# Team Programming in Golog under Partial Observability

**Alessandro Farinelli**
DIS, Università di Roma
"La Sapienza"
00198 Rome, Italy

**Alberto Finzi** *
Institut für Informations-
systeme, TU Wien
1040 Vienna, Austria

**Thomas Lukasiewicz** †
DIS, Università di Roma
"La Sapienza"
00198 Rome, Italy

## Abstract

In this paper, we present the agent programming language TEAMGOLOG, which is a novel approach to programming a team of cooperative agents under partial observability. Every agent is associated with a partial control program in Golog, which is completed by the TEAMGOLOG interpreter in an optimal way by assuming a decision-theoretic semantics. The approach is based on the key concepts of a synchronization state and a communication state, which allow the agents to passively resp. actively coordinate their behavior, while keeping their belief states, observations, and activities invisible to the other agents. We show the usefulness of the approach in a rescue simulated domain.

## 1 Introduction

During the recent years, the development of controllers for autonomous agents has become increasingly important in AI. One way of designing such controllers is the programming approach, where a control program is specified through a language based on high-level actions as primitives. Another way is the planning approach, where goals or reward functions are specified and the agent is given a planning ability to achieve a goal or to maximize a reward function. An integration of both approaches has recently been proposed through the seminal language DTGolog [Boutilier *et al.*, 2000], which integrates explicit agent programming in Golog [Reiter, 2001] with decision-theoretic planning in (fully observable) MDPs [Puterman, 1994]. It allows for partially specifying a control program in a high-level language as well as for optimally filling in missing details through decision-theoretic planning, and it can thus be seen as a decision-theoretic extension to Golog, where choices left to the agent are made by maximizing expected utility. From a different perspective, it can also be seen as a formalism that gives advice to a decision-theoretic planner, since it naturally constrains the search space.

DTGolog has several other nice features, since it is closely related to first-order extensions of decision-theoretic planning (see especially [Boutilier *et al.*, 2001; Yoon *et al.*, 2002; Guestrin *et al.*, 2003]), which allow for (i) compactly representing decision-theoretic planning problems without explicitly referring to atomic states and state transitions, (ii) exploiting such compact representations for efficiently solving large-scale problems, and (iii) nice properties such as *modularity* (parts of the specification can be easily added, removed, or modified) and *elaboration tolerance* (solutions can be easily reused for similar problems with few or no extra cost).

However, DTGolog is designed only for the single-agent framework. That is, the model of the world essentially consists of a single agent that we control by a DTGolog program and the environment summarized in "nature". But there are many applications where we encounter multiple agents that cooperate with each other. For example, in *robotic rescue*, mobile agents may be used in the emergency area to acquire new detailed information (such as the locations of injured people in the emergency area) or to perform certain rescue operations. In general, acquiring information as well as performing rescue operations involves several and different rescue elements (agents and/or teams of agents), which cannot effectively handle the rescue situation on their own. Only the cooperative work among all the rescue elements may solve it. Since most of the rescue tasks involve a certain level of risk for humans (depending on the type of rescue situation), mobile agents can play a major role in rescue situations, especially teams of cooperative heterogeneous mobile agents.

Another crucial aspect of real-world environments is that they are typically only partially observable, due to noisy and inaccurate sensors, or because some relevant parts of the environment simply cannot be sensed. For example, especially in the robotic rescue domain described above, every agent has generally only a very partial view on the environment.

The practical importance of controlling a system of cooperative agents under partial observability by a generalization of DTGolog has already been recognized in recent works by Ferrein *et al.* [2005] and Finzi and Lukasiewicz [2006]. A drawback of these two works, however, is that they are implicitly centralized by the assumption of a global world model resp. the assumption that every agent knows the belief states, observations, and actions of all the other agents (and so [Ferrein *et al.*, 2005; Finzi and Lukasiewicz, 2006] have no explicit communication between the agents), which is very often not possible or not desirable in realistic applications.

In this paper, we present the agent programming language

*Alternate address: DIS, Università di Roma "La Sapienza".
†Alternate address: Institut für Informationssysteme, TU Wien.

TEAMGOLOG, which is a novel generalization of DTGolog for controlling a system of cooperative agents under partial observability, which does not have such centralization assumptions. It is thus guided by the idea of truly distributed acting in multi-agent systems with a minimal interaction between the agents. The main contributions are as follows:

- We introduce the agent programming language TEAMGOLOG for controlling a system of cooperative (middle-size) agents under partial observability. We define a decision-theoretic semantics of TEAMGOLOG, which are inspired by *decentralized partially observable MDPs (Dec-POMDPs)* [Nair *et al.*, 2003; Goldman and Zilberstein, 2004].
- We introduce the concepts of a *synchronization state* and a *communication state*, which are used to coordinate the agents, taking inspiration from *artificial social systems* [Shoham and Tennenholtz, 1995]: The behavior of each agent is encoded in advance in its domain theory and program, and depends on the online trace of synchronization and communication states.
- We define an interpreter for TEAMGOLOG, and provide a number of theoretical results around it. In particular, we show that the interpreter generates optimal policies.
- We describe a first prototype implementation of the TEAMGOLOG interpreter. We also provide experimental results from the rescue simulation domain, which give evidence of the usefulness of our approach in realistic applications.

