Game-Theoretic Agent Programming in Golog under Partial Observability

Alberto Finzi[1, 2] and Thomas Lukasiewicz[2, 1]

[1] Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria

[2] Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Rome, Italy
{finzi, lukasiewicz}@dis.uniroma1.it

Abstract. We present the agent programming language POGTGolog, which integrates explicit agent programming in Golog with game-theoretic multi-agent planning in partially observable stochastic games. It deals with the case of one team of cooperative agents under partial observability, where the agents may have different initial belief states and not necessarily the same rewards. POGTGolog allows for specifying a partial control program in a high-level logical language, which is then completed by an interpreter in an optimal way. To this end, we define a formal semantics of POGTGolog programs in terms of Nash equilibria, and we specify a POGTGolog interpreter that computes one of these Nash equilibria. We illustrate the usefulness of POGTGolog along a rugby scenario.

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 [3], which integrates explicit agent programming in Golog [16] with decision-theoretic planning in (fully observable) Markov decision processes (MDPs) [15]. 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 [2,19,8]), 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).

As a serious drawback, 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, which may compete against each other, or which may also cooperate with each other. For example, in *robotic soccer*, we have two competing teams of agents, where each team consists of cooperating agents. Here, the optimal actions of one agent generally depend on the actions of all the other ("enemy" and "friend") agents. In particular, there is a bidirected dependence between the actions of two different agents, which generally makes it inappropriate to model enemies and friends of the agent that we control simply as a part of "nature". As an example for an important cooperative domain, 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.

This is the motivation behind GTGolog [5], which is a generalization of DTGolog that integrates agent programming in Golog with game-theoretic multi-agent planning in (fully observable) stochastic games [14], also called Markov games [17,11].

Example 1.1. Consider a rugby player $\boldsymbol{a}_1$, who is deciding his next $n>0$ moves and wants to cooperate with a team mate $\boldsymbol{a}_2$. He has to deliberate about if and when it is worth to pass the ball. His options can be encoded by the following GTGolog program:

proc $step(n)$
if $(haveBall(\boldsymbol{a}_1) \wedge n>0)$ **then**
  $\pi x\,(\pi y\,(\textbf{choice}(\boldsymbol{a}_1 : moveTo(x) \mid passTo(\boldsymbol{a}_2)) \parallel$
    $\textbf{choice}(\boldsymbol{a}_2 : moveTo(y) \mid receive(\boldsymbol{a}_1))));$
$step(n-1)$
end.

This program encodes that while $\boldsymbol{a}_1$ is the ball owner and $n>0$, the two agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ perform a parallel action choice in which $\boldsymbol{a}_1$ (resp., $\boldsymbol{a}_2$) can either go somewhere or pass (resp., receive) the ball. Here, the preconditions and effects of the actions are to be formally specified in a suitable action theory. Given this high-level program and the action theory for $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, the program interpreter then fills in the best moves for $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, reasoning about all the possible interactions between the two agents.

Another crucial aspect of real-world environments, however, 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. However, both DT- and GTGolog assume full observability, and have not been generalized to the partially observable case so far.

In this paper, we try to fill this gap. We present the agent programming language POGTGolog, which extends GTGolog and thus also DTGolog by partial observability. The main contributions of this paper can be summarized as follows:

- We present the agent programming language POGTGolog, which integrates explicit agent programming in Golog with game-theoretic multi-agent planning in partially observable stochastic games [9]. POGTGolog allows for modeling one team of cooperative agents under partial observability, where the agents may have different initial belief states and not necessarily the same rewards (and thus in some sense the team does not necessarily have to be homogeneous).
- POGTGolog allows for specifying a partial control program in a high-level language, which is then completed in an optimal way. To this end, we associate with every POGTGolog program a set of (finite-horizon) policies, which are possible (finite-horizon) instantiations of the program where missing details are filled in. We then define a semantics of a POGTGolog program in terms of Nash equilibria, which are optimal policies (that is, optimal instantiations) of the program.
- We define a POGTGolog interpreter and show that it computes a Nash equilibrium of POGTGolog programs. We also prove that POGTGolog programs can represent partially observable stochastic games, and that the POGTGolog interpreter can be used to compute one of their (finite-horizon) Nash equilibria. Furthermore, we illustrate the usefulness of the POGTGolog approach along a rugby scenario.

2 Preliminaries

In this section, we first recall the main concepts of the situation calculus and of the agent programming language Golog; for further details see especially [16]. We then recall the basics of normal form games and of partially observable stochastic games (POSGs).

2.1 The Situation Calculus

The situation calculus [12,16] is a first-order language for representing dynamically changing worlds. Its main ingredients are *actions*, *situations*, and *fluents*. An *action* is a first-order term of the form $a(u_1, \ldots, u_n)$, where the function symbol a is its *name* and the u_i's are its *arguments*. All changes to the world are the result of actions. For example, the action $moveTo(r, x, y)$ may stand for moving the 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 do is a distinguished binary function symbol, 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, the situation $do(moveTo(r, 1, 2), do(moveTo(r, 3, 4), S_0))$ stands for executing $moveTo(r, 1, 2)$ after executing $moveTo(r, 3, 4)$ in the initial situation S_0. We write $Poss(a, s)$, where $Poss$ is a distinguished binary predicate symbol, to denote that the action a is possible to execute in the situation s. A *(relational) 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 represented by a *basic action theory* $AT = (\Sigma, \mathcal{D}_{una}, \mathcal{D}_{S_0}, \mathcal{D}_{ssa}, \mathcal{D}_{ap})$, where:

- Σ is the set of (domain-independent) foundational axioms for situations [16].
- $\mathcal{D}_{una}$ is the set of unique names axioms for actions, which express that different actions are interpreted in a different way.
- $\mathcal{D}_{S_0}$ is a set of first-order formulas describing the initial state of the domain (represented by S_0). For example, $at(r, 1, 2, S_0) \wedge at(r', 3, 4, S_0)$ may express that the agents r and r' are initially at the positions $(1, 2)$ and $(3, 4)$, respectively.
- $\mathcal{D}_{ssa}$ is the set of *successor state axioms* [16]. For each fluent $F(\boldsymbol{x}, s)$, it contains an axiom of the form $F(\boldsymbol{x}, do(a, s)) \equiv \Phi_F(\boldsymbol{x}, a, s)$, where $\Phi_F(\boldsymbol{x}, a, s)$ is a formula with free variables among $\boldsymbol{x}, a, s$. 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, the axiom $at(r, x, y, do(a, s)) \equiv a = moveTo(r, x, y) \vee (at(r, x, y, s) \wedge \neg\exists x', y' \, (a = moveTo(r, x', y')))$ may express that the agent r is at the position (x, y) in the situation $do(a, s)$ iff either r moves to (x, y) in the situation s, or r is already at the position (x, y) and does not move away in s.
- $\mathcal{D}_{ap}$ is the set of *action precondition axioms*. For each action a, it contains an axiom of the form $Poss(a(\boldsymbol{x}), s) \equiv \Pi(\boldsymbol{x}, s)$, which characterizes the preconditions of the action a. For example, $Poss(moveTo(r, x, y), s) \equiv \neg\exists r' \, at(r', x, y, s)$ may express that it is possible to move the agent r to the position (x, y) in the situation s iff no other agent r' is at (x, y) in s.

