*). Then by the conjunction rule we split the Hoare triple and arrive at Hoare triples that are frame or effect axioms, and we are done.

7 Future Steps

A sketch of future steps are given by identifying challenges going forward. Beyond these, we should finalize the correctness proof. To our knowledge, there are no existing proofs of implemented GOAL agents using verification frameworks.

Challenge 1: Complex Scenarios. We should experiment with more complex scenarios, i.e. additional blocks, rooms and goal sequences of multiple colors.

Challenge 2: Starting Configuration. We assume knowledge about the starting configuration to enable transformation of the GOAL code into agent logic. Relaxing this constraint requires that we incorporate some notion of environment inaction. It likely also requires that we are able to reason with variables.

Challenge 3: Non-deterministic Environment. We currently assume the environment to be deterministic. Often environments will have non-deterministic action outcomes. Supporting this is an important issue to address.

Challenge 4: Multiple Communicating Agents. To enhance the utility of the framework, it is essential that we are able to deal with scenarios involving multiple agents. Communication between agents also plays a central role here.

Challenge 5: Durative Actions The aspect of time, in particular duration of actions, is often involved in practical applications of multi-agent system.

We expect Challenge 1 to be achievable without extending the framework. Challenge 2-5 requires work on extending the framework. For challenge 4 some existing work exists on the topic that needs to studied further [4].

Extending the framework to deal with the identified challenges is a non-trivial task. It will likely require the use of higher-order logic to effectively address the issues imposed by the challenges. It may be useful to have a tool in which we can experiment more freely with the formalism of the framework. This serves as a first motivation for us to formalize the framework in the interactive proof assistant Isabelle/HOL [9]. Isabelle/HOL is a theorem prover based on higher-order logic. We see the main benefits of using a theorem prover to be two-fold. Firstly, we will be able to work with examples in the verification framework interactively. Secondly, having a computer-checked proof of the correctness verification framework itself is valuable. This will also help prevent errors being introduced into the framework as we experiment with extensions.

8 Conclusion

We have presented a project on verification of GOAL agent programs using a verification framework. We explained a method for transformation of GOAL program code into agent logic for which a temporal logic proof theory is used to prove properties of the agents. We exemplified this using a BW4T scenario.

We identified key challenges to address: complex scenarios, starting configuration assumptions, non-deterministic environments, multiple communicating agents and durative actions. The future steps are to address these issues.

A notable characteristic of the agent logic is that it is directly linked to an implemented GOAL agent. We consider this relation a foundation for developing verification techniques that are desirable for programmers.

In conclusion, the use of verification frameworks is an interesting alternative to current popular approaches for verifying multi-agent systems.

Acknowledgements

The presented paper is a result of research efforts with Koen Hindriks and Jørgen Villadsen. Asta From has helped proofread drafts. Jørgen Villadsen is the main supervisor of my PhD project and Sebastian Mödersheim is the co-supervisor.

References

1. Bellifemine, F., Poggi, A., Rimassa, G.: JADE: A FIPA2000 Compliant Agent Development Environment. AGENTS, 216—217, ACM (2001)
2. de Boer, F., Hindriks, K., van der Hoek, W., Meyer, J.J.: A verification framework for agent programming with declarative goals. Journal of Applied Logic **5**(2), 277 – 302 (2007)
3. Bordini, R.H., Fisher, M., Visser, W., Wooldridge, M.: Verifying Multi-agent Programs by Model Checking. AAMAS **12**(2), 239–256 (2006)
4. Bulling, N., Hindriks, K.V.: Towards a Verification Framework for Communicating Rational Agents. MATES, 177–182, Springer (2009)
5. Clocksin, W.F., Mellish, C.S.: Programming in Prolog: Using the ISO standard. Springer (2012)
6. Dix, J., Logan, B., Winikoff, M.: Engineering Reliable Multiagent Systems (Dagstuhl Seminar 19112). Dagstuhl Reports **9**(3), 52–63 (2019)
7. Hindriks, K.V., de Boer, F.S., van der Hoek, W., Meyer, J.J.C.: Agent Programming with Declarative Goals. In: Castelfranchi, C., Lespérance, Y. (eds.) Intelligent Agents VII, 228–243, Springer (2001)
8. Johnson, M., Jonker, C., Riemsdijk, M., Feltovich, P.J., Bradshaw, J.: Joint Activity Testbed: Blocks World for Teams (BW4T). ESAW, 254–256, Springer (2009)
9. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic. Springer (2002)
10. KR'91 Rao, A.S., Georgeff, M.P.: Modeling Rational Agents within a BDI-Architecture. KR'91, 473—484, Morgan Kaufmann (1991)
11. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, 3rd edn. (2009)
12. Shoham, Y.: Agent-oriented programming. Artificial Intelligence **60**(1), 51 – 92 (1993)