Notice that detailed proofs of all results in this extended abstract are provided in the full version of this paper.

## 2 The Situation Calculus and Golog

The situation calculus [McCarthy and Hayes, 1969; Reiter, 2001] is a first-order language for representing dynamic domains. Its main ingredients are *actions*, *situations*, and *fluents*. An *action* is a first-order term of the form $a(\vec{u})$, where $a$ is an action name, and $\vec{u}$ are its arguments. For example, $moveTo(r, x, y)$ may represent the action of moving an agent $r$ to the position $(x, y)$. A *situation* is a first-order term encoding a sequence of actions. It is either a constant symbol or of the form $do(a, s)$, where $a$ is an action and $s$ is a situation. The constant symbol $S_0$ is the *initial situation* and represents the empty sequence, while $do(a, s)$ encodes the sequence obtained from executing $a$ after the sequence of $s$. For example, $do(moveTo(r, 1, 2), do(moveTo(r, 3, 4), S_0))$ stands for executing $moveTo(r, 1, 2)$ after executing $moveTo(r, 3, 4)$ in $S_0$. A *fluent* represents a world or agent property that may change when executing an action. It is a predicate symbol whose most right argument is a situation. For example, $at(r, x, y, s)$ may express that the agent $r$ is at the position $(x, y)$ in the situation $s$. In the situation calculus, a dynamic domain is encoded as a *basic action theory* $AT = (\Sigma, \mathcal{D}_{S_0}, \mathcal{D}_{ssa}, \mathcal{D}_{una}, \mathcal{D}_{ap})$, where:

- $\Sigma$ is the set of foundational axioms for situations.
- $\mathcal{D}_{una}$ is the set of *unique name axioms for actions*, saying that different action terms stand for different actions.
- $\mathcal{D}_{S_0}$ is a set of first-order formulas describing the *initial state of the domain* (represented by $S_0$). E.g., $at(r, 1, 2, S_0)$ may express that the agent $r$ is initially at the position $(1, 2)$.
- $\mathcal{D}_{ssa}$ is the set of *successor state axioms* [Reiter, 2001]. For each fluent $F(\vec{x}, s)$, it contains an axiom $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is a formula with free variables among $\vec{x}$, $a$, and $s$. These axioms specify the truth of the fluent $F$ in the next situation $do(a, s)$ in terms of the current situation $s$, and are a solution to the frame problem (for deterministic actions). For example,

$$at(o, x, y, do(a, s)) \equiv a = moveTo(o, x, y) \lor \\ at(o, x, y, s) \land \neg\exists x', y' \, (a = moveTo(o, x', y'))$$

may express that the object $o$ is at $(x, y)$ in $do(a, s)$ iff it is moved there in $s$, or already there and not moved away in $s$.

- $\mathcal{D}_{ap}$ is the set of *action precondition axioms*. For each action $a$, it contains an axiom $Poss(a(\vec{x}), s) \equiv \Pi(\vec{x}, s)$, which characterizes the preconditions of $a$. For example, $Poss(moveTo(o, x, y), s) \equiv \neg\exists o' \, (at(o', x, y, s))$ may express that it is possible to move the object $o$ to $(x, y)$ in $s$ iff no other object $o'$ is at $(x, y)$ in $s$.

Golog is an agent programming language that is based on the situation calculus. It allows for constructing complex actions from the primitive actions defined in a basic action theory $AT$, where standard (and not so standard) Algol-like constructs can be used, in particular, (i) action sequences: $p_1; p_2$; (ii) tests: $\phi?$; (iii) nondeterministic action choices: $p_1|p_2$; (iv) nondeterministic choices of action argument: $\pi x \, (p(x))$; and (v) conditionals, while-loops, and procedures.

## 3 Team Golog under Partial Observability

We now introduce the agent programming language TEAMGOLOG, which is a generalization of Golog for programming teams of cooperative agent under partial observability.

Our approach is based on the key concepts of a *synchronization state* and a *communication state*, which allow the agents to passively resp. actively coordinate their behavior, while keeping their belief states, observations, and activities invisible to the other agents. Here, the synchronization state is fully observable by all the agents, but outside their control. The communication state is a multi-dimensional state, containing one dimension for each agent, which is also fully observable by all the agents. But every agent may change its part of the communication state whenever necessary, which encodes its explicit communication to all the other agents.

Since both the synchronization state $S$ and the communication state $C$ are fully observable by all the agents, they can be used to condition and coordinate the behavior of the agents. At the same time, each agent can keep its belief state, observations, and actions invisible to the other agents. We thus realize a maximally distributed acting of the agents. The TEAMGOLOG program of each agent encodes the agent's behavior conditioned on $S$ and $C$, and thus on the current situation. Hence, TEAMGOLOG programs bear close similarity to social laws in *artificial social systems* [Shoham and Tennenholtz, 1995]. The basic idea behind such systems is to formulate a mechanism, called social law, that minimizes the need for both centralized control and online resolution of conflicts.

