INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

GAME-THEORETIC GOLOG UNDER PARTIAL OBSERVABILITY

ALBERTO FINZI   THOMAS LUKASIEWICZ

INFSYS RESEARCH REPORT 1843-05-02

DECEMBER 2006

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS Research Report
INFSYS Research Report 1843-05-02, December 2006

Game-Theoretic Golog under Partial Observability

December 27, 2006

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

Abstract. In this paper, we present the agent programming language POGTGolog (Partially Observable Game-Theoretic Golog), which integrates explicit agent programming in Golog with game-theoretic multi-agent planning in partially observable stochastic games. In this framework, we assume one team of cooperative agents acting under partial observability, where the agents may also 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 then specify a POGTGolog interpreter that computes one of these Nash equilibria.

[1]Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Rome, Italy; e-mail: {finzi, lukasiewicz}@dis.uniroma1.it.
[2]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; e-mail: lukasiewicz@kr.tuwien.ac.at.

Acknowledgements: This paper is a significantly extended and revised version of a paper that has appeared in *Proceedings KI-2006* [15]. Preliminary results of this paper have also appeared in *Proceedings GTDT-2005* [13] and *Proceedings AAMAS-2005* [14].

This work was partially supported by the Austrian Science Fund (FWF) under project P18146-N04 and by a Heisenberg Professorship of the German Research Foundation (DFG).

Contents

1 Introduction

During the recent years, the development of controllers for autonomous agents has become increasingly important in Artificial Intelligence (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 [5], which integrates explicit agent programming in Golog [41] with decision-theoretic planning in (fully observable) Markov decision processes (MDPs) [38]. 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. 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 by providing fragments of control knowledge (that is, partially specified control programs).

The agent programming language DTGolog has several other nice features, since it is closely related to first-order extensions of decision-theoretic planning (see especially [4, 49, 22]), 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, 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 [11], which is a generalization of DTGolog that integrates agent programming in Golog with game-theoretic multi-agent planning in (fully observable) stochastic games [33], also called Markov games [45, 27]. The following example shows a program in GTGolog.