We use the concurrent version of the situation calculus [16], which is an extension of the standard situation calculus by concurrent actions. A *concurrent action* c is a set of standard actions, which are concurrently executed when c is executed.

2.2 Golog

Golog is an agent programming language that is based on the situation calculus. It allows for constructing *programs* from primitive actions that are defined in a basic action theory AT, where standard (and not so standard) Algol-like control constructs can be used. More precisely, *programs* p in Golog have one of the following forms (where c is a primitive action, ϕ is a *condition*, which is obtained from a situation calculus formula over fluents by suppressing all situation arguments, $p, p_1, p_2, \ldots, p_n$ are programs, $P_1, \ldots, P_n$ are procedure names, and $x, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are arguments):

1. *Primitive action*: c. Do c.
2. *Test action*: $\phi?$. Test the truth of ϕ in the current situation.
3. *Sequence*: $[p_1; p_2]$. Do p_1 followed by p_2.
4. *Nondeterministic choice of two programs*: $(p_1 \mid p_2)$. Do either p_1 or p_2.
5. *Nondeterministic choice of program argument*: $\pi x \, (p(x))$. Do any $p(x)$.
6. *Nondeterministic iteration*: $p^\star$. Do p zero or more times.
7. *Conditional*: **if** ϕ **then** p_1 **else** p_2. If ϕ is true, then do p_1 else do p_2.

8. *While-loop*: **while** ϕ **do** p. While ϕ is true in the current situation, do p.
9. *Procedures*: **proc** $P_1(\boldsymbol{x}_1)\ p_1$ **end**; ...; **proc** $P_n(\boldsymbol{x}_n)\ p_n$ **end**; p.

For example, the Golog program **while** $\neg at(r,1,2)$ **do** $\pi x, y\,(moveTo(r,x,y))$ stands for "while the agent r is not at the position $(1,2)$, move r to a nondeterministically chosen position (x,y)". Golog has a declarative formal semantics, which is defined in the situation calculus. Given a Golog program p, its execution is represented by a situation calculus formula $Do(p,s,s')$, which encodes that the situation s' can be reached by executing the program p in the situation s.

2.3 Normal Form Games

Normal form games from classical game theory [18] describe the possible actions of $n \geq 2$ agents and the rewards that the agents receive when they simultaneously execute one action each. For example, in *two-finger Morra*, two players E and O simultaneously show one or two fingers. Let f be the total numbers of fingers shown. If f is odd, then O gets f dollars from E, and if f is even, then E gets f dollars from O. More formally, a *normal form game* $G = (I, (A_i)_{i \in I}, (R_i)_{i \in I})$ consists of a set of *agents* $I = \{1, \ldots, n\}$ with $n \geq 2$, a nonempty finite set of *actions* A_i for each agent $i \in I$, and a *reward function* $R_i \colon A \to \mathbf{R}$ for each agent $i \in I$, which associates with every *joint action* $a \in A = \bigtimes_{i \in I} A_i$ a *reward* $R_i(a)$ to agent i. If $n = 2$, then G is called a *two-player normal form game* (or simply *matrix game*). If additionally $R_1 = -R_2$, then G is a *zero-sum matrix game*; we then often omit R_2 and abbreviate R_1 by R.

The behavior of the agents in a normal form game is expressed through the notions of pure and mixed strategies, which specify which of its actions an agent should execute and which of its actions an agent should execute with which probability, respectively. For example, in two-finger Morra, a pure strategy for player E (or O) is to show two fingers, and a mixed strategy for player E (or O) is to show one finger with the probability $7/12$ and two fingers with the probability $5/12$. Formally, a *pure strategy* for agent $i \in I$ is any action $a_i \in A_i$. A *pure strategy profile* is any joint action $a \in A$. If the agents play a, then the *reward* to agent $i \in I$ is given by $R_i(a)$. A *mixed strategy* for agent $i \in I$ is any probability distribution π_i over its set of actions A_i. A *mixed strategy profile* $\pi = (\pi_i)_{i \in I}$ consists of a mixed strategy π_i for each agent $i \in I$. If the agents play π, then the *expected reward* to agent $i \in I$, denoted $\mathbf{E}[R_i(a) \mid \pi]$ (or $R_i(\pi)$), is defined as $\sum_{a=(a_j)_{j \in I} \in A} R_i(a) \cdot \Pi_{j \in I} \pi_j(a_j)$.

Towards optimal behavior of the agents in a normal form game, we are especially interested in mixed strategy profiles π, called Nash equilibria, where no agent has the incentive to deviate from its part, once the other agents play their parts. Formally, given a normal form game $G = (I, (A_i)_{i \in I}, (R_i)_{i \in I})$, a mixed strategy profile $\pi = (\pi_i)_{i \in I}$ is a *Nash equilibrium* of G iff for every agent $i \in I$, it holds that $R_i(\pi_i' \circ \pi_{-i}) \leq R_i(\pi)$ for every mixed strategy π_i', where $\pi_i' \circ \pi_{-i}$ is obtained from π by replacing π_i by π_i'. For example, in two-finger Morra, the mixed strategy profile where each player shows one finger resp. two fingers with the probability $7/12$ resp. $5/12$ is a Nash equilibrium. Every normal form game G has at least one Nash equilibrium among its mixed (but not necessarily pure) strategy profiles, and many have multiple Nash equilibria. In the

two-player case, they can be computed by linear complementary programming and linear programming in the general and the zero-sum case, respectively. A *Nash selection function* f associates with every normal form game G a unique Nash equilibrium $f(G)$. The expected reward to agent $i \in I$ under $f(G)$ is denoted by $v_f^i(G)$.