There are many real-world situations where we encounter such a form of coordination. E.g., the traffic law "right has precedence over left" regulates the order in which cars can

pass a street cross. In most cases, this law is sufficient to make the cars pass the street cross without any further interaction between the car drivers. Only in exceptional cases, such as the one where on each street a car is approaching the cross or when a car has a technical defect, some additional communication between the car drivers is necessary. Similarly, a soccer team can fix in advance the behavior of its team members in certain game situations (such as defense or attack), thus minimizing the explicit communication between the members during the game (which may be observed by the adversary). In these two examples, the synchronization state encodes the situation at the street cross resp. the game situation, while the communication state encodes the explicit communication. The correct behavior of the car drivers resp. soccer players is encoded by traffic laws resp. the strategy fixed by the team in their training units and before the game.

In the rest of this section, we first define a variant of Dec-POMDPs, which underlies the decision-theoretic semantics of TEAMGOLOG programs. We then define the domain theory and the syntax of TEAMGOLOG programs.

### 3.1 Weakly Correlated Dec-POMDPs

We consider the following variant of Dec-POMDPs for $n \geq 2$ agents, which essentially consist of a transition function between global states (where every global state consists of a communication state for each agent and a synchronization state) and a POMDP for each agent and each global state, where every agent can also send a message to the others by changing its communication state. A *weakly correlated Dec-POMDP* $(I, S, (C_i)_{i\in I}, P, (S_i)_{i\in I}, (A_i)_{i\in I}, (O_i)_{i\in I}, (P_i)_{i\in I}, (R_i)_{i\in I})$ consists of a set of $n \geq 2$ *agents* $I = \{1, \ldots, n\}$, a nonempty finite set of *synchronization states* $S$, a nonempty finite set of *communication states* $C_i$ for every agent $i \in I$, a transition function $P: C \times S \rightarrow PD(C \times S)$, which associates with every *global state*, consisting of a *joint communication state* $c \in C = \times_{i\in I} C_i$ and a synchronization state $s \in S$, a probability distribution over $C \times S$, and for every agent $i \in I$: (i) a nonempty finite set of *local states* $S_i$, a nonempty finite set of *actions* $A_i$, (ii) a nonempty finite set of *observations* $O_i$, (iii) a transition function $P_i: C \times S \times S_i \times A_i \rightarrow PD(C_i \times S_i \times O_i)$, which associates with every global state $(c, s) \in (C, S)$, local state $s_i \in S_i$, and action $a_i \in A_i$ a probability distribution over $C_i \times S_i \times O_i$, and (iv) a *reward function* $R_i: C \times S \times S_i \times A_i \rightarrow \mathbf{R}$, which associates with every global state $(c, s) \in C \times S$, local state $s_i \in S_i$, and action $a_i \in A_i$ a *reward* $R_i(c, s, s_i, a_i)$ to agent $i$.

The $q$- and $v$-functions for agent $i \in I$ of a finite-horizon value iteration are defined in Fig. 1 for $n > 0$ and $m \geq 0$, where $P_{c'_i}(\cdot|c, s)$ is the conditioning of $P(\cdot|c, s)$ on $c'_i$, and $c'_{-i}$ denotes $c'$ without $c'_i$. That is, an optimal action of agent $i$ in the global state $(c, s)$ and the local state $s_i$ when there are $n$ steps to go is given by $\mathrm{argmin}_{a_i \in A_i} Q_i^n(c, s, s_i, a_i)$. Notice that these are the standard definitions of $q$- and $v$-functions, adapted to our framework of local and global states.

### 3.2 Domain Theory

TEAMGOLOG programs are interpreted relative to a domain theory, which extends a basic action theory by stochastic actions, reward functions, and utility functions. Formally,

$$
\begin{aligned}
&Q_i^0(c, s, s_i, a_i) = R_i(c, s, s_i, a_i) \\
&Q_i^n(c, s, s_i, a_i) = R_i(c, s, s_i, a_i) + \\
&\quad \sum_{c' \in C} \sum_{s' \in S} \sum_{s'_i \in S_i} \sum_{o_i \in O_i} P_i(c'_i, s'_i, o_i | c, s, s_i, a_i) \cdot \\
&\qquad P_{c'_i}(c'_{-i}, s' | c, s) \cdot V_i^{n-1}(c', s', s'_i) \\
&V_i^m(c, s, s_i) = \min_{a_i \in A_i} Q_i^m(c, s, s_i, a_i)\,,
\end{aligned}
$$

Figure 1: $Q$- and $V$-Functions

a *domain theory* $DT_i = (AT_i, ST_i, OT_i)$ consists of $n \geq 2$ *agents* $I = \{1, \ldots, n\}$, and for each agent $i \in I$: a basic action theory $AT_i$, a *stochastic theory* $ST_i$, and an *optimization theory* $OT_i$, where the latter two are defined below.