Example 1.1 *(Rugby Domain)* 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
    $\pi x\, (\pi y\, ($choice$(\boldsymbol{a}_1 : moveTo(x) \mid passTo(\boldsymbol{a}_2)) \parallel$
        choice$(\boldsymbol{a}_2 : moveTo(y) \mid receive(\boldsymbol{a}_1))))$;
$step(n{-}1)$
end.
```

This program encodes that while agent $\boldsymbol{a}_1$ 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.

However, 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, in the robotic rescue domain, every agent has generally only a very partial view on the environment and the other agents. But 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. POGTGolog allows for modeling one team of cooperative agents under partial observability, where the agents may also 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). We assume a system of multiple agents with free communication (as in [42, 39]).

- 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 associates with each POGTGolog program one of its Nash equilibria. We also prove that POGTGolog programs faithfully extend partially observable stochastic games. Furthermore, we illustrate the usefulness of the POGTGolog approach along several examples.

The rest of this paper is organized as follows. In Section 2, we recall the basic concepts of the situation calculus, Golog, normal form games, and partially observable stochastic games. Section 3 defines the domain theory, syntax, and semantics of POGTGolog programs. In Section 4, we formally specify a POGTGolog interpreter and provide especially an optimality result for the interpreter. Section 5 illustrates the overall framework through another example. In Section 6, we discuss related work. Finally, Section 7 summarizes our main results and gives an outlook on future research. Notice that detailed proofs of all results are given in Appendix A.

2 Preliminaries

In this section, we first recall the main concepts of the situation calculus (in its standard and concurrent version) and of the agent programming language Golog; for further details and background see especially [41]. We then recall the basics of normal form games and partially observable stochastic games.

2.1 The Situation Calculus

The situation calculus [31, 41] 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. 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. We write $Poss(a, s)$, where $Poss$ is a distinguished binary predicate symbol, to denote that it is possible to execute the action a 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. A situation calculus formula is *uniform* in a situation s iff (i) it does not mention the predicates $Poss$ and $\sqsubset$ (which denotes the proper subsequence relationship on situations), (ii) it does not quantify over situation variables, (iii) it does not mention equality on situations, and (iv) every situation in the situation argument of a fluent coincides with s [41].

Example 2.1 *(Rugby Domain cont'd)* The action $moveTo(r, x, y)$ may stand for moving the agent r to the position (x, y), while the situation $do(moveTo(r, 1, 2), do(moveTo(r, 3, 4), S_0))$ stands for executing $moveTo(r, 1, 2)$ after the execution of $moveTo(r, 3, 4)$ in the initial situation S_0. The (relational) fluent $at(r, x, y, s)$ 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 [41].
- $\mathcal{D}_{una}$ is the set of unique names axioms for actions, which express that different actions are interpreted in a different way (that is, any two actions with different names or different arguments have a different meaning).
- $\mathcal{D}_{S_0}$ is a set of first-order formulas that are uniform in S_0, which describe the initial state of the domain (represented by S_0).
- $\mathcal{D}_{ssa}$ is the set of *successor state axioms* [40, 41]. 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 uniform in s 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).
- $\mathcal{D}_{ap}$ is the set of *action precondition axioms*. For each action a, it contains an axiom of the form $Poss(a(\boldsymbol{x}), s) \equiv \Pi(\boldsymbol{x}, s)$, where Π is a formula uniform in s with free variables among $\boldsymbol{x}, s$. This axiom characterizes the preconditions of a.

Example 2.2 *(Rugby Domain cont'd)* The formula $at(r,1,2,S_0)\wedge at(r',3,4,S_0)$ in $\mathcal{D}_{S_0}$ may express that the agents r and r' are initially at the positions $(1,2)$ and $(3,4)$, respectively. The axiom $at(r,x,y,do(a, s))\equiv a = moveTo(r,x,y)\vee(at(r,x,y,s)\wedge\neg\exists x',y'\,(a = moveTo(r,x',y')))$ in $\mathcal{D}_{ssa}$ 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. The axiom $Poss(moveTo(r,x,y),s)\equiv \neg\exists r'\,at(r',x,y,s)$ in $\mathcal{D}_{ap}$ may express that it is possible to move the agent r to (x,y) in s iff no agent r' is at (x,y) in s.

In this paper, we use the concurrent version of the situation calculus [41], which is an extension of the above standard situation calculus by concurrent actions. A *concurrent action* c is a set of standard actions, which are concurrently executed when c is executed. A situation is then a sequence of concurrent actions of the form $do(c_m,\dots,do(c_0,S_0))$, where $do(c,s)$ states that all the simple actions a in c are executed at the same time in the situation s.

In order to encode concurrent actions, some slight modifications to standard basic action theories are necessary. In particular, the successor state axioms in $\mathcal{D}_{ssa}$ are now defined relative to concurrent actions. Furthermore, the action preconditions in $\mathcal{D}_{ap}$ are extended by further axioms expressing (i) that a singleton concurrent action $c = \{a\}$ is executable if its standard action a is executable, (ii) that if a concurrent action is executable, then it is nonempty and all its standard actions are executable, and (iii) preconditions for concurrent actions. Note that precondition axioms for standard actions are in general not sufficient, since two standard actions may each be executable, but their concurrent execution may not be permitted. This *precondition interaction problem* [41] (see also [36] for a discussion) requires some domain-dependent extra precondition axioms.

Example 2.3 *(Rugby Domain cont'd)* The axiom $at(r,x,y,do(a,s))\equiv a = moveTo(r,x,y)\vee(at(r,x, y,s)\wedge\neg\exists x',y'\,(a = moveTo(r,x',y')))$ in $\mathcal{D}_{ssa}$ in the standard situation calculus is replaced by the axiom $at(r,x,y,do(c,s))\equiv moveTo(r,x,y)\in c\vee(at(r,x,y,s)\wedge\neg\exists x',y'(moveTo(r,x',y')\in c))$ in the concurrent one.

2.2 Golog

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

Example 2.4 *(Rugby Domain cont'd)* 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 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 p in the situation s.

2.3 Normal Form Games

Normal form games from classical game theory [47] describe the possible actions of $n \geqslant 2$ agents and the rewards that the agents receive when they simultaneously execute one action each. 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 $n \geqslant 2$, a nonempty finite set of *actions* A_i for each agent $i \in I$, and a *reward function* $R_i\colon A \rightarrow \mathbf{R}$ for each $i \in I$, which associates with every *joint action* $a \in A = \times_{i\in I} A_i$ a *reward* $R_i(a)$ to agent i. If $n = 2$, then G is a *two-player normal form game* (or *matrix game*). If also $R_1 = -R_2$, 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. 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 $\sum_{a=(a_j)_{j\in I}\in A} R_i(a) \cdot \Pi_{j\in I}\pi_j(a_j)$.

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 $R_i(\pi \leftarrow \pi'_i) \leqslant R_i(\pi)$ for every mixed strategy π'_i, where $\pi \leftarrow \pi'_i$ is obtained from π by replacing π_i by π'_i. 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 $v^i_f(G)$.

Example 2.5 *(two-finger Morra)* 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. A pure strategy for player E (or O) is to show two fingers, and a mixed strategy for E (or O) is to show one finger with the probability $7/12$ and two fingers with the probability $5/12$. 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.

2.4 Partially Observable Stochastic Games

Partially observable stochastic games [33] generalize normal form games, partially observable Markov decision processes (POMDPs) [34], and decentralized POMDPs (Dec-POMDPs) [35, 21, 32]. 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* $I = \{1, \ldots, n\}$, $n \geqslant 2$, 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 agent 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 $H \geqslant 0$, 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 $G_i(H, b, \pi \leftarrow \pi'_i) \leqslant G_i(H, b, \pi)$ 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 [11] that allows for partial observability. We first define the domain theory and belief states of POGTGolog. We then introduce the syntax and the semantics of POGTGolog programs.

We focus on the case of one team of cooperative agents acting under partial observability, where the agents may also have different initial belief states and not necessarily the same rewards. That is, the agents are collaborative in the sense that they belong to the same team, but they may also have to solve conflicts of opinions about the state of the world (expressed through different initial beliefs) and its significance (expressed through different rewards). Like in [42, 39], we assume *free communication* between the agents, which means that the agents can communicate without cost, and thus we can assume that (i) each agent is aware about the initial local belief state of every other agent, and (ii) after each action execution, each agent can observe the actions of the other agents 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 and reward/utility functions. 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 defined below.