2.4 Partially Observable Stochastic Games

Partially observable stochastic games [9] generalize normal form games, partially observable Markov decision processes (POMDPs) [10], and decentralized POMDPs [7,13]. A partially observable stochastic game consists of a set of states S, a normal form game for each state $s \in S$, a set of joint observations of the agents O, and a transition function that associates with every state $s \in S$ and joint action of the agents $a \in A$ a probability distribution on all combinations of next states $s' \in S$ and joint observations $o \in O$. Formally, a *partially observable stochastic game (POSG)* $G = (I, S, (A_i)_{i \in I}, (O_i)_{i \in I}, P, (R_i)_{i \in I})$ consists of a set of *agents* $I = \{1, \ldots, n\}$, $n \geq 2$, a nonempty finite set of *states* S, two nonempty finite sets of *actions* A_i and *observations* O_i for each $i \in I$, a transition function $P \colon S \times A \to PD(S \times O)$, which associates with every state $s \in S$ and joint action $a \in A = \bigtimes_{i \in I} A_i$ a probability distribution over $S \times O$, where $O = \bigtimes_{i \in I} O_i$, and a *reward function* $R_i \colon S \times A \to \mathbf{R}$ for each $i \in I$, which associates with every state $s \in S$ and joint action $a \in A$ a *reward* $R_i(s, a)$ to i.

Since the actual state $s \in S$ of the POSG G is not fully observable, every agent $i \in I$ has a belief state b_i that associates with every state $s \in S$ the belief of agent i about s being the actual state. A *belief state* $b = (b_i)_{i \in I}$ of G consists of a probability function b_i over S for each agent $i \in I$. The POSG G then defines probabilistic transitions between belief states as follows. The new belief state $b^{a,o} = (b_i^{a,o})_{i \in I}$ after executing the joint action $a \in A$ in $b = (b_i)_{i \in I}$ and jointly observing $o \in O$ is given by $b_i^{a,o}(s') = \sum_{s \in S} P(s', o \mid s, a) \cdot b_i(s) \;/\; P_b(b_i^{a,o} \mid b_i, a)$, where $P_b(b_i^{a,o} \mid b_i, a) = \sum_{s' \in S} \sum_{s \in S} P(s', o | s, a) \cdot b_i(s)$ is the probability of observing o after executing a in b_i.

The notions of finite-horizon pure (resp., mixed) policies and their rewards (resp., expected rewards) can now be defined as usual using the above probabilistic transitions between belief states. Informally, given a finite horizon $H \geq 0$, a pure (resp., mixed) time-dependent policy associates with every belief state b of G and number of steps to go $h \in \{0, \ldots, H\}$ a pure (resp., mixed) normal form game strategy.

Finally, the notion of a finite-horizon Nash equilibrium for a POSG G is then defined as follows. A policy π is a *Nash equilibrium* of G under a belief state b iff for every agent $i \in I$, it holds that $G_i(H, b, \pi_i' \circ \pi_{-i}) \leq G_i(H, b, \pi_i \circ \pi_{-i})$ for all policies π_i', where $G_i(H, b, \alpha)$ denotes the *H-step reward* to agent $i \in I$ under an initial belief state $b = (b_i)_{i \in I}$ and the policy α. A policy π is a *Nash equilibrium* of G iff it is a Nash equilibrium of G under every belief state b.

3 Partially Observable GTGolog (POGTGolog)

In this section, we present the agent programming language POGTGolog, which is a generalization of GTGolog [5] that allows for partial observability. We first describe the domain theory and the syntax and semantics of POGTGolog programs.

We focus on the case of one team of cooperative agents under partial observability, where the agents may have different initial belief states and not necessarily the same rewards (and so may also be heterogeneous). We assume that (i) each agent knows the initial local belief state of every other agent, and (ii) after each action execution, each agent can observe the actions of every other agent and receives their local observations.

3.1 Domain Theory

POGTGolog programs are interpreted relative to a domain theory, which extends a basic action theory by stochastic actions, reward functions, and utility functions. Formally, in addition to a basic action theory AT, a *domain theory* $DT = (AT, ST, OT)$ consists of a *stochastic theory* ST and an *optimization theory* OT, which are both defined below.

We assume a team $I = \{1, \ldots, n\}$ consisting of $n \geq 2$ cooperative agents $1, \ldots, n$. 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. A *single-agent action* of agent $i \in I$ (resp., *multi-agent action*) is any concurrent action over A_i (resp., A). 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 *single-agent observation* of agent $i \in I$ is any $o_i \in O_i$. A *multi-agent observation* is any $o \in \bigtimes_{i \in I} O_i$.

A *stochastic theory* ST is a set of axioms that define stochastic actions. We represent stochastic actions through a finite set of deterministic actions, as usual [6,3]. 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. We use the predicate $stochastic(c, s, n, o, \mu)$ to encode that when executing the stochastic action c in the situation s, "nature" chooses the deterministic action n producing the observation o with the probability μ. Here, for every stochastic action c and situation s, the set of all (n, o, μ) such that $stochastic(c, s, n, o, \mu)$ is a probability function on the set of all deterministic components n and observations o of c in s. We also use the notation $prob(c, s, n, o)$ to denote the probability μ such that $stochastic(c, s, n, o, \mu)$. We assume that c and all its nature choices n have the same preconditions. A stochastic action c is indirectly represented by providing a *successor state axiom* for every associated nature choice n. The stochastic action c is *executable* in a situation s with observation o, denoted $Poss(c_o, s)$, iff $prob(c, s, n, o) > 0$ for some n.

The *optimization theory* OT specifies a reward and a utility function. The former associates with every situation s and multi-agent action c, a reward to every agent $i \in I$, denoted $reward(i, c, s)$. 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 (Rugby Domain). Consider the following rugby domain, which is inspired by the soccer domain in [11]. The rugby field consists of 22 rectangles, which are divided into a 4×5 grid of 20 squares and two goal rectangles (see Fig. 1). We assume a team of two agents $\boldsymbol{a} = \{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ against a (static) team of two agents $\boldsymbol{o} = \{\boldsymbol{o}_1, \boldsymbol{o}_2\}$, where $\boldsymbol{a}_1$ and $\boldsymbol{o}_1$ are the *captains* of $\boldsymbol{a}$ and $\boldsymbol{o}$, respectively. Each agent occupies a square and is able to do one of the following actions on each turn: N, S, E, W, $stand$,