The finite nonempty set of primitive actions $A$ is partitioned into nonempty sets of primitive actions $A_1, \ldots, A_n$ of agents $1, \ldots, n$, respectively. We assume a finite nonempty set of observations $O$, which is partitioned into nonempty sets of observations $O_1, \ldots, O_n$ of agents $1, \ldots, n$, respectively.

A *stochastic theory* $ST_i$ for agent $i \in I$ is a set of axioms that define stochastic actions for agent $i$. We represent stochastic actions through a finite set of deterministic actions, as usual [Finzi and Pirri, 2001; Boutilier *et al.*, 2000]. When a stochastic action is executed, then with a certain probability, "nature" executes exactly one of its deterministic actions and produces exactly one possible observation. As underlying decision-theoretic semantics, we assume the weakly correlated Dec-POMDPs of Section 3.1, along with the relational fluents that associate with every situation $s$ a communication state $c_j$ of agent $j \in I$, a synchronization state $z$, and a local state $s_i$ of agent $i$, respectively. The communication and synchronization properties are visible by all the agents, the others are private and hidden. We use the predicate $stochastic(a, s, n, o, \mu)$ to encode that when executing the stochastic action $a$ in the situation $s$, "nature" chooses the deterministic action $n$ producing the observation $o$ with the probability $\mu$. Here, for every stochastic action $a$ and situation $s$, the set of all $(n, o, \mu)$ such that $stochastic(a, s, n, o, \mu)$ is a probability function on the set of all deterministic components $n$ and observations $o$ of $a$ in $s$. We also use the notation $prob(a, s, n, o)$ to denote the probability $\mu$ such that $stochastic(a, s, n, o, \mu)$. We assume that $a$ and all its nature choices $n$ have the same preconditions. A stochastic action $a$ is indirectly represented by providing a *successor state axiom* for every associated nature choice $n$. The stochastic action $a$ is *executable* in a situation $s$ with observation $o$, denoted $Poss(a_o, s)$, iff $prob(a, s, n, o) > 0$ for some $n$. The *optimization theory* $OT_i$ for agent $i \in I$ specifies a reward and a utility function for agent $i$. The former associates with every situation $s$ and action $a$, a reward to agent $i \in I$, denoted $reward(i, a, s)$. The utility function maps every reward and success probability to a real-valued utility $utility(v, pr)$. We assume $utility(v, 1) = v$ and $utility(v, 0) = 0$ for all $v$. An example is $utility(v, pr) = v \cdot pr$. The utility function suitably mediates between the agent reward and the failure of actions due to unsatisfied preconditions.

**Example 3.1** *(Rescue Domain)* We consider a rescue domain where several autonomous mobile agents have to localize some victims in the environment and report their positions

to a remote operator. We assume a team of three heterogeneous agents $a_1$, $a_2$, and $a_3$ endowed with shape recognition (SH), infrared (IF), and $CO_2$ sensors, respectively. A victim position is communicated to the operator once sensed and analyzed by all the three sensing devices. Each agent $a_i$ can execute one of the actions $goTo_i(pos)$, $analyze_i(pos, type_i)$, and $reportToOp_i(pos)$. The action theory $AT_i$ is described by the fluents $at_i(pos, s)$, $analyzed_i(pos, type_i, s)$, and $reported_i(x, s)$, which are accessible only by agent $a_i$. E.g., the successor state axiom for $at_i(pos, s)$ is

$$at_i(pos, do(a, s)) \equiv a = goTo_i(pos) \vee at_i(pos, s) \wedge \neg\exists pos'\,(a = goTo_i(pos'))$$

and the precondition axiom for the action $analyze_i$ is given by $Poss(analyze_i(pos, type_i), s) \equiv at_i(pos, s)$. As for the global state, the communication state is defined by the fluent $cs_i(data, s)$, where $i$ is the agent, and $data$ is the shared info, e.g., $cs_1(atVictim((2,2), IF), s)$ means that $a_1$ detected a victim in position $(2,2)$ through the IF sensor. Other global data are $repVictim(p)$ (victim reported in position $p$) and $noVictim(p)$ (position $p$ was inspected and there is no victim). Some synchronization states are $start(s)$ and $reset(s)$ standing for starting resp. resetting the rescue session. In $ST_i$, we define the stochastic versions of the actions in $AT_i$, e.g., $goToS_i(pos)$ and $analyzeS_i(pos, type_i)$. Each of these can fail resulting in an empty action, e.g.,

$$prob(goToS_i(pos), s, goTo_i(pos), obs(succ)) = 0.9,$$
$$prob(goToS_i(pos), s, nop, obs(fail)) = 0.7\,.$$