We assume a team $I = \{1, \ldots, n\}$ of $n \geqslant 2$ cooperative agents $1, \ldots, n$. The nonempty finite 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). Every multi-agent action c is associated with a nonempty finite set of *multi-agent observations* $O_c = \bigtimes_{i \in I} O_{c,i}$, where every $O_{c,i}$ is a nonempty finite set of pairwise exclusive and exhaustive conditions (called *single-agent observations*) of agent $i \in I$.

Example 3.1 *(Rugby Domain cont'd)* Let $I = \{1, 2\}$. Then, the concurrent actions $\{\mathit{moveTo}(1,1,2)\} \subseteq A_1$ and $\{\mathit{moveTo}(2,2,3)\} \subseteq A_2$ are single-agent actions of agents 1 and 2, respectively, and thus also multi-agent actions, while the concurrent action $\{\mathit{moveTo}(1,1,2), \mathit{moveTo}(2,2,3)\}$ is a multi-agent action.

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 [18, 5]. 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 $\mathit{stochastic}(c, s, n, o, \mu)$ to encode that when executing the stochastic action c in the situation s, "nature" chooses the deterministic action n producing the observation $o \in O_c$ with the probability μ. Here, for every stochastic action c and situation s, the set of all (n, o, μ) such that $\mathit{stochastic}(c, s, n, o, \mu)$ is a probability function on the set of all deterministic components n and observations $o \in O_c$ of c in s. We also use the notation $\mathit{prob}(c, s, n, o)$ to denote the probability μ such that $\mathit{stochastic}(c, s, n, o, \mu)$. We assume that c and all its nature choices n have the same preconditions. A stochastic action c is then 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 the observation $o \in O_c$, denoted $\mathit{Poss}(c_o, s)$, iff $\mathit{prob}(c, s, n, o) > 0$ for some n.

Example 3.2 *(Rugby Domain cont'd)* The stochastic action $\mathit{moveS}(\alpha, x, y)$, which (i) is observed as being either successful or a failure, and (ii) moves the agent α to either the position (x, y) or the position $(x, y{+}1)$, can be encoded as follows:

$$\begin{aligned}
&\mathit{stochastic}(\mathit{moveS}(\alpha, x, y), s, n, o, \mu) \overset{def}{=} \exists y' (n = \mathit{moveTo}(\alpha, x, y') \wedge \\
&\quad (o = \mathit{obs}(\mathit{success}) \wedge (y' = y + 1 \wedge \mu = 0.5 \vee y' = y \wedge \mu = 0.3) \vee \\
&\quad\; o = \mathit{obs}(\mathit{failure}) \wedge y' = y + 1 \wedge \mu = 0.2))\,.
\end{aligned}$$

The precondition and successor state axioms of $\mathit{moveS}(\alpha, x, y)$ are then specified through the precondition and successor state axioms of $\mathit{moveTo}(\alpha, x, y')$.

The *optimization theory* OT specifies a reward function, a utility function, and some Nash selection functions. The reward function associates with each situation s and multi-agent action a, a reward to each agent $i \in I$, denoted $\mathit{reward}(i, a, s)$. Note that the reward function for stochastic actions is defined through

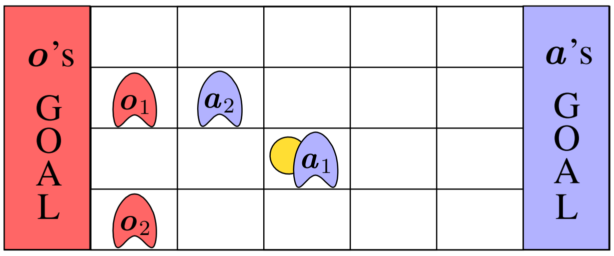

Figure 1: Rugby Domain: Two teams $\{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ and $\{\boldsymbol{o}_1, \boldsymbol{o}_2\}$.

a reward function for their deterministic components. The utility function $utility$ maps every pair consisting of a reward v and a probability value pr (that is, a real from the unit interval $[0, 1]$) to a real-valued utility $utility(v, pr)$. We assume that $utility(v, 1) = v$ and $utility(v, 0) = 0$ for all rewards v. Informally, the utility function suitably mediates between the rewards of the agents and the failure of actions due to unsatisfied preconditions: differently from actions in decision-theoretic planning, the actions here may fail due to unsatisfied preconditions. Thus, the usefulness of an action/program does not only depend on its reward, but also on the probability that it is executable. Similarly to all arithmetic operations, utility functions are pre-interpreted (that is, rigid) and thus not explicitly axiomatized in the domain theory.

Example 3.3 *(Rugby Domain cont'd)* The reward function $reward(1, \{moveTo(1, x, y)\}, s) = y$ may encode that the reward to agent 1 when moving to (x, y) in the situation s is given by y. An example of a utility function is $utility(v, pr) = v \cdot pr$.

The following example describes a more detailed domain theory for the Rugby Domain, which is inspired by the soccer domain in [27].

Example 3.4 *(Rugby Domain cont'd)* The rugby field (see Fig. 1) is a 4×7 grid of 28 squares, and it includes two designated areas representing two goals. 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, $stand$, $passTo(\alpha)$, and $receive(\alpha)$ (move up, move down, move right, move left, no move, pass the ball to α, and receive the ball from α, respectively). The ball is represented by a circle 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 another agent, if the other agent stands, then the ball possession changes. Thus, a good defensive maneuver is to stand where the ball owner wants to go.

We define the domain theory $DT = (AT, ST, OT)$ as follows. The basic action theory AT defines the deterministic action $move(\alpha, m)$ (encoding that agent α executes m), where $\alpha \in \boldsymbol{a} \cup \boldsymbol{o}$, $m \in \{N, S, E, W,$ $stand, passTo(\alpha'), receive(\alpha')\}$, 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:

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

Here, $\mathit{cngBall}(\alpha,c,s)$ is true iff the ball possession changes to α after an action c in s (in the case of an adversary block), that is,

$$\begin{array}{l} \mathit{cngBall}(\alpha,c,s) \overset{def}{=} \exists\beta,x,y\,(\beta\neq\alpha \wedge \neg\mathit{sameTeam}(\alpha,\beta) \wedge \mathit{haveBall}(\beta,s)\,\wedge \\ \quad \mathit{at}(\beta,x,y,do(c,s)) \wedge \mathit{at}(\alpha,x,y,s) \wedge \mathit{move}(\alpha,\mathit{stand})\in c)\,. \end{array}$$

The predicate $\mathit{rcvBall}(\alpha,c,s)$ is true iff agent α receives the ball from the ball owner or because of an offside ball passage, that is,

$$\begin{array}{l} \mathit{rcvBall}(\alpha,c,s) \overset{def}{=} \exists\beta,\alpha'\,(\beta\neq\alpha \wedge \mathit{haveBall}(\beta,s)\,\wedge \\ \quad \mathit{move}(\beta,\mathit{passTo}(\alpha'))\in c \wedge ((\mathit{move}(\alpha',\mathit{receive}(\beta))\,\wedge \\ \quad \neg\mathit{offside}(\beta,\alpha',s) \wedge \alpha'=\alpha) \vee (\neg\mathit{move}(\alpha',\mathit{receive}(\beta))\in c\,\vee \\ \quad \mathit{offside}(\beta,\alpha',s)) \wedge \neg\mathit{sameTeam}(\alpha,\alpha') \wedge \mathit{captain}(\alpha)))\,. \end{array}$$

Here, $\mathit{offside}(\beta,\alpha,s)$ is true iff the players α and β are team-mates, and α is closer to the goal of the adversary team than β. The deterministic actions $\mathit{move}(\alpha,m)$ are associated with precondition axioms by:

$$\begin{array}{l} \mathit{Poss}(\mathit{move}(\alpha,m),s) \equiv \neg\exists x,y\,(\mathit{at}(\alpha,x,y,s)\,\wedge \\ \quad ((y=4 \wedge m=N) \vee (y=1 \wedge m=S) \vee \\ \quad (x=6 \wedge m=E) \vee (x=0 \wedge m=W))) \vee \\ \quad \exists\beta(m=\mathit{passTo}(\beta) \wedge \mathit{haveBall}(\alpha,s)) \vee \\ \quad \exists\beta(m=\mathit{receive}(\beta) \wedge \mathit{haveBall}(\beta,s))\,. \end{array}$$

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

$$\begin{array}{l} \mathit{stochastic}(\{\mathit{moveS}(\alpha,m)\},s,\{a\},\{\mathit{obs}(\beta,\mathit{out})\},\mu) \overset{def}{=} \\ \quad \exists\mu_1,\mu_2\,((a=\mathit{move}(\alpha,m) \wedge (\mathit{out}=\mathit{success} \wedge \mu_1=0.8\,\vee \\ \qquad\qquad \mathit{out}=\mathit{failure} \wedge \mu_1=0.1) \vee a=\mathit{move}(\alpha,\mathit{stand})\,\wedge \\ \qquad\qquad (\mathit{out}=\mathit{success} \wedge \mu_1=0.01 \vee \mathit{out}=\mathit{failure} \wedge \mu_1=0.09)) \wedge \\ \qquad\quad (\mathit{visible}(\alpha,\beta,a,s) \wedge \mu_2=0.7 \vee \mathit{visible}(\alpha,\mathit{nil},a,s) \wedge \mu_2=0.1\,\vee \\ \qquad\qquad \neg\mathit{visible}(\alpha,\mathit{nil},a,s) \wedge \mu_2=0.2) \wedge \mu=\mu_1{\cdot}\mu_2)\,; \end{array}$$

$$\begin{array}{l} \mathit{stochastic}(\{\mathit{moveS}(\alpha,m),\mathit{moveS}(\alpha',m')\},s,\{a_\alpha,a_{\alpha'}\},\{o_\alpha,o_{\alpha'}\},\mu) \overset{def}{=} \\ \quad \exists\mu_1,\mu_2\,(\mathit{stochastic}(\{\mathit{moveS}(\alpha,m)\},s,\{a_\alpha\},\{o_\alpha\},\mu_1)\,\wedge \\ \qquad\qquad \mathit{stochastic}(\{\mathit{moveS}(\alpha',m')\},s,\{a_{\alpha'}\},\{o_{\alpha'}\},\mu_2) \wedge \mu=\mu_1\cdot\mu_2)\,. \end{array}$$

Here, $\mathit{visible}(\alpha, \alpha', a, s)$ is true if α can observe α' after the execution of a in s. The stochastic action $\mathit{moveS}(\alpha, m)$ is associated with the observations $\mathit{obs}(\beta, \mathit{out})$, where $\beta \in \{\alpha', \mathit{nil}\}$ and $r \in \{\mathit{success}, \mathit{failure}\}$. That is, after the execution of the action $\mathit{move}(\alpha, m)$, 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 $\mathit{obs}(\alpha', \mathit{out})$ 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:

$$\begin{aligned}&\mathit{reward}(\alpha, c, s) = r \overset{def}{=} \exists \alpha' (\mathit{goal}(\alpha', \mathit{do}(c, s)) \wedge (\alpha' {=} \alpha \wedge r = M \vee\\&\quad \mathit{sameTeam}(\alpha, \alpha') \wedge r = M' \vee \neg \mathit{sameTeam}(\alpha', \alpha) \wedge r = -M)) \vee\\&\quad \neg \exists \alpha' \, (\mathit{goal}(\alpha', \mathit{do}(c, s)) \wedge \mathit{evalTeamPos}(\alpha, c, r, s)) \,.\end{aligned}$$

Here, the reward of agent α is very high (that is, M stands for a "big" integer) if α itself scores a goal, and a bit lower (that is, $M' < M$) if the goal is scored by a team-mate. Otherwise, the reward depends on $\mathit{evalTeamPos}(\alpha, c, r, s)$, that is, the position of its team relative to the adversary team as well as the ball possession. Here, the predicate $\mathit{goal}(\alpha, s)$ is defined as follows:

$$\mathit{goal}(\alpha, s) \overset{def}{=} \exists x, y (\mathit{haveBall}(\alpha, s) \wedge \mathit{at}(\alpha, x, y, s) \wedge \mathit{posGoal}(\alpha, x, y)) \,,$$

where $\mathit{posGoal}(\alpha, x, y)$ is true iff (x, y) are the goal coordinates of the adversary team of α. The predicate $\mathit{evalTeamPos}(c, r, s)$ is defined as follows:

$$\begin{aligned}&\mathit{evalTeamPos}(\alpha, c, r, s) \overset{def}{=} \exists \alpha', r' ((\mathit{haveBall}(\alpha', \mathit{do}(c, s)) \wedge \mathit{evalPos}(\alpha', r', s) \wedge\\&\quad (\mathit{sameTeam}(\alpha, \alpha') \wedge r = r' \vee \neg \mathit{sameTeam}(\alpha', \alpha) \wedge r = -\, r')) \,.\end{aligned}$$

Informally, the value r in $\mathit{evalTeamPos}(\alpha, c, r, s)$ depends on the ball owner position $\mathit{evalPos}(\alpha, r, s)$ relative to the goal, the team-mates, and the adversary team.

3.2 Belief States

We now introduce belief states along with the executability of actions in belief states and the semantics of actions in terms of transitions between belief states. We also describe how the initial belief state of the agents can be encoded.

A *belief state (over situations)* has the form $b = (b_i)_{i \in I}$, where every b_i is a finite set of pairs (s, μ) consisting of a 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 formula $\phi(s)$ that is uniform in s in the belief state $b = (b_i)_{i \in I}$, 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, $\mathit{reward}(c, b)$ denotes the vector $r = (r_i)_{i \in I}$, where every r_i with $i \in I$ is the sum of all $\mathit{reward}(i, c, s) \cdot \mu$ such that $(s, \mu) \in b_i$.

Example 3.5 *(Rugby Domain cont'd)* Consider the following scenario relative to the domain theory of Example 3.4 (see Fig. 2). We focus 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 $b_{\boldsymbol{a}_1}$ coincides with the state shown in Fig. 2, upper part: There is only the situation s_1 with the probability 1 such that $\mathit{at}(\boldsymbol{a}_1, 2, 1, s_1)$, $\mathit{at}(\boldsymbol{a}_2, 2, 4, s_1)$, $\mathit{at}(\boldsymbol{o}_2, 1, 1, s_1)$, $\mathit{at}(\boldsymbol{o}_1, 5, 2, s_1)$, and $\mathit{haveBall}(\boldsymbol{a}_1, s_1)$ are all

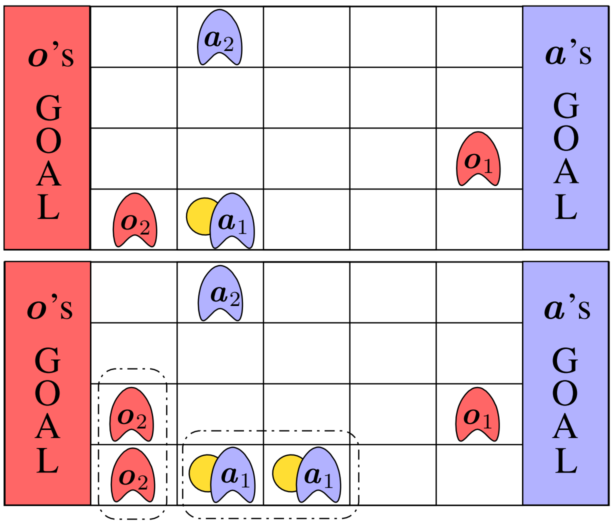

Figure 2: Rugby Domain: Belief states of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively.

true. That is, the captain $\boldsymbol{o}_1$ of $\boldsymbol{o}$ is very close to the goal of $\boldsymbol{a}$. From the perspective of $\boldsymbol{a}_1$, its team can score a goal as follows: $\boldsymbol{a}_1$ can pass to $\boldsymbol{a}_2$, which has a paved way towards the goal. Unfortunately, $\boldsymbol{a}_1$ has to cooperate with $\boldsymbol{a}_2$, whose vision of the situation is more confused (see Fig. 2, lower part): From $\boldsymbol{a}_2$'s point of view (that is, belief state $b_{\boldsymbol{a}_2}$), $\boldsymbol{o}_2$ could be at either (a) $(1,1)$ or (b) $(1,2)$, and $\boldsymbol{a}_1$ could be at either (c) $(2,1)$ or (d) $(3,1)$. Hence, $\boldsymbol{a}_2$'s belief state $b_{\boldsymbol{a}_2}$ consists of four possible states with, for example, the following probability distribution: $\{(s_{a,c}, 0.5), (s_{a,d}, 0.3), (s_{b,c}, 0.1), (s_{b,d}, 0.1)\}$.

A deterministic action c is *executable* in a belief state $b = (b_i)_{i \in I}$ iff $Poss(c, b) > 0$ (that is, c is executable in a situation s such that $(s, \mu) \in b_i$ for some $i \in I$). A stochastic action c is *executable* in a belief state $b = (b_i)_{i \in I}$ producing the observation $o \in O_c$ iff $Poss(c_o, b) > 0$ (that is, c is executable in a situation s such that $(s, \mu) \in b_i$ for some $i \in I$ and $prob(c, s, n, o) > 0$ for some n).

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 $o \in O_c$, 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 making the observation $o \in O_c$ after the execution of the stochastic action c in $b = (b_i)_{i \in I}$, denoted $prob(c, b, o)$, is the vector $pr = (pr_i)_{i \in I}$, where every probability 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 3.6 *(Rugby Domain cont'd)* The successor belief state after executing the stochastic action $c = \{moveS(\boldsymbol{a}_1, m), moveS(\boldsymbol{a}_2, m)\}$ in the belief state $b = (\{(S_0, 1)\}, \{(S_0, 1)\})$ and jointly observing the failure resp. success of the action of agent $\boldsymbol{a}_1$ resp. $\boldsymbol{a}_2$ (that is, $obs_{\boldsymbol{a}_1}(failure)$ resp. $obs_{\boldsymbol{a}_2}(success)$) is given by the following $b' = (b'_{\boldsymbol{a}_1}, b'_{\boldsymbol{a}_2})$ (where $c_{i,j} = c_i \cup c'_j$ with $c_0 = \{moveTo(\boldsymbol{a}_1, stand)\}$, $c_1 = \{moveTo(\boldsymbol{a}_1,$

$m)\}$, $c'_0 = \{moveTo(\boldsymbol{a}_2, stand)\}$, and $c'_1 = \{moveTo(\boldsymbol{a}_2, m)\}$, and the probabilities follow from the stochastic theory ST in Example 3.4):

$$\begin{aligned} b'_{\boldsymbol{a}_1} = b'_{\boldsymbol{a}_2} = \{&(do(c_{0,0}, S_0), \tfrac{0.09}{0.1+0.09} \cdot \tfrac{0.01}{0.8+0.01}), (do(c_{1,0}, S_0), \tfrac{0.1}{0.1+0.09} \cdot \tfrac{0.01}{0.8+0.01}), \\ &(do(c_{0,1}, S_0), \tfrac{0.09}{0.1+0.09} \cdot \tfrac{0.8}{0.8+0.01}), (do(c_{1,1}, S_0), \tfrac{0.1}{0.1+0.09} \cdot \tfrac{0.8}{0.8+0.01})\}\,. \end{aligned}$$

We next describe how the initial belief state of the agents can be encoded. Let $b = (b_i)_{i \in I}$ be an initial belief state with $b_i = \{(s_{i,j}, \mu_{i,j}) \mid j \in \{1, \ldots, n_i\}\}$ for all $i \in I$. For every $i \in I$ and $j \in \{1, \ldots, n_i\}$, we assume a deterministic action $g_{i,j}$, which performs a transition from S_0 into the situation $s_{i,j}$. For every $i \in I$, we then generate b_i by a stochastic action g_i, which has every $g_{i,j}$ (along with the probability $\mu_{i,j}$) such that $j \in \{1, \ldots, n_i\}$ as a deterministic component, and we generate b by the multi-agent stochastic action $g = \{g_i \mid i \in I\}$. Since the deterministic actions $g_{i,j}$ generate only the possible situations $s_{i,j}$ in the initial belief state, we assume the precondition axiom $Poss(g_{i,j}, s) \equiv s = S_0$. We also want that the other deterministic actions are only executable after S_0. Hence, for each primitive action, the precondition axiom $Poss(a, s) \equiv \Psi(a, s)$ is slightly rewritten as $Poss(a, s) \equiv \Psi(a, s) \wedge s \neq S_0$. Finally, instead of using $\mathcal{D}_{S_0}$ to specify the initial situation S_0, we use it to describe each of the possible situations in the initial belief state.

Example 3.7 *(Rugby Domain cont'd)* Consider again the belief state $b = (b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2})$ described in Example 3.4, which is given by $b_{\boldsymbol{a}_1} = \{(s_1, 1)\}$ and $b_{\boldsymbol{a}_2} = \{(s_{a,c}, 0.5), (s_{a,d}, 0.3), (s_{b,c}, 0.1), (s_{b,d}, 0.1)\}$. In order to encode $b_{\boldsymbol{a}_2}$, we use the deterministic actions $g_{2;a,c}$, $g_{2;a,d}$, $g_{2;b,c}$, and $g_{2;b,d}$ to generate all possible situations in $b_{\boldsymbol{a}_2}$:

$$\begin{aligned} &s_{a,c} = do(g_{2;a,c}, S_0),\ \ s_{a,d} = do(g_{2;a,d}, S_0), \\ &s_{b,c} = do(g_{2;b,c}, S_0),\ \ s_{b,d} = do(g_{2;b,d}, S_0)\,, \end{aligned}$$

and we use the stochastic action g_2 to associate them with their probabilities in $b_{\boldsymbol{a}_2}$:

$$\begin{aligned} &stochastic(\{g_2\}, s, \{a\}, \{o\}, \mu) \overset{def}{=} \\ &\quad a = g_{2;a,c} \wedge \mu = 0.5 \vee a = g_{2;a,d} \wedge \mu = 0.3 \vee \\ &\quad a = g_{2;b,c} \wedge \mu = 0.1 \vee a = g_{2;b,d} \wedge \mu = 0.1\,. \end{aligned}$$

Assuming a similar stochastic action g_1 for $b_{\boldsymbol{a}_1}$ (associated with a unique deterministic action $g_{1,1}$ such that $s_1 = do(g_{1,1}, S_0)$), the belief state $b = (b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2})$ is generated by executing the multi-agent stochastic action $g = \{g_1, g_2\}$ in S_0:

$$\begin{aligned} &stochastic(\{g_1, g_2\}, s, \{a, b\}, \{o_1, o_2\}, \mu) \overset{def}{=} \\ &\quad \exists \mu_1, \mu_2 (stochastic(\{g_1\}, s, \{a\}, \{o_1\}, \mu_1) \wedge \\ &\qquad\qquad stochastic(\{g_2\}, s, \{b\}, \{o_2\}, \mu_2) \wedge \mu = \mu_1 \times \mu_2). \end{aligned}$$

Notice that the properties of the possible situations in a belief state can be defined as usual. For example, the properties of the ones in $b_{\boldsymbol{a}_2}$ can be defined by:

$$\begin{aligned} &at(\boldsymbol{a}_1, 2, 1, s_{a,c}) \wedge at(\boldsymbol{o}_2, 1, 1, s_{a,c}),\ \ at(\boldsymbol{a}_1, 3, 1, s_{a,d}) \wedge at(\boldsymbol{o}_2, 1, 1, s_{a,d}), \\ &at(\boldsymbol{a}_1, 2, 1, s_{b,c}) \wedge at(\boldsymbol{o}_2, 1, 2, s_{b,c}),\ \ at(\boldsymbol{a}_1, 3, 1, s_{b,d}) \wedge at(\boldsymbol{o}_2, 1, 2, s_{b,d})\,. \end{aligned}$$

3.3 Syntax of POGTGolog

In the sequel, let DT be a domain theory. We define POGTGolog by induction as follows. A *program* p in POGTGolog has one of the following forms (where α is a multi-agent action or the empty action nop (which is always executable and does not change the state of the world), ϕ is a condition, $p, p_1, p_2, \ldots, p_n$ are programs without procedure declarations, $P_1, \ldots, P_n$ are procedure names, $x, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are arguments, $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ is a nonempty finite set of ground terms, $a_{i,1}, \ldots, a_{i,n_i}$ are single-agent actions of agent $i \in I$, and $J \subseteq I$ with $|J| \geqslant 2$):

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 program argument:* $\pi[x : \tau]\ (p(x))$. Do any $p(\tau_i)$.
8. *Nondeterministic iteration*: $p^\star$. Do p zero or more times.
9. *Conditional*: **if** ϕ **then** p_1 **else** p_2.
10. *While-loop*: **while** ϕ **do** p.
11. *Procedures*: **proc** $P_1(\boldsymbol{x}_1)\ p_1$ **end** ; . . . ; **proc** $P_n(\boldsymbol{x}_n)\ p_n$ **end** ; p.

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 (that is, $[[p_1; p_2]; p_3]$ and $[p_1; [p_2; p_3]]$ have the same meaning), and we often use "$p_1; p_2$" to abbreviate "$[p_1; p_2]$".

Example 3.8 *(Rugby Domain cont'd)* Consider again the scenario (and its belief states $b_{\boldsymbol{a}_1}$ and $b_{\boldsymbol{a}_2}$) of Example 3.5 relative to the domain theory of Example 3.4 (see Fig. 2). 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 $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ have two possible chances to coordinate themselves to pass the ball; thereafter, both of them have to run towards the goal (with or without the ball):