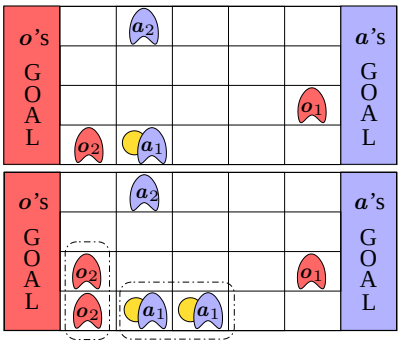

Fig. 1. Rugby Domain: Initial belief states of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively

$passTo(\alpha)$, and $receive$ (move up, move down, move right, move left, no move, pass, and receive the ball, respectively). The ball is represented by an oval and also occupies a square. An agent is a *ball owner* iff it occupies the same square as the ball. The ball follows the moves of the ball owner, and we have a goal when the ball owner steps into the adversary goal. An agent can also pass the ball to another agent of the same team, but this is possible only if the receiving agent is not closer to the opposing end of the field than the ball, otherwise, an offside fault is called by the referee, and the ball possession goes to the captain of the opposing team. When the ball owner goes into the square occupied by the other agent, if the other agent stands, possession of ball changes. Thus, a good defensive maneuver is to stand where the other agent wants to go.

We define the domain theory $DT = (AT, ST, OT)$ as follows. Concerning the basic action theory AT, we assume the deterministic action $move(\alpha, m)$ (encoding that agent α executes m), where $\alpha \in \boldsymbol{a} \cup \boldsymbol{o}$, $m \in \{N, S, E, W, stand, passTo(\alpha'), receive\}$, and α' is a team mate of α, and the fluents $at(\alpha, x, y, s)$ (encoding that agent α is at position (x, y) in situation s) and $haveBall(\alpha, s)$ (encoding that agent α has the ball in situation s). They are defined by the following successor state axioms:

$$
\begin{aligned}
&at(\alpha, x, y, do(c, s)) \equiv \exists x', y' \, (at(\alpha, x', y', s) \; \wedge \exists m \, (move(\alpha, m) \in c \wedge \\
&\quad ((m = stand \vee m = receive \vee \exists \beta \, (m = passTo(\beta))) \wedge x = x' \wedge y = y') \vee \\
&\quad (m = N \wedge x = x' \wedge y = y'+1) \vee (m = S \wedge x = x' \wedge y = y'-1) \vee \\
&\quad (m = E \wedge x = x'+1 \wedge y = y') \vee (m = W \wedge x = x'-1 \wedge y = y'))) \, ; \\
&haveBall(\alpha, do(c, s)) \equiv \exists \beta \, (haveBall(\beta, s) \wedge (\alpha = \beta \wedge \neg \exists \beta' (cngBall(\beta', c, s) \vee \\
&\quad rcvBall(\beta', c, s))) \vee (\alpha \neq \beta \wedge (cngBall(\alpha, c, s) \vee rcvBall(\alpha, c, s)))) \, .
\end{aligned}
$$

Here, $cngBall(\alpha, c, s)$ is true iff the ball possession changes to α after an action c in s (in the case of either an adversary block or an offside ball passage). The predicate $rcvBall(\alpha, c, s)$ is true iff agent α receives the ball from the ball owner or is in offside.

As for the stochastic theory ST, we assume the stochastic action $moveS(\alpha, m)$, which represents agent α's attempt in doing m among N, S, E, W, $stand$, $passTo(\beta)$, and $receive$. It can either succeed, and then the deterministic action $move(\alpha, m)$ is executed, or it can fail, and then the deterministic action $move(\alpha, stand)$ (that is, no change) is executed. Furthermore, after each execution of $moveS(\alpha, m)$, agent α can observe the presence of a team mate α' in the direction of the movement, given that

agent α' is visible, that is, not covered by another agent:

$$
\begin{array}{l}
stochastic(\{moveS(\alpha,m)\},s,\{a\},\{obs(\beta,out)\},\mu) \equiv \\
\quad \exists \mu_1,\mu_2\,((a=move(\alpha,m)\wedge(out=succ\wedge\mu_1=0.8\vee out=fail\wedge\mu_1=0.1)\vee \\
\qquad a=move(\alpha,stand)\wedge(out=succ\wedge\mu_1=0.01\vee out=fail\wedge\mu_1=0.09))\wedge \\
\qquad (visible(\alpha,\alpha',a,s)\wedge(\beta=\alpha'\wedge\mu_2=0.7\vee\beta=none\wedge\mu_2=0.1)\vee \\
\qquad \neg visible(\alpha,\alpha',a,s)\wedge(\beta=\alpha'\wedge\mu_2=0\vee\beta=none\wedge\mu_2=0.2))\wedge\mu=\mu_1{\cdot}\mu_2)\,; \\
stochastic(\{moveS(\alpha,m),moveS(\alpha',m')\},s,\{a_\alpha,a_{\alpha'}\},\{o_\alpha,o_{\alpha'}\},\mu) \equiv \\
\quad \exists \mu_1,\mu_2\,(stochastic(\{moveS(\alpha,m)\},s,\{a_\alpha\},\{o_\alpha\},\mu_1)\wedge \\
\qquad stochastic(\{moveS(\alpha',m')\},s,\{a_{\alpha'}\},\{o_{\alpha'}\},\mu_2)\wedge\mu=\mu_1\cdot\mu_2)\,.
\end{array}
$$

Here, $visible(\alpha,\alpha',a,s)$ is true if α can observe α' after the execution of a in s. The stochastic action $moveS(\alpha,m)$ is associated with the observations $obs(\beta,out)$, where $\beta\in\{\alpha',none\}$ and $r\in\{succ,fail\}$. That is, after the execution of the action $move(\alpha,m)$, agent α can observe both whether its team mate α' is present or not (first argument) and the success or failure of the action (second argument). Note that we assume that $obs(\alpha',out)$ has the probability zero, if α' is not visible. Notice also that in the last axiom, we assume the independence of the observations.

As for the optimization theory OT, the reward function for the agents is defined by:

$$
\begin{array}{l}
reward(\alpha,c,s)=r\equiv\exists\alpha'(goal(\alpha',do(c,s))\wedge(\alpha'\in\boldsymbol{a}\wedge r=M\vee \\
\quad \alpha'\in\boldsymbol{o}\wedge r=-M))\vee\neg\exists\alpha'\,(goal(\alpha',do(c,s))\wedge evalTeamPos(c,r,s))\,.
\end{array}
$$

Here, the reward of agent α is very high (that is, M stands for a "big" integer), if a team mate scores a goal. Otherwise, the reward depends on $evalTeamPos(c,r,s)$, that is, the position of its team relative to the adversary team as well as the ball possession.