In $OT_i$, we provide a high reward for a fully analyzed victim correctly reported to the operator, a low reward for the analysis of a detected victim, and a (distance-dependent) cost is associated with the action $goTo$. Since two agents can obstacle each other when operating in the same location, we penalize the agents analyzing the same victim at the same time. More precisely, we employ the following reward:

$$\begin{aligned} &reward(i, a, s) = r \;=_{def}\; \exists p, t\,(a = analyze_i(p, t) \wedge \\ &\quad (detVictim(p, s) \wedge (\neg conflicts_i(a, s) \wedge r = 50 \vee \\ &\quad conflicts_i(a, s) \wedge r = 10) \vee \neg detVictim(p, s) \wedge r = -10) \vee \\ &\quad a = reportToOp_i(p) \wedge fullyAnalyzed(p, s) \wedge r = 200 \vee \\ &\quad a = goTo_i(p) \wedge \exists p'(at_i(p', s) \wedge r = -dist(p', p)))\,, \end{aligned}$$

where $conflicts_i$ is true if another agent communicates the analysis of the same location in the global state; $detVictim(p, s)$ is true if at least one agent has discovered a victim in $p$, i.e., $cs_i(atVictim(p, t), s)$ for some $i$ and $t$; and $fullyAnalyzed(p, s)$ means that all the analysis has been performed, i.e., $cs_1(atVictim(p, SH), s) \wedge cs_2(atVictim(p, IF), s) \wedge cs_3(atVictim(p, CO_2), s)$. Notice that the action $goTo_i(p)$ has a cost depending on the distance between starting point and destination, hence, in a greedy policy, the agent should go towards the closest non-analyzed victim and analyze it. However, given the penalty on the conflicts, the agents are encouraged to distribute their analysis on different victims taking into account the decisions of the other agents.

### 3.3 Belief States

We next introduce belief states over situations for single agents, and define the semantics of actions in terms of transitions between belief states. A *belief state* $b$ of agent $i \in I$ is a set of pairs $(s, \mu)$ consisting of an ordinary situation $s$ and a real $\mu \in (0, 1]$ such that (i) all $\mu$ sum up to 1, and (ii) all situations $s$ in $b$ are associated with the same joint communication state and the same synchronization state. Informally, every $b$ represents the local belief of agent $i \in I$ expressed as a probability distribution over its local states, along with unique joint communication and synchronization states. The *probability* of a fluent formula $\phi(s)$ (uniform in $s$) in the belief state $b$, denoted $\phi(b)$, is the sum of all $\mu$ such that $\phi(s)$ is true and $(s, \mu) \in b$. In particular, $Poss(a, b)$, where $a$ is an action, is defined as the sum of all $\mu$ such that $Poss(a, s)$ is true and $(s, \mu) \in b$, and $reward(i, a, b)$ is defined in a similar way.

Given a deterministic action $a$ and a belief state $b$ of agent $i \in I$, the *successor belief state after executing $a$ in $b$*, denoted $do(a, b)$, is the belief state $b' = \{(do(a, s), \mu/Poss(a, b)) \mid (s, \mu) \in b,\ Poss(a, s)\}$. Furthermore, given a stochastic action $a$, an observation $o$ of $a$, and a belief state $b$ of agent $i \in I$, the *successor belief state after executing $a$ in $b$ and observing $o$*, denoted $do(a_o, b)$, is the belief state $b'$, where $b'$ is obtained from all pairs $(do(n, s), \mu \cdot \mu')$ such that $(s, \mu) \in b$, $Poss(a, s)$, and $\mu' = prob(a, s, n, o) > 0$ by normalizing the probabilities to sum up to 1.

The probability of making the observation $o$ after executing the stochastic action $a$ in the local belief state $b$ of agent $i \in I$, denoted $prob(a, b, o)$, is defined as the sum of all $\mu \cdot \mu'$ such that $(s, \mu) \in b$ and $\mu' = prob(a, s, n, o) > 0$.

**Example 3.2** *(Rescue Domain cont'd)* Suppose that agent $a_1$ is aware of its initial situation, and thus has the initial belief state $\{(S_0, 1)\}$. After executing the stochastic action $goToS_1(1, 1)$ and observing its success $obs(succ)$, the belief state of $a_1$ then changes to $\{(S_0, 0.1), (do(goTo_1(1, 1), S_0), 0.9)\}$ (here, $prob(goToS_1(pos), s, goTo_1(pos), obs(succ)) = 0.9$, and $goToS_1(pos)$ is always executable).