```
proc $schema()$
choice($\boldsymbol{a}_1$: $moveS(\boldsymbol{a}_1, E) \mid moveS(\boldsymbol{a}_1, stand) \mid moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2))$) $\|$
    choice($\boldsymbol{a}_2$: $moveS(\boldsymbol{a}_2, S) \mid moveS(\boldsymbol{a}_2, E) \mid moveS(\boldsymbol{a}_2, receive(\boldsymbol{a}_1))$);
choice($\boldsymbol{a}_1$: $moveS(\boldsymbol{a}_1, E) \mid moveS(\boldsymbol{a}_1, stand) \mid moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2))$) $\|$
    choice($\boldsymbol{a}_2$: $moveS(\boldsymbol{a}_2, E) \mid moveS(\boldsymbol{a}_2, receive(\boldsymbol{a}_1))$);
$\{moveS(\boldsymbol{a}_1, E), moveS(\boldsymbol{a}_2, E)\}$;
$\{moveS(\boldsymbol{a}_1, E), moveS(\boldsymbol{a}_2, E)\}$
end.
```

3.4 Policies and Nash Equilibria of POGTGolog

We now define the formal semantics of POGTGolog programs p relative to a domain theory DT in terms of Nash equilibria of p, which are optimal finite-horizon policies of p. We first associate with every POGT-Golog program p, belief state b, and horizon $H \geqslant 0$, a set of H-step policies π along with their expected utility U_i to every agent $i \in I$. We then define the notion of an H-step 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 $H \geqslant 0$, an H-step policy π of a POGTGolog program p in a belief state b relative to a domain theory DT 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. Every such H-step policy π is associated with an expected H-step reward to $i \in I$, an H-step success probability (which is the probability that π is executable in b), and an expected H-step utility to $i \in I$ (which is computed from the expected H-step reward and the H-step success probability using the utility function).

Formally, for every POGTGolog program p, the *nil-terminated variant* of p, denoted $\widehat{p}$, is inductively defined 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 $H \geqslant 0$, and a start belief state b, we say that π is an *H-step policy* of p in b relative to DT with *expected H-step reward v_i*, *H-step success probability pr_i*, and *expected H-step utility* $U_i(H, b, \pi) = utility(v_i, pr_i)$ to agent $i \in I$ iff $DT \models G(\widehat{p}, b, H, \pi, (v_i)_{i \in I}, (pr_i)_{i \in I})$, where the macro $G(\widehat{p}, b, h, \pi, v, pr)$, for every number of steps to go $h \in \{0, \ldots, H\}$, is defined by induction as follows ($\widehat{p}$, b, and h are the input values of G, while π, $v = (v_i)_{i \in I}$, and $pr = (pr_i)_{i \in I}$ are the output values of G):

- *Null program* or *zero horizon*:
  If $\widehat{p} = nil$ or $h = 0$, then:

  $$G(\widehat{p}, b, h, \pi, v, pr) \stackrel{def}{=} \pi = nil \wedge v = \mathbf{0} \wedge pr = \mathbf{1}\,.$$

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

- *Deterministic first program action (resp., stochastic first program action with observation):* If $\widehat{p} = [c\,; p']$, where c is a deterministic action (resp., stochastic action with observation), and $h > 0$, then:

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

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 $pr = \mathbf{0}$. Here, $stop$ is a zero-cost action, which takes the agents to an absorbing state, where they stop the execution of the policy and wait for further instructions. 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 (choice of nature)*:
  If $\widehat{p} = [c\,; p']$, where c is a stochastic action, and $h > 0$, then:

$$\begin{aligned}
&G([c\,; p'], b, h, \pi, v, pr) \overset{def}{=} \\
&\quad \exists l, \pi_1, \ldots, \pi_l, v_1, \ldots, v_l, pr_1, \ldots, pr_l\, (\textstyle\bigwedge_{q=1}^{l} G([c_{o_q}; p'], b, h, c_{o_q}; \pi_q, \\
&\qquad v_q, pr_q) \wedge \pi = c\,; \textbf{for } q = 1 \textbf{ to } l \textbf{ do if } o_q \textbf{ then } \pi_q \wedge \\
&\qquad v = \textstyle\sum_{q=1}^{l} v_q \cdot prob(c, b, o_q) \wedge pr = \sum_{q=1}^{l} pr_q \cdot prob(c, b, o_q))\,.
\end{aligned}$$

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

- *Nondeterministic first program action (choice of agent $i \in I$)*:
  If $\widehat{p} = [\textbf{choice}(i\colon a_1 | \cdots | a_n)\,; p']$ and $h > 0$, then:

$$\begin{aligned}
&G([\textbf{choice}(i\colon a_1 | \cdots | a_n)\,; p'], b, h, \pi, v, pr) \overset{def}{=} \\
&\quad \exists \pi_1, \ldots, \pi_n, v_1, \ldots, v_n, pr_1, \ldots, pr_n, k\, (\textstyle\bigwedge_{q=1}^{n} G([a_q\,; p'], b, h, a_q\,; \pi_q, \\
&\qquad v_q, pr_q) \wedge k \in \{1, \ldots, n\} \wedge \pi = a_k\,; \textbf{for } q = 1 \textbf{ to } n \textbf{ do if } \psi_q \textbf{ then } \pi_q \;\wedge \\
&\qquad v = v_k \wedge pr = pr_k)\,.
\end{aligned}$$

  Informally, every policy π of p consists of any action a_k and one policy π_q of p' for every possible action a_q. The expected reward and the success probability of π are given by the expected reward v_q and the success probability pr_q of π_q. For the other agents to observe which action among $a_1, \ldots, a_n$ was actually executed by agent i, we use a cascade of if-then-else statements with conditions ψ_q.

- *Nondeterministic first program action (joint choice of the agents in J)*:
  If $\widehat{p} = [\,\|_{j \in J}\textbf{choice}(j\colon a_{j,1} | \cdots | a_{j,n_j})\,; p']$ and $h > 0$, then:

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

  Here, $A = \times_{j \in J} \{a_{j,1}, \ldots, a_{j,n_j}\}$. We denote by $PD(S)$ the set of all probability distributions over S, and $(\Pi_{j \in J}\pi_j)(a) = \Pi_{j \in J}\pi_j(a_j)$ for all $a = (a_j)_{j \in J}$. Informally, every policy π of p consists of one probability distribution π_j over $\{a_{j,1}, \ldots, a_{j,n_j}\}$ for every agent $j \in J$, and one policy π_a of p' for every possible joint action $a \in A$. The expected reward and the success probability of π are given by the expected reward and the expected success probability of the policies π_a. Here, π_j specifies the probabilities with which agent $j \in J$ should execute the actions $\{a_{j,1}, \ldots, a_{j,n_j}\}$. Hence, assuming the usual probabilistic independence between the distributions π_j with $j \in J$ in stochastic games, every possible joint action a is executed with the probability $(\Pi_{j \in J} \pi_j)(a)$. Note that the conditions ϕ_a with $a \in A$ are to observe what the agents have actually executed.

- *Test action*:
  If $\widehat{p} = [\phi? \,; p']$ and $h > 0$, then:

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

  Informally, let $\widehat{p} = [\phi? \,; p']$. If ϕ is false in b, then p has only the policy $\pi = \mathit{stop}$, 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'.

- *Nondeterministic choice of two programs*:
  If $\widehat{p} = [(p_1 \mid p_2); p']$ and $h > 0$, then:

$$G([(p_1 \mid p_2); p'], b, h, \pi, v, pr) \stackrel{def}{=} \\ \exists \pi_1, \pi_2, v_1, v_2, pr_1, pr_2, k \, (\bigwedge_{q \in \{1,2\}} G([p_q; p'], b, h, \pi_q, v_q, pr_q) \wedge \\ k \in \{1, 2\} \wedge \pi = \pi_k \wedge v = v_k \wedge pr = pr_k) \,.$$

- *Conditional*:
  If $\widehat{p} = [\textbf{if } \phi \textbf{ then } p_1 \textbf{ else } p_2; p']$ and $h > 0$, then:

$$G([\textbf{if } \phi \textbf{ then } p_1 \textbf{ else } p_2; p'], b, h, \pi, v, pr) \stackrel{def}{=} \\ G([([\phi?; p_1] \mid [\neg\phi?; p_2]); p'], b, h, \pi, v, pr) \,.$$

  This case is reduced to test action and nondeterministic choice of two programs.

- *While-loop*:
  If $\widehat{p} = [\textbf{while } \phi \textbf{ do } p; p']$ and $h > 0$, then:

$$G([\textbf{while } \phi \textbf{ do } p; p'], b, h, \pi, v, pr) \stackrel{def}{=} G([[\phi?; p]^\star; \neg\phi?], b, h, \pi, v, pr) \,.$$

  This case is reduced to test action and nondeterministic iteration.

- *Nondeterministic choice of program argument*:
  If $\widehat{p} = [\pi[x : \tau](p(x)); p']$, where $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, and $h > 0$, then:

$$G([\pi[x : \tau](p(x)); p'], b, h, \pi, v, pr) \stackrel{def}{=} \\ G([(\cdots (p(\tau_1) | p(\tau_2)) | \cdots | p(\tau_n)); p'], b, h, \pi, v, pr) \,.$$