3.2 Belief States

We next introduce belief states over situations, and define the semantics of actions in terms of transitions between belief states. A *belief state (over situations)* has the form $b=(b_i)_{i\in I}$, where every b_i is a set of pairs (s,μ) consisting of a standard situation s and a real $\mu\in(0,1]$ such that all μ sum up to 1. Informally, every b_i represents the belief of agent $i\in I$ expressed as a probability distribution over ordinary situations. The *probability* of a fluent formula $\phi(s)$ in $b=(b_i)_{i\in I}$, denoted $\phi(b)$, is the probability vector $pr=(pr_i)_{i\in I}$, where every pr_i with $i\in I$ is the sum of all μ such that $\phi(s)$ is true and $(s,\mu)\in b_i$. Similarly, $reward(c,b)$ denotes the vector $r=(r_i)_{i\in I}$, where every r_i with $i\in I$ is the sum of all $reward(i,c,s)\cdot\mu$ such that $(s,\mu)\in b_i$.

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

The probability of observing o after executing the stochastic action c in the belief state $b=(b_i)_{i\in I}$, denoted $prob(c,b,o)$, is the vector $pr=(pr_i)_{i\in I}$, where every pr_i with $i\in I$ is the sum of all $\mu\cdot\mu'$ such that $(s,\mu)\in b_i$ and $\mu'=prob(c,s,n,o)>0$.

Example 3.2 (Rugby Domain cont'd). Consider the following scenario relative to the domain theory of Example 3.1 (see Fig. 1). We focus only on controlling the members of the team $\boldsymbol{a}$, which cooperate to score a goal against the (static) team $\boldsymbol{o}$. The captain $\boldsymbol{a}_1$ of $\boldsymbol{a}$ has a complete view of the situation, and its belief state $b_{\boldsymbol{a}_1}$ is shown in Fig. 1, upper part: There is only the situation s_1 with probability 1 such that $at(\boldsymbol{a}_1, 2, 1, s_1)$, $at(\boldsymbol{a}_2, 2, 4, s_1)$, $at(\boldsymbol{o}_2, 1, 1, s_1)$, $at(\boldsymbol{o}_1, 5, 2, s_1)$, and $haveBall(\boldsymbol{a}_1, s_1)$ are true. That is, the captain $\boldsymbol{o}_1$ of $\boldsymbol{o}$ is very close to the goal of $\boldsymbol{a}$. From the perspective of $\boldsymbol{a}_1$, the goal seems quite done: $\boldsymbol{a}_1$ can pass to $\boldsymbol{a}_2$, which has a paved way towards the goal. But $\boldsymbol{a}_1$ has to cooperate with $\boldsymbol{a}_2$, whose vision of the situation is more confused and expressed by the belief state $b_{\boldsymbol{a}_2}$ in Fig. 1, lower part: $\boldsymbol{a}_2$ could be either at (a) $(1, 1)$ or at (b) $(1, 2)$, and $\boldsymbol{a}_1$ could be either at (c) $(2, 1)$ or at (d) $(3, 1)$. Hence, $\boldsymbol{a}_2$'s belief state may e.g. be given by $b_{\boldsymbol{a}_2} = \{(s_{a,c}, 0.5), (s_{a,d}, 0.3), (s_{b,c}, 0.1), (s_{b,d}, 0.1)\}$.

3.3 Syntax

Given the actions specified by a domain theory DT, a *program* p in POGTGolog has one of the following forms (where α is a multi-agent action, ϕ is a condition, p, p_1, p_2 are programs, $a_{i,1}, \ldots, a_{i,n_i}$ are actions of agents $i \in I$, and $J \subseteq I$ with $|J| \geq 2$):

1. *Deterministic or stochastic action:* α. Do α.
2. *Nondeterministic action choice of agent* $i \in I$: **choice**$(i\colon a_{i,1} | \cdots | a_{i,n_i})$. Do an optimal action (for agent $i \in I$) among $a_{i,1}, \ldots, a_{i,n_i}$.
3. *Nondeterministic joint action choice:* $\|_{j \in J}$**choice**$(j : a_{j,1} | \cdots | j : a_{j,n_j})$. Do any action $\|_{j \in J} a_{j,i_j}$ with an optimal probability $\pi = \Pi_{j \in J} \pi_{j,i_j}$.
4. *Test action*: $\phi?$. Test the truth of ϕ in the current situation.
5. *Action sequence*: $[p_1; p_2]$. Do p_1 followed by p_2.
6. *Nondeterministic choice of two programs:* $(p_1 \,|\, p_2)$. Do p_1 or p_2.
7. *Nondeterministic choice of an argument:* $\pi x\, (p(x))$. Do any $p(x)$.
8. *Nondeterministic iteration*: $p^\star$. Do p zero or more times.
9. *Conditionals*: **if** ϕ **then** p_1 **else** p_2.
10. *While-loops*: **while** ϕ **do** p.
11. *Procedures, including recursion.*

Hence, compared to Golog, we now also have multi-agent actions and stochastic actions (instead of only primitive resp. deterministic actions). Furthermore, we now additionally have different kinds of nondeterministic action choices for the agents in 2 and 3, where one or any subset of the agents in I can choose among a finite set of single-agent actions. The formal semantics of 2 and 3 is defined in such a way that an optimal action is chosen for the agents (see Section 3.4). As usual, the sequence operator ";" is associative (for example, $[[p_1; p_2]; p_3]$ and $[p_1; [p_2; p_3]]$ have the same semantics), and we often use "$p_1; p_2$" to abbreviate "$[p_1; p_2]$" when there is no danger of confusion.