### 3.4 Syntax

Given the actions specified by a domain theory $DT_i$, a *program* $p$ in TEAMGOLOG for agent $i \in I$ has one of the following forms (where $\phi$ is a condition, $p, p_1, p_2$ are programs, and $a, a_1, \ldots, a_n$ are actions of agent $i$):

1. *Deterministic or stochastic action:* $a$. Do $a$.
2. *Nondeterministic action choice:* **choice**$(i{:}\, a_1 | \cdots | a_n)$. Do an optimal action among $a_1, \ldots, a_n$.
3. *Test action*: $\phi?$. Test $\phi$ in the current situation.
4. *Action sequence*: $p_1; p_2$. Do $p_1$ followed by $p_2$.
5. *Nondeterministic choice of two programs:* $(p_1 \mid p_2)$. Do $p_1$ or $p_2$.
6. *Nondeterministic choice of an argument:* $\pi x\,(p(x))$. Do any $p(x)$.
7. *Nondeterministic iteration*: $p^\star$. Do $p$ zero or more times.
8. *Conditional*: **if** $\phi$ **then** $p_1$ **else** $p_2$.
9. *While-loop*: **while** $\phi$ **do** $p$.
10. *Procedures, including recursion.*

**Example 3.3** *(Rescue Domain cont'd)* The following code represents an incomplete procedure $explore_i$ of agent $i$:

$$\begin{array}{l}\textbf{proc}(explore_i,\\ \quad \pi x\,(goToS_i(x);\\ \quad\quad \textbf{if } obs(succ) \textbf{ then } [analyzeS_i(x, type_i);\\ \quad\quad\quad \textbf{if } obs(succ) \wedge fullyAnalyzed(x) \textbf{ then}\\ \quad\quad\quad\quad reportToOp_i(repVictim(x))]);\\ \quad explore_i)\,.\end{array}$$

Here, agent $i$ first has to decide where to go. Once the position is reached, agent $i$ analyzes the current location deploying one of its sensing devices. If a victim is detected, then the position of the victim is communicated to the operator.

# 4 TEAMGOLOG Interpreter

In this section, we first specify the decision-theoretic semantics of TEAMGOLOG programs in terms of an interpreter. We then provide theoretical results about the interpreter.

## 4.1 Formal Specification

We now define the formal semantics of a TEAMGOLOG program $p$ for agent $i \in I$ relative to a domain theory $DT$. We associate with every TEAMGOLOG program $p$, belief state $b$, and horizon $H \geq 0$, an optimal $H$-step policy $\pi$ along with its expected utility $U$ to agent $i \in I$. Intuitively, this $H$-step policy $\pi$ is obtained from the $H$-horizon part of $p$ by replacing every nondeterministic action choice by an optimal action.

Formally, given a TEAMGOLOG program $p$ for agent $i \in I$ relative to a domain theory $DT$, a horizon $H \geq 0$, and a start belief state $b$ of agent $i$, we say that $\pi$ is an *$H$-step policy* of $p$ in $b$ with *expected $H$-step utility* $U$ to agent $i$ iff $DT \models G(p, b, H, \pi, \langle v, pr\rangle)$ and $U = utility(v, pr)$, where the macro $G(p, b, h, \pi, \langle v, pr\rangle)$ is defined by induction on the different constructs of TEAMGOLOG. The definition of $G$ for some of the constructs is given as follows (the complete definition is given in the full version of this paper):

- Null program ($p = nil$) or zero horizon ($h = 0$):

$$G(p, b, h, \pi, \langle v, pr\rangle) \;=_{def}\; \pi = stop \wedge \langle v, pr\rangle = \langle 0, 1\rangle\,.$$

Intuitively, $p$ ends when it is null or at the horizon end.

- Stochastic first program action with observation and $h > 0$:

$$\begin{array}{l}G([a_o\,;p'], b, h, \pi, \langle v, pr\rangle) \;=_{def}\; (Poss(a_o, b) = 0 \wedge\\ \quad \pi = stop \wedge \langle v, pr\rangle = \langle 0, 1\rangle) \;\vee\; (Poss(a_o, b) > 0 \wedge\\ \quad \exists\,(\bigwedge_{q=1}^{l} G(p', do(a_o, b), h{-}1, \pi_q, \langle v_q, pr_q\rangle) \wedge\\ \quad\quad \pi = a_o\,;\textbf{for } q = 1 \textbf{ to } l \textbf{ do if } o_q \textbf{ then } \pi_q \wedge\\ \quad\quad v = reward(i, a_o, b) + \sum_{q=1}^{l} v_q \cdot prob(a_o, b, o_q) \wedge\\ \quad\quad pr = Poss(a_o, b) \cdot \sum_{q=1}^{l} pr_q \cdot prob(a_o, b, o_q)))\,.\end{array}$$

Here, $\exists(F)$ is obtained from $F$ by existentially quantifying all free variables in $F$. Moreover, $o_1, \ldots, o_l$ are the different pairs of a joint communication state and a synchronization state that are compatible with $a_o$, and $prob(a_o, b, o_q)$ is the probability of arriving in such $o_q$ after executing $a_o$ in $b$. Informally, suppose $p = [a_o\,;p']$, where $a_o$ is a stochastic action with observation. If $a_o$ is not executable in $b$, then $p$ has only the policy $\pi = stop$ along with the expected reward $v = 0$ and the success probability $pr = 0$. Otherwise, the optimal execution of $[a_o\,;p']$ in $b$ depends on that one of $p$ in $do(a_o, b)$.