  This case is reduced to nondeterministic choice of two programs.

- *Nondeterministic iteration*:
  If $\widehat{p} = [p^\star; p']$ and $h > 0$, then:

  $$G([p^\star; p'], b, h, \pi, v, pr) \overset{def}{=} \\ G([[\textbf{proc}\ nit\ (nop \mid [p\, ; nit])\ \textbf{end};\ nit]; p'], b, h, \pi, v, pr)\,.$$

  This case is reduced to procedures and nondeterministic choice of two programs.

- *Procedures:* We consider the two cases of (1) handling procedure declarations and (2) handling procedure calls. To this end, we slightly extend the first argument of G by a store for procedure declarations, which can be safely ignored in all the above constructs of POGTGolog.

  (1) If $\widehat{p} = [\textbf{proc}\ P_1(\boldsymbol{x}_1)\ p_1\ \textbf{end}\, ; \ldots ; \textbf{proc}\ P_n(\boldsymbol{x}_n)\ p_n\ \textbf{end}\, ; p]\langle\rangle$ and $h > 0$, then:

  $$G([\textbf{proc}\ P_1(\boldsymbol{x}_1)\ p_1\ \textbf{end}\, ; \ldots ; \textbf{proc}\ P_n(\boldsymbol{x}_n)\ p_n\ \textbf{end}\, ; p]\langle\rangle, b, h, \pi, v, pr) \overset{def}{=} \\ G([p]\langle\textbf{proc}\ P_1(\boldsymbol{x}_1)\ p_1\ \textbf{end}\, ; \ldots ; \textbf{proc}\ P_n(\boldsymbol{x}_n)\ p_n\ \textbf{end}\rangle, b, h, \pi, v, pr)\,.$$

  We store the procedure declarations at the end of the first argument of G.

  (2) If $\widehat{p} = [P_i(\boldsymbol{x}_i); p']\langle d\rangle$ and $h > 0$, then:

  $$G([P_i(\boldsymbol{x}_i); p']\langle d\rangle, b, h, \pi, v, pr) \overset{def}{=} G([p_d(P_i(\boldsymbol{x}_i)); p']\langle d\rangle, b, h, \pi, v, pr)\,.$$

  We replace a procedure call $P_i(\boldsymbol{x}_i)$ by its code $p_d(P_i(\boldsymbol{x}_i))$ from d.

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

Example 3.9 *(Rugby Domain cont'd)* Consider again the scenario (and its belief states $b_{\boldsymbol{a}_1}$ and $b_{\boldsymbol{a}_2}$) of Example 3.5 relative to the domain theory of Example 3.4 (see Fig. 2). Assuming the horizon $H = 4$, a 4-step policy π of the POGTGolog program $schema$ of Example 3.8 is given by:

$$DT \models G([schema; nil], (b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2}), 4, \pi, (v_1, v_2), (pr_1, pr_2)).$$

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 (pure) 4-step policy $\pi_{\boldsymbol{a}_1} = c\, ; \pi^1_{\boldsymbol{a}_1}$, where $c = \{moveS(\boldsymbol{a}_1, passTo(\boldsymbol{a}_2)), moveS(\boldsymbol{a}_2, receive(\boldsymbol{a}_1))\}$, 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 $\pi^1_{\boldsymbol{a}_1}$ 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:

$$\begin{aligned}\pi_{\boldsymbol{a}_2} = c\, ;\ &\textbf{if}\ obs(\boldsymbol{a}_1, success)\ \textbf{then}\ \pi^{1,o_1}_{\boldsymbol{a}_2}\\ &\quad \textbf{else if}\ obs(\boldsymbol{a}_1, failure)\ \textbf{then}\ \pi^{1,o_2}_{\boldsymbol{a}_2}\\ &\qquad \textbf{else if}\ obs(nil, success)\ \textbf{then}\ \pi^{2,o_3}_{\boldsymbol{a}_2}\\ &\qquad\quad \textbf{else if}\ obs(nil, failure)\ \textbf{then}\ \pi^{2,o_4}_{\boldsymbol{a}_2}\ ,\end{aligned}$$

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 executing c in $(b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2})$. 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 first define an interpreter for POGTGolog programs, and we then provide some optimality and faithfulness results for the interpreter.

4.1 Formal Specification

We now define an interpreter for POGTGolog programs p relative to a domain theory DT by specifying the macro $OptG(\widehat{p}, b, H, \pi, v, pr)$, 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 $H \geqslant 0$, and which computes as output an optimal H-step policy π (one among all the H-step Nash equilibria of p in b; see Theorem 4.1) and the vectors $v = (v_i)_{i \in I}$ and $pr = (pr_i)_{i \in I}$, where each v_i is the expected H-step reward of π in b to i, each $pr_i \in [0, 1]$ is the H-step success probability of π in b for i, and $utility(v_i, pr_i)$ is the expected H-step utility of π in b to i. Informally, the macro G of Section 3.4 defines all the legal H-step policies of a POGTGolog program p, while the macro $OptG$ defines an optimal such legal one. Therefore, we define the macro $OptG(\widehat{p}, b, h, \pi, v, pr)$ in nearly the same way as the macro $G(\widehat{p}, b, h, \pi, v, pr)$ in Section 3.4, except for the following modifications:

- *Nondeterministic first program action (choice of agent $i \in I$)*: The definition of $OptG$ 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 definition of $OptG$ 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 "$(\pi_j)_{j \in J} = select\text{-}Nash(\{utility(v_a, pr_a)|_J \mid a \in A\})$", 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 definition of $OptG$ 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 $utility(v_{q,j}, pr_{q,j})$.

4.2 Optimality and Faithfulness

The following theorem shows the important result that the macro $OptG$ is optimal in the sense that, for every horizon $H \geqslant 0$, 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, let $H \geqslant 0$ be a horizon, and let $DT \models OptG(\widehat{p}, b, H, \pi, v, pr)$. Then, π is* *an H-step Nash equilibrium of p in b relative to DT, and $\mathit{utility}(v_i, pr_i)$ is its expected H-step utility to agent $i \in I$.*

The following theorem shows that POGTGolog programs faithfully extend POSGs, that is, in the special case where they syntactically model POSGs, they are also semantically interpreted as POSGs. Thus, POGTGolog programs have a nice semantic behavior in such special cases. Formally, the theorem says that given any $H \geqslant 0$, every POSG can be encoded as a program p in POGTGolog such that OptG specifies 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, and let $H \geqslant 0$ be a horizon. Then, there exists a domain theory $DT = (AT, ST, OT)$, and a set of POGTGolog programs $\{p^h \mid h \in \{0, \ldots, H\}\}$ relative to DT such that $\sigma = (\sigma_i)_{i \in I}$ is an H-step Nash equilibrium of G, where every $(\sigma_i(b, h))_{i \in I} = (\pi_i)_{i \in I}$ is given by $DT \models \mathit{OptG}(\widehat{p}^h, B_b, h{+}1, \Pi_{i \in I} \pi_i \,; \pi', v, pr)$, for every belief state b of G and every $h \in \{0, \ldots, H\}$, with B_b being a belief state of DT associated with b. Furthermore, the expected H-step reward of σ in b to agent $i \in I$ is given by $\mathit{utility}(v_i, pr_i)$, where $DT \models \mathit{OptG}(\widehat{p}^H, B_b, H{+}1, \pi, v, pr)$, for every belief state b of G.*

5 Example

In this section, we provide an extended example to illustrate the overall framework at work. This example is inspired by the stratagus domain in [28].

Example 5.1 *(Stratagus Domain)* The stratagus field consists of 9×9 positions (see Fig. 3). We assume a team of two agents $\boldsymbol{a} = \{\boldsymbol{a}_1, \boldsymbol{a}_2\}$, which occupy one position each. The stratagus field has designated areas representing two *gold-mines*, one *forest*, and one *base* for each agent (see Fig. 3). The two agents can move one step in one of the directions *N*, *S*, *E*, and *W*, or remain stationary. Each of the two agents can also pick up one unit of wood (resp., gold) at the forest (resp., gold-mines), and drop these resources at the base. We assume that if $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are in the same location, then they cannot pick up anything. Each action of the two agents can fail, resulting in a stationary move. Any carried object drops when the two agents collide. After each step, the agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ receive the rewards $r_{\boldsymbol{a}_1}$ and $r_{\boldsymbol{a}_2}$, respectively, where r_k for $k \in \{\boldsymbol{a}_1, \boldsymbol{a}_2\}$ is 0, 1, and 2 when k brings nothing, one unit of wood, and one unit of gold to its base, respectively.