Example 3.3 (Rugby Domain cont'd). Consider again the scenario (and its belief states $b_{\boldsymbol{a}_1}$ and $b_{\boldsymbol{a}_2}$) of Example 3.2 relative to the domain theory of Example 3.1 (see Fig. 1). Both agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ have to decide when (and if) it is worth to pass the ball, considering that if $\boldsymbol{a}_1$ tries to pass while $\boldsymbol{a}_2$ is in offside (for example, in $s_{a,d}$ or $s_{b,d}$), then the

ball goes to the captain $\boldsymbol{o}_1$ of the adversary team $\boldsymbol{o}$, which is in a very good position to score a goal. The subsequent POGTGolog program, denoted $schema$, represents a way of acting of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in this scenario, where the agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ have two possible chances to coordinate themselves in order to pass the ball, and thereafter both of them have to run towards the goal (with or without the ball).

choice($\boldsymbol{a}_1$: $moveS(\boldsymbol{a}_1, E) \mid moveS(\boldsymbol{a}_1, stand) \mid moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2))) \parallel$
  choice($\boldsymbol{a}_2$: $moveS(\boldsymbol{a}_2, S) \mid moveS(\boldsymbol{a}_2, E) \mid moveS(\boldsymbol{a}_2, receive)$);
choice($\boldsymbol{a}_1$: $moveS(\boldsymbol{a}_1, E) \mid moveS(\boldsymbol{a}_1, stand) \mid moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2))) \parallel$
  choice($\boldsymbol{a}_2$: $moveS(\boldsymbol{a}_2, E) \mid moveS(\boldsymbol{a}_2, receive)$);
$\{moveS(\boldsymbol{a}_1, E), moveS(\boldsymbol{a}_2, E)\}$;
$\{moveS(\boldsymbol{a}_1, E), moveS(\boldsymbol{a}_2, E)\}$.

3.4 Semantics

We now define the formal semantics of POGTGolog programs p relative to a domain theory $DT = (AT, ST, OT)$ in terms of Nash equilibria. We first associate with every POGTGolog program p, a belief state b, and horizon $H \geq 0$, a set of executable H-step policies π along with their expected utility U_i to every agent $i \in I$. We then define the notion of a Nash equilibrium to characterize a subset of optimal such policies, which is the natural semantics of a POGTGolog program relative to a domain theory.

Intuitively, given a horizon $H \geq 0$, an H-step policy π of a POGTGolog program p relative to a domain theory is obtained from the H-horizon part of p by replacing every single-agent choice by a single action, and every multi-agent choice by a collection of probability distributions, one over the actions of each agent. Formally, for every POGTGolog program p, we define the *nil-terminated variant* of p, denoted $\widehat{p}$, by $\widehat{p} = [p_1; \widehat{p}_2]$, if $p = [p_1; p_2]$, and $\widehat{p} = [p; nil]$, otherwise. Given a POGTGolog program p relative to a domain theory DT, a horizon $H \geq 0$, and a start belief state b, we say that π is an *H-step policy* of p in b iff $DT \models G(\widehat{p}, b, H, \pi, \langle v, pr \rangle)$, where $v = (v_i)_{i \in I}$ and $pr = (pr_i)_{i \in I}$. The *expected H-step utility* of π in b to $i \in I$, denoted $U_i(H, b, \pi)$, is $utility(v_i, pr_i)$. Here, we define the macro $G(\widehat{p}, b, h, \pi, \langle v, pr \rangle)$ by induction as follows:

- Null program ($\widehat{p} = nil$) or zero horizon ($h = 0$):

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

  Intuitively, p ends when it is null or at the horizon end.
- First program action c is deterministic (resp., stochastic with observation):

  $$\begin{array}{l} G([c; p'], b, h, \pi, \langle v, pr \rangle) =_{def} \\ \quad (Poss(c, b) = \mathbf{0} \wedge \pi = stop \wedge v = \mathbf{0} \wedge pr = \mathbf{1}) \vee \\ \quad (Poss(c, b) > \mathbf{0} \wedge \exists \pi', v', pr' \, (G(p', do(c, b), h{-}1, \pi', \langle v', pr' \rangle) \wedge \\ \qquad \pi = c; \pi' \wedge v = v' + reward(c, b) \wedge pr = pr' \cdot Poss(c, b)) . \end{array}$$

  Here, $(s_i)_{i \in I} \; op \; (t_i)_{i \in I} = (s_i \; op \; t_i)_{i \in I}$ for $op \in \{+, \cdot\}$. Informally, suppose that $\widehat{p} = [c; p']$, where c is a deterministic action (resp., stochastic action with observation). If c is not executable in the belief state b, then p has only the policy $\pi = stop$ along with the expected reward $v = \mathbf{0}$ and the success probability $pr = \mathbf{0}$. Otherwise, the optimal execution of $[c; p']$ in the belief state b depends on that one of p in $do(c, b)$. Observe that c is executable in b with the probability $Poss(c, b)$, which affects the overall success probability pr.

- Stochastic first program action c (choice of nature):

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

  Here, $o_1,\ldots,o_l$ are the possible observations. The generated policy is a conditional plan in which every such observation o_q is considered.
- Nondeterministic first program action (choice of agent $i \in I$):

$$G([\textbf{choice}(i\colon a_1|\cdots|a_n)\,;p'],b,h,\pi,\langle v,pr\rangle) =_{def}$$
$$\exists \pi_q, v_q, pr_q, k\,(\textstyle\bigwedge_{q=1}^{n} G([a_q\,;p'],b,h,a_q\,;\pi_q,\langle v_q,pr_q\rangle) \wedge$$
$$k \in \{1,\ldots,n\} \wedge \pi = a_k\,; \textbf{for } q=1 \textbf{ to } n \textbf{ do if } \psi_q \textbf{ then } \pi_q \ \wedge$$
$$v = v_k \wedge pr = pr_k)\,.$$

  Here, the ψ_q's denote conditions that the other agents in I test to observe j's choice.
- Nondeterministic first program action (joint choice of the agents in J):

$$G([\,\|_{j\in J}\textbf{choice}(j\colon a_{j,1}|\cdots|a_{j,n_j});p'],b,h,\pi,\langle v,pr\rangle) =_{def}$$
$$\exists \pi_a, v_a, pr_a, \pi_a\,(\textstyle\bigwedge_{a\in A} G([\bigcup_{j\in J} a_j;p'],b,h,\bigcup_{j\in J} a_j;\pi_a,\langle v_a,pr_a\rangle) \wedge$$
$$\textstyle\bigwedge_{j\in J}(\pi_j \in PD(\{a_{j,1},\ldots,a_{j,n_j}\})) \wedge$$
$$\pi = \|_{j\in J}\pi_j; \textbf{for each } a\in A \textbf{ do if } \phi_a \textbf{ then } \pi_a \wedge$$
$$v = \textstyle\sum_{a\in A} v_a \cdot \Pi_{j\in J}\pi_j(a_j) \wedge pr = \sum_{a\in A} pr_a \cdot \Pi_{j\in J}\pi_j(a_j)\,.$$