- Stochastic first program action and $h > 0$:

$$\begin{array}{l}G([a\,;p'], b, h, \pi, \langle v, pr\rangle) \;=_{def}\\ \quad \exists\,(\bigwedge_{q=1}^{l} G([a_{o_q}\,;p'], b, h, a_{o_q}\,;\pi_q, \langle v_q, pr_q\rangle) \wedge\\ \quad\quad \pi = a_{o_q}\,;\textbf{for } q = 1 \textbf{ to } l \textbf{ do if } o_q \textbf{ then } \pi_q \wedge\\ \quad\quad v = \sum_{q=1}^{l} v_q \cdot prob(a, b, o_q) \wedge\\ \quad\quad pr = \sum_{q=1}^{l} pr_q \cdot prob(a, b, o_q))\,.\end{array}$$

Here, $o_1, \ldots, o_l$ are the possible observations of the stochastic action $a$. The generated policy is a conditional plan in which every such observation $o_q$ is considered.

- Nondeterministic first program action and $h > 0$:

$$\begin{array}{l}G([\textbf{choice}(i{:}\,a_1 \,|\, \cdots \,|\, a_n)\,;p'], b, h, \pi, \langle v, pr\rangle) \;=_{def}\\ \quad \exists\,(\bigwedge_{q=1}^{n} G([a_q\,;p'], b, h, a_q\,;\pi_q, \langle v_q, pr_q\rangle) \wedge\\ \quad\quad k = \mathrm{argmax}_{q\in\{1,\ldots,n\}}\; utility(v_q, pr_q) \wedge\\ \quad\quad \pi = \pi_k \wedge v = v_k \wedge pr = pr_k)\,.\end{array}$$

## 4.2 Theoretical Results

The following result shows that the TEAMGOLOG interpreter indeed generates an optimal $H$-step policy $\pi$ along with its expected utility $U$ to agent $i \in I$ for a given TEAMGOLOG program $p$, belief state $b$, and horizon $H \geq 0$.

**Theorem 4.1** *Let $p$ be a* TEAMGOLOG *program for agent $i \in I$ w.r.t. a domain theory $DT_i$, let $b$ be a belief state, and let $H \geq 0$ be a horizon. Then, the optimal $H$-step policy $\pi$ of $p$ in $b$ along with its expected utility $U$ to agent $i \in I$ is given by $DT_i \models G(p, b, H, \pi, \langle v, pr\rangle)$ and $U = utility(v, pr)$.*

The next result gives an upper bound for the number of leaves in the evaluation tree, which is polynomial when the horizon is bounded by a constant. Here, $n$ is the maximum among the maximum number of actions in nondeterministic action choices, the maximum number of observations after actions, the maximum number of arguments in nondeterministic choices of an argument, and the number of pairs consisting of a synchronization state and a communication state.

**Theorem 4.2** *Let $p$ be a* TEAMGOLOG *program for agent $i \in I$ w.r.t. a domain theory $DT_i$, let $b$ be a belief state, and let $H \geq 0$ be a horizon. Then, computing the $H$-step policy $\pi$ of $p$ in $b$ along with its expected utility $U$ to agent $i \in I$ via $G$ generates $O(n^{3H})$ leaves in the evaluation tree.*

# 5 Rescue Scenario

Consider the rescue scenario in Fig. 2. We assume that three victims have already been detected in the environment, but not completely analyzed: in position $(3, 7)$, the presence of Alice was detected by $a_1$ through the SH sensor; in position $(7, 7)$, agent $a_2$ discovered Bob through IF, and $a_3$ analyzed him through the $CO_2$ sensor; finally, in position $(4, 2)$, victim Carol was detected by $a_2$ with IF. We assume that this information is available in the global state, that is, the properties $cs_1(atVictim((3, 7), SH), s)$, $cs_3(atVictim((7, 7), IF), s)$, $cs_2(atVictim((7, 7), CO_2), s)$, and $cs_2(atVictim((4, 2), IF), s)$ hold in the communication state of the agents. As for the local state, we assume the belief states $b_1 = \{(s_{1,1}, 0.8), (s_{1,2}, 0.2)\}$, $b_2 = \{(s_2, 1)\}$, and $b_3 = \{(s_3, 1)\}$, with $at_1(3, 6, s_{1,1})$, $at_1(3, 5, s_{1,2})$, $at_2(7, 7, s_2)$, and $at_3(3, 7, s_3)$.