  Here, $A = \times_{j\in J}\{a_{j,1},\ldots,a_{j,n_j}\}$, and each π_j with $j \in J$ is a probability distribution over $\{a_{j,1},\ldots,a_{j,n_j}\}$. Informally, we compute the joint policy for each possible combination of actions $a \in A$. The conditions ϕ_a with $a \in A$ are to observe what the agents have actually executed.
- Nondeterministic choice of two programs:

$$G([(p_1 \mid p_2);p'],b,h,\pi,\langle v,pr\rangle) =_{def}$$
$$\exists \pi_q, v_q, pr_q, k\,(\textstyle\bigwedge_{q\in\{1,2\}} G([p_q;p'],b,h,\pi_q,\langle v_q,pr_q\rangle) \wedge$$
$$k \in \{1,2\} \wedge \pi = \pi_k \wedge v = v_k \wedge pr = pr_k)\,.$$

- Test action:

$$G([\phi?\,;p'],b,h,\pi,\langle v,pr\rangle) =_{def} (\phi[b] = \mathbf{0} \wedge \pi = stop \wedge v = \mathbf{0} \wedge pr = \mathbf{0}) \vee$$
$$\exists pr'(\phi[b] > \mathbf{0} \wedge G(p',b,h,\pi,\langle v,pr'\rangle) \wedge pr = pr' \cdot \phi[b])\,.$$

  Informally, let $p = [\phi?\,;p']$. If ϕ is false in b, then p has only the policy $\pi = stop$ along with the expected reward $v = \mathbf{0}$ and the success probability $pr = \mathbf{0}$. Otherwise, π is a policy of p with the expected reward v and success probability $pr' \cdot \phi[b]$ iff π is a policy of p' with the expected reward v and success probability pr'.
- The macro G is naturally extended to nondeterministic choices of action arguments, nondeterministic iterations, conditionals, while-loops, and procedures.

We are now ready to define the notion of a Nash equilibrium as follows. An H-step policy of a POGTGolog program p in a belief state b is an *H-step Nash equilibrium* of p in b iff, for every agent $i \in I$, it holds that $U_i(H,b,\pi) \leq U_i(H,b,\pi')$ for all H-step policies π' of p in b obtained from π by modifying only actions of agent i.

Example 3.4 (Rugby Domain cont'd). Consider again the scenario (and its belief states $b_{\boldsymbol{a}_1}$ and $b_{\boldsymbol{a}_2}$) of Example 3.2 relative to the domain theory of Example 3.1 (see Fig. 1). Assuming the horizon $H=4$, a 4-step policy π of the POGTGolog program $schema$ of Example 3.3 is given by $DT \models G([schema; nil],\ (b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2}), 4, \pi, \langle (v_1, v_2), (pr_1, pr_2)\rangle)$. For agent $\boldsymbol{a}_1$, an optimal way of acting is to pass the ball as soon as possible, which can be encoded by the following (pure) 4-step policy $\pi_{\boldsymbol{a}_1} = c\,;\, \pi^1_{\boldsymbol{a}_1}$, where $c = \{moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2)), moveS(\boldsymbol{a}_2, receive)\}$, and $\pi^1_{\boldsymbol{a}_1}$ is an optimal 3-step policy of $schema'$ in the belief state $(do(c, b_{\boldsymbol{a}_1}), do(c, b_{\boldsymbol{a}_2}))$. Here, $schema'$ is obtained from $schema$ by removing the first nondeterministic joint action choice. The policy $\pi_{\boldsymbol{a}_1}$ gives to agent $\boldsymbol{a}_2$ three $moveS(\boldsymbol{a}_2, E)$ attempts to achieve the touch-line. From the standpoint of $\boldsymbol{a}_2$, instead, it is worth to do a $moveS(\boldsymbol{a}_2, S)$ to observe if agent $\boldsymbol{a}_1$ is aligned, trying to minimize the likelihood of a wrong passage. In this case, $\boldsymbol{a}_1$ has to delay the passage waiting for the move of $\boldsymbol{a}_2$. The resulting (pure) 4-step policy $\pi_{\boldsymbol{a}_2}$ is more favorable to $\boldsymbol{a}_2$'s belief state:

$$\begin{array}{l}\pi_{\boldsymbol{a}_2} = c\,;\ \textbf{if}\ obs(\boldsymbol{a}_1, succ)\ \textbf{then}\ \pi^{1,o_1}_{\boldsymbol{a}_2} \\ \qquad\qquad \textbf{else if}\ obs(\boldsymbol{a}_1, fail)\ \textbf{then}\ \pi^{1,o_2}_{\boldsymbol{a}_2} \\ \qquad\qquad\quad \textbf{else if}\ obs(none, succ)\ \textbf{then}\ \pi^{2,o_3}_{\boldsymbol{a}_2} \\ \qquad\qquad\qquad \textbf{else if}\ obs(none, fail)\ \textbf{then}\ \pi^{2,o_4}_{\boldsymbol{a}_2}\,,\end{array}$$

where $c = \{moveS(\boldsymbol{a}_1, S), moveS(\boldsymbol{a}_2, stand)\}$ and $\pi^{k,o_i}_{\boldsymbol{a}_2}$ is an optimal 3-step policy of $schema'$, when observing o_i after the execution of c from $(b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2})$. Given this conflict of opinions, an optimal compromise for both $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ is a Nash equilibrium.

4 A POGTGolog Interpreter

In this section, we define an interpreter for POGTGolog programs relative to domain theories and provide optimality and representation results.

We define an interpreter for POGTGolog programs p relative to a domain theory DT by specifying the macro $DoG(\widehat{p}, b, H, \pi, \langle v, pr\rangle)$, which takes as input the nil-terminated variant $\widehat{p}$ of a POGTGolog program p, a belief state $b = (b_i)_{i\in I}$, and a finite horizon $H \geq 0$, and which computes as output an optimal H-step policy π and the vectors $v = (v_i)_{i\in I}$ and $pr = (pr_i)_{i\in I}$, respectively, where v_i is the expected H-step reward of π to i, and $pr_i \in [0,1]$ is the H-step success probability of π for i.