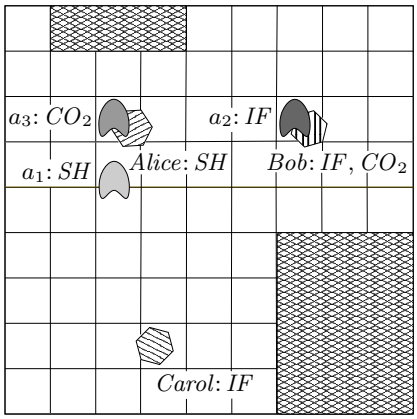


Figure 2: Rescue Scenario

Given this situation, the task of the team of agents is to fully analyze the discovered victims and report their positions to the operator once the victim analysis is completed. This task can be encoded by the following procedure:

$$
\begin{array}{l}
\textbf{proc}(explore_i, \\
\quad \pi x \in \{(3,7),(7,7),(4,2)\}\ (goToS_i(x); \\
\quad\quad \textbf{if}\ obs(succ)\ \textbf{then}\ [analyzeS_i(x, type_i); \\
\quad\quad\quad \textbf{if}\ obs(succ) \wedge fullyAnalyzed(x)\ \textbf{then} \\
\quad\quad\quad\quad reportToOp_i(repVictim(x))]); \\
\quad explore_i)\ ,
\end{array}
$$

where $type_1 = SH$, $type_2 = IF$, and $type_3 = CO_2$. Every agent $a_i$ with $i \in \{1,2,3\}$ has to separately compile the procedure $explore_i$ using its global and local information. Assuming the horizon $H = 5$ and the initial belief state $b_i$, the optimal 5-step policy $\pi_i$ for agent $a_i$, produced by the TEAMGOLOG interpreter, is such that $DT_i \models G([explore_i;\ nil], b_i, 5, \pi_i, \langle v_i, pr_i \rangle)$. Here, $\pi_1$, $\pi_2$, and $\pi_3$ are complex conditional plans branching over all possible observations and global states. E.g., the beginning of $\pi_1$ is as follows:

$$
\begin{array}{l}
goToS_1(7,7); \\
\textbf{if}\ obs(succ)\ \textbf{then}\ [analyzeS_1((7,7), SH); \\
\quad \textbf{if}\ obs(succ) \wedge fullyAnalyzed(7,7)\ \textbf{then} \\
\quad\quad reportToOp_1(repVictim(7,7))]; \\
goToS_1(4,2); \ldots
\end{array}
$$

This scenario has been realized in an abstract simulator. The simulator captures the key features of the environment and allows to execute agent actions computing associated rewards. A greedy control strategy has been devised as a comparison with the derived policies $\pi_i$. In the greedy strategy, at each time step, each agent searches for the best victim to analyze, based on the current distance to the victim. Whenever a victim has been completely analyzed, the agent reports the victim state to the operator. Both the greedy strategy and the policies $\pi_i$ were able to correctly report all victims to the operator, however, the policies $\pi_i$ revealed to be superior with respect to the greedy strategy gaining more reward. The greedy gain is around 30% of the policies $\pi_i$, because these were able to minimize conflicts among team members. Notice that, information available to the two strategies are the same, and that the better performance of the policies $\pi_i$ are achieved using planning over the global and local states.

## 6 Summary and Outlook

We have presented the agent programming language TEAMGOLOG for programming a team of cooperative agents under partial observability. The approach is based on a decision-theoretic semantics and the key concepts of a synchronization state and a communication state, which allow the agents to passively resp. actively coordinate their behavior, while keeping their belief states, observations, and activities invisible to the other agents. We have also provided experimental results from the rescue simulation domain.

An interesting topic for future research is to develop an adaptive version of this approach. Another topic for future work is to explore whether the approach can be generalized to multi-agent systems with competitive agents.

**Acknowledgments.** This work has been partially supported by the Austrian Science Fund Project P18146-N04 and by a Heisenberg Professorship of the German Research Foundation (DFG). We thank the reviewers for their constructive comments, which helped to improve this work.

## References

[Boutilier *et al.*, 2000] C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. AAAI-2000*, pp. 355–362, 2000.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proc. IJCAI-2001*, pp. 690–700, 2001.

[Ferrein *et al.*, 2005] A. Ferrein, C. Fritz, and G. Lakemeyer. Using Golog for deliberation and team coordination in robotic soccer. *Künstliche Intelligenz*, 1:24–43, 2005.

[Finzi and Lukasiewicz, 2006] A. Finzi and T. Lukasiewicz. Game-theoretic agent programming in Golog under partial observability. In *Proc. KI-2006*, *LNCS/LNAI*, 2006.

[Finzi and Pirri, 2001] A. Finzi and F. Pirri. Combining probabilities, failures and safety in robot control. In *Proc. IJCAI-2001*, pp. 1331–1336, 2001.

[Goldman and Zilberstein, 2004] C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res.*, 22:143–174, 2004.

[Guestrin *et al.*, 2003] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational MDPs. In *Proc. IJCAI-2003*, pp. 1003–1010, 2003.

[McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In *Machine Intelligence*, volume 4, pp. 463–502. 1969.

[Nair *et al.*, 2003] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proc. IJCAI-2003*, pp. 705–711, 2003.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. 2001.

[Shoham and Tennenholtz, 1995] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artif. Intell.*, 73(1–2):231–252, 1995.

[Yoon *et al.*, 2002] S. W. Yoon, A. Fern, and R. Givan. Inductive policy selection for first-order MDPs. In *Proc. UAI-2002*, pp. 568–576, 2002.