We define the macro $DoG(\widehat{p}, b, h, \pi, \langle v, pr\rangle)$ in nearly the same way as the macro $G(\widehat{p}, b, h, \pi, \langle v, pr\rangle)$ in Section 3.4, except for the following modifications:

- Nondeterministic first program action (choice of agent $i \in I$): The characterization of DoG is obtained from the one of G by replacing the condition "$k \in \{1, \ldots, n\}$" by the condition "$k = \text{argmax}_{q\in\{1,\ldots,n\}}\ utility(v_{q,i}, pr_{q,i})$", where $v_q = (v_{q,i})_{i\in I}$ and $pr_q = (pr_{q,i})_{i\in I}$. Informally, given the possible actions $a_1, \ldots, a_n$ for agent $i \in I$, we select an optimal one for i, that is, one with greatest $utility(v_{q,i}, pr_{q,i})$.
- Nondeterministic first program action (joint choice of the agents in J): The characterization of DoG is obtained from the one of G by replacing "$\bigwedge_{j\in J}(\pi_j \in PD(\{a_{j,1}, \ldots, a_{j,n_j}\}))$" by "$(\pi_j)_{j\in J} = selectNash(\{utility(v_a, pr_a)|_J \mid a \in A\})$", where $utility((s_i)_{i\in I}, (t_i)_{i\in I}) = (utility(s_i, t_i))_{i\in I}$, and $s|_J$ is the restriction of s to J, for $s = (s_i)_{i\in I}$ and $J \subseteq I$. Informally, we compute a local Nash equilibrium

$(\pi_j)_{j\in J}$ from a normal form game using the Nash selection function $selectNash$. Note that we assume that all agents have the same Nash selection functions, and thus they automatically select a common unique Nash equilibrium.

- Nondeterministic choice of two programs: The characterization of DoG is obtained from the one of G by replacing "$k \in \{1,2\}$". by "$k = \text{argmax}_{q\in\{1,2\}}\ utility(v_{q,j}, pr_{q,j})$". Informally, given two possible programs p_1 and p_2, we select an optimal one for agent j, that is, one with greatest $utility(v_{q,j}, pr_{q,j})$.

The following theorem shows the important result that the macro DoG is optimal in the sense that, for every horizon $H \geq 0$, among the set of all H-step policies π of a POGTGolog program p relative to a domain theory DT in a belief state b, it computes an H-step Nash equilibrium and its expected H-step utility.

Theorem 4.1. *Let $DT = (AT, ST, OT)$ be a domain theory, and let p be a POGTGolog program relative to DT. Let b be a belief state (over situations), let $H \geq 0$ be a horizon, and let $DT \models DoG(\widehat{p}, b, H, \pi, \langle v, pr\rangle)$. Then, π is an H-step Nash equilibrium of p in b, and $utility(v_i, pr_i)$ is its expected H-step utility to agent $i \in I$.*

The next theorem shows that, given any horizon $H \geq 0$, every POSG can be encoded as a program p in POGTGolog, such that DoG computes one of its H-step Nash equilibria and its expected H-step reward.

Theorem 4.2. *Let $G = (I, Z, (A_i)_{i\in I}, (O_i)_{i\in I}, P, (R_i)_{i\in I})$ be a POSG, let $H \geq 0$ be a horizon, and let b_0 be a belief state of G. Then, there exists a domain theory $DT = (AT, ST, OT)$, and a set of POGTGolog programs $\{\widehat{p}^h \mid h \in \{0, \ldots, H\}\}$ relative to DT such that $\delta = (\delta_i)_{i\in I}$ is an H-step Nash equilibrium for G, where every $(\delta_i(b, h))_{i\in I} = (\pi_i)_{i\in I}$ is given by $DT \models DoG(\widehat{p}^h, B_b, h{+}1, \|_{i\in I}\pi_i\,;\pi',\ \langle v, pr\rangle)$, for every belief state b reachable from b_0 and every $h \in \{0, \ldots, H\}$, where B_b is the belief state over situations associated with the belief state b of G. Furthermore, the expected H-step reward $G_i(H, b, \delta)$ to agent $i \in I$ is given by $utility(v_i,\ pr_i)$, where $DT \models DoG(\widehat{p}^H, B_b, H{+}1, \pi, \langle v, pr\rangle)$, for every belief state b reachable from b_0.*

5 Summary and Outlook

We have presented the agent programming language POGTGolog, which combines explicit agent programming in Golog with game-theoretic multi-agent planning in POSGs, and which allows for modeling one team of cooperative agents under partial observability, where the agents may have different initial belief states and not necessarily the same rewards. It allows for specifying a partial control program in a high-level logical language, which is then completed by an interpreter in an optimal way. To this end, we have defined a formal semantics of POGTGolog programs in terms of Nash equilibria, and specified a POGTGolog interpreter that computes one of these Nash equilibria. We have illustrated the usefulness of this approach along a rugby scenario.

An interesting topic of future research is to generalize POGTGolog to the case in which we can give up the assumption that every agent knows the initial local belief states of all the other agents, their locally executed actions, and their local observations.

This may, for example, be achieved by explicit communication between the agents or by independence assumptions between the local actions and observations of different agents. A further direction of future research is to generalize POGTGolog to the case of two competing teams of cooperative agents under partially observability.

Acknowledgments. This work was 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 comments, which helped to improve this work.

References

1. F. Bacchus, J. Y. Halpern, and H. J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artif. Intell.*, 111:171–208, 1999.
2. C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proceedings IJCAI-2001*, pp. 690–700.
3. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings AAAI-2000*, pp. 355–362.
4. A. Ferrein, C. Fritz, and G. Lakemeyer. Using Golog for deliberation and team coordination in robotic soccer. *Künstliche Intelligenz*, 1:24–43, 2005.
5. A. Finzi and T. Lukasiewicz. Game-theoretic agent programming in Golog. In *Proceedings ECAI-2004*, pp. 23–27.
6. A. Finzi and F. Pirri. Combining probabilities, failures and safety in robot control. In *Proceedings IJCAI-2001*, pp. 1331–1336.
7. C. V. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res.*, 22:143–174, 2004.
8. C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational MDPs. In *Proceedings IJCAI-2003*, pp. 1003–1010.
9. E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings AAAI-2004*, pp. 709–715.
10. L. Pack Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1–2):99–134, 1998.
11. M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings ICML-1994*, pp. 157–163.
12. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of Artificial Intelligence. In *Machine Intelligence* 4, pp. 463–502. Edinburgh University Press, 1969.
13. R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings IJCAI-2003*, pp. 705–711. 2003.
14. G. Owen. *Game Theory: Second Edition*. Academic Press, 1982.
15. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.
16. R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.
17. J. van der Wal. *Stochastic Dynamic Programming*, volume 139 of *Mathematical Centre Tracts*. Morgan Kaufmann, 1981.
18. J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior*. Princeton University Press, 1947.
19. S. W. Yoon, A. Fern, and B. Givan. Inductive policy selection for first-order MDPs. In *Proceedings UAI-2002*, pp. 569–576.