We define the domain theory $DT = (AT, ST, OT)$ as follows. As for the basic action theory AT, we assume the deterministic actions $\mathit{move}(\alpha, m)$ (agent α performs m among N, S, E, W, and stand), $\mathit{pickUp}(\alpha, o)$ (agent α picks up the object o), and $\mathit{drop}(\alpha, o)$ (agent α drops the object o), as well as the relational fluents $at(q, x, y, s)$ (agent or object q is at the position (x, y) in the situation s), and $\mathit{holds}(\alpha, o, s)$ (agent α holds the object o in the situation s), which are defined through the following successor state axioms:

$$
\begin{aligned}
& at(q, x, y, do(c, s)) \equiv agent(q) \wedge (at(q, x, y, s) \wedge move(q, stand) \in c \,\vee \\
& \quad \exists x', y'(at(q, x', y', s) \wedge \exists m(move(\alpha, m) \in c \wedge \phi(x, y, x', y', m)))) \,\vee \\
& \quad object(q) \wedge (at(q, x, y, s) \wedge \neg \exists \alpha(pickUp(\alpha, q) \in c) \,\vee \\
& \quad \exists \alpha((drop(\alpha, q) \in c \vee collision(c, s)) \wedge at(\alpha, x, y, s) \wedge holds(\alpha, q, s)))\,; \\
& holds(\alpha, o, do(c, s)) \equiv holds(\alpha, o, s) \wedge drop(\alpha, o) \notin c \,\wedge \\
& \quad \neg collision(c, s) \vee pickUp(\alpha, o) \in c.
\end{aligned}
$$

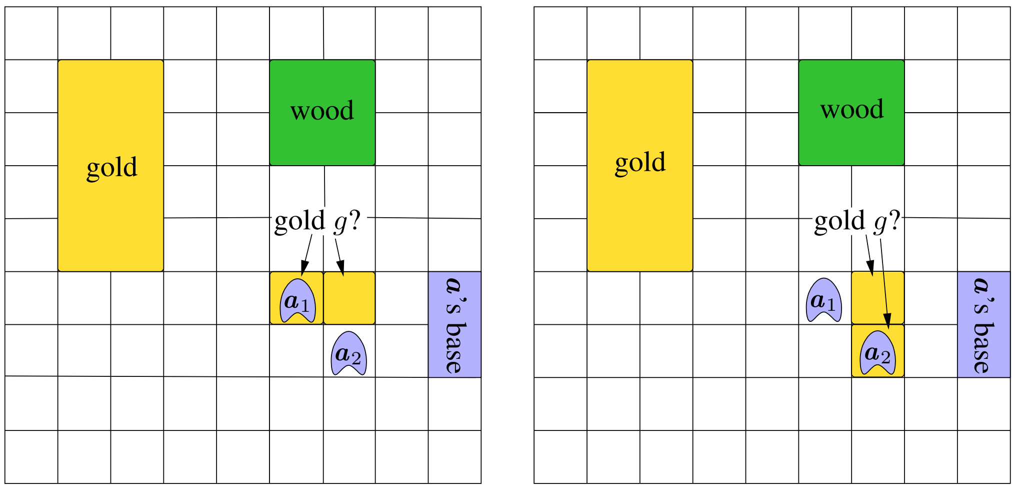

Figure 3: Stratagus Domain: Initial belief states of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively.

Here, $\phi(x, y, x', y', m)$ is true iff the coordinates change from the position (x', y') to the position (x, y) due to $m \in \{N, S, E, W, \mathit{stand}\}$, that is,

$$\begin{aligned}&\phi(x, y, x', y', m) \overset{def}{=} (m \notin \{N, S, E, W\} \wedge x = x' \wedge y = y') \vee \\ &\quad (m = N \wedge x = x' \wedge y = y'{+}1) \vee (m = S \wedge x = x' \wedge y = y'{-}1) \vee \\ &\quad (m = E \wedge x = x'{+}1 \wedge y = y') \vee (m = W \wedge x = x'{-}1 \wedge y = y')\,,\end{aligned}$$

and $\mathit{collision}(c, s)$ encodes that executing the action c in the situation s causes a collision between the agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in the situation $do(c, s)$, that is,

$$\begin{aligned}&\mathit{collision}(c, s) \overset{def}{=} \exists \alpha, \beta, x, y\ (\alpha \neq \beta \wedge \\ &\quad \exists x', y' (at(\alpha, x', y', s) \wedge \exists m\ (move(\alpha, m) \in c \wedge \phi(x, y, x', y', m))) \wedge \\ &\quad \exists x'', y'' (at(\beta, x'', y'', s) \wedge \exists m\ (move(\beta, m) \in c \wedge \phi(x, y, x'', y'', m))) \wedge \\ &\qquad (x' \neq x \vee y' \neq y \vee x'' \neq x \vee y'' \neq y))\,.\end{aligned}$$

That is, we have a collision between two agents iff (i) at least one them moves, and (ii) they are in the same location thereafter. The deterministic actions $move(\alpha, m)$, $drop(\alpha, o)$, and $\mathit{pickUp}(\alpha, o)$ have the following precondition axioms:

$$\begin{aligned}&Poss(move(\alpha, m), s) \equiv \neg\exists x, y\ (at(\alpha, x, y, s) \wedge ((y = 9 \wedge m = N) \vee \\ &\quad (y = 1 \wedge m = S) \vee (x = 9 \wedge m = E) \vee (x = 1 \wedge m = W)))\,; \\ &Poss(drop(\alpha, o), s) \equiv holds(\alpha, o, s)\,; \\ &Poss(\mathit{pickUp}(\alpha, o), s) \equiv \neg\exists x\ holds(\alpha, x, s) \wedge \exists x, y (at(\alpha, x, y, s) \wedge at(o, x, y, s))\,.\end{aligned}$$

Here, the first axiom forbids α to go out of the 9×9 game-field, while the second axiom states that α can only drop the object o if α is holding o, and the third axiom permits α to pick up o if α is at same location and not holding anything else.

As for the stochastic theory ST, we assume the stochastic actions $moveS(\alpha, m)$, $\mathit{pickUpS}(\alpha, o)$, and $dropS(\alpha, o)$, which are specified below. Each execution of such an action c is followed by an observation among $obs_\alpha(success)$ and $obs_\alpha(failure)$. Intuitively, α can observe if the execution of α was successful or not.

$$\begin{aligned}
&stochastic(\{moveS(\alpha,m)\},s,\{a\},\{o\},\mu) \overset{def}{=} o = obs_\alpha(success) \wedge \\
&\quad (a = move(\alpha,m) \wedge \mu = 0.54 \vee a = move(\alpha,stand) \wedge \mu = 0.36) \vee \\
&\quad o = obs_\alpha(failure) \wedge a = move(\alpha,stand) \wedge \mu = 0.1\,; \\
&stochastic(\{pickUpS(\alpha,o)\},s,\{a\},\{o\},\mu) \overset{def}{=} o = obs_\alpha(success) \wedge \\
&\quad (a = pickUp(\alpha,o) \wedge \mu = 0.72 \vee a = move(\alpha,stand) \wedge \mu = 0.18) \vee \\
&\quad o = obs_\alpha(failure) \wedge a = move(\alpha,stand) \wedge \mu = 0.1\,; \\
&stochastic(\{dropS(\alpha,o)\},s,\{a\},\{o\},\mu) \overset{def}{=} o = obs_\alpha(success) \wedge \\
&\quad (a = drop(\alpha,o) \wedge \mu = 0.81 \vee a = move(\alpha,stand) \wedge \mu = 0.09) \vee \\
&\quad o = obs_\alpha(failure) \wedge a = move(\alpha,stand) \wedge \mu = 0.1\,.
\end{aligned}$$

Observe that the observation $obs_\alpha(failure)$ is reliable, that is, if agent α observes $obs_\alpha(failure)$, then its executed action was not successful with the probability 1. Multi-agent stochastic actions are defined as follows (assuming independence):

$$\begin{aligned}
&stochastic(\{moveS(\alpha,m), moveS(\alpha',m')\},s,\{a_\alpha,a_{\alpha'}\},\{o_\alpha,o_{\alpha'}\},\mu) \overset{def}{=} \\
&\quad \exists \mu_1,\mu_2(stochastic(\{moveS(\alpha,m)\},s,\{a_\alpha\},\{o_\alpha\},\mu_1) \wedge \\
&\qquad\qquad stochastic(\{moveS(\alpha',m')\},s,\{a_{\alpha'}\},\{o_{\alpha'}\},\mu_2) \wedge \mu = \mu_1 \cdot \mu_2)\,.
\end{aligned}$$

As for the optimization theory OT, we use the product as the utility function *utility*. Furthermore, we define the reward function for agent α as follows:

$$\begin{aligned}
&reward(\alpha,c,s) = r \overset{def}{=} \exists r_\alpha, r_\beta(rewardAct(\alpha,c,s) = r_\alpha \wedge \\
&\quad rewardAct(\beta,c,s) = r_\beta \wedge \alpha \neq \beta \wedge r = r_\alpha + 0.5 \cdot r_\beta)\,.
\end{aligned}$$

Here, $rewardAct(\alpha,c,s)$ is defined as follows:

$$\begin{aligned}
&rewardAct(\alpha,c,s) = r \overset{def}{=} \exists o,x,y\,(at(\alpha,x,y,s) \wedge holding(\alpha,o,s) \wedge \\
&\quad \neg collision(c,s) \wedge base(\alpha,x,y) \ \wedge drop(\alpha,o) \in c \wedge (gold(o) \wedge r = 2 \vee \\
&\quad wood(o) \wedge r = 1)) \vee \neg\exists o,x,y\,(at(\alpha,x,y,s) \wedge holding(\alpha,o,s) \wedge \\
&\quad \neg collision(c,s) \wedge base(\alpha,x,y) \wedge drop(\alpha,o) \in c) \wedge r = 0\,.
\end{aligned}$$

Note that the reward to agent α is higher if α itself can drop an object to the base. Thus, even though there is a joint interest in bringing objects to the base, each agent prefers to be the one who achieves the goal.

Consider the scenario shown in Fig. 3, where the two agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are looking for a unit of gold g, and they are trying to bring it to their base. Suppose that the initial belief state of agent $\boldsymbol{a}_1$ (resp., $\boldsymbol{a}_2$) is as in Fig. 3, left (resp., right) side. In particular, agent $\boldsymbol{a}_1$ (resp., $\boldsymbol{a}_2$) initially believes that the unit of gold g is at either $(6,4)$ or $(7,4)$ (resp., $(7,3)$ or $(7,4)$). Formally, let the belief state of agent $\boldsymbol{a}_1$ (resp., $\boldsymbol{a}_2$) be given by $b_{\boldsymbol{a}_1} = \{(s_{6,4},0.2),(s_{7,4},0.8)\}$ (resp., $b_{\boldsymbol{a}_2} = \{(s_{7,3},0.4),(s_{7,4},0.6)\}$), where every included $s_{i,j}$ denotes one of the three possible situations shown in Fig. 3, and $at(g,i,j,s_{i,j})$ is true. How should the two agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ now act in such an initial situation? Both agents believe that the unit of gold g could be in their position or in the position $(7,4)$. Thus, the probability of finding the unit of gold g in the position $(7,4)$ is higher, but if both agents decide to go there, then they could block each other's pick-up actions, since joint pick-up actions in the same position are not allowed. However, once one of the two agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ has caught the unit of gold g, it should go to its base and drop it. The following POGTGolog program, denoted schema, encodes a possible way of acting of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in this scenario:

```
**proc** $\mathit{schema}(n)$
$\mathit{pickOrGo}(n)$;
$\pi\, \boldsymbol{a}, o\, (\mathit{holding}(\boldsymbol{a}, o)?\,;\, \mathit{carryToBase}(\boldsymbol{a}))$
**end**.
```

In this program, each agent tries to pick up an object, solving a "pick up or go" dilemma when they aim at picking up the same object (through $\mathit{pickOrGo}$). Once an object is gathered, the agent has to find a way to bring it to the base (through $\mathit{carryToBase}$). The procedure $\mathit{pickOrGo}(n)$ is defined as follows:

```
**proc** $\mathit{pickOrGo}(n)$
**if** $n > 0 \wedge \neg\exists\alpha, o\, (\mathit{holding}(\alpha, o))$ **then** [
    $\pi\, d_1, p_2, o_1, o_2\, ((\mathit{direction}(p_1) \wedge \mathit{direction}(p_2) \wedge \mathit{object}(o_1) \wedge \mathit{object}(o_2))?\,;$
        **choice**$(\boldsymbol{a}_1\colon \mathit{moveS}(\boldsymbol{a}_1, d_1) \mid \mathit{moveS}(\boldsymbol{a}_1, \mathit{stand}) \mid \mathit{pickUpS}(\boldsymbol{a}_1, o_1)) \parallel$
            **choice**$(\boldsymbol{a}_2\colon \mathit{moveS}(\boldsymbol{a}_2, d_2) \mid \mathit{moveS}(\boldsymbol{a}_2, \mathit{stand}) \mid \mathit{pickUpS}(\boldsymbol{a}_2, o_2)));$
    $\mathit{pickOrGo}(n{-}1)]$
**end**.
```

The following procedure $\mathit{carryToBase}(\boldsymbol{a})$ describes a partially specified behavior where the agent $\boldsymbol{a}$ is trying to move to its base to drop down an object:

```
**proc** $\mathit{carryToBase}(\boldsymbol{a})$
**choice**$(\boldsymbol{a}\colon \mathit{moveS}(\boldsymbol{a}, N) \mid \mathit{moveS}(\boldsymbol{a}, S) \mid \mathit{moveS}(\boldsymbol{a}, E) \mid \mathit{moveS}(\boldsymbol{a}, W));$
**if** $\mathit{atBase}$ **then** $\pi x\, (\mathit{dropS}(\boldsymbol{a}, x))$
    **else** $\mathit{carryToBase}(\boldsymbol{a})$
**end**.
```

Informally, in the program schema, the agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ first have to decide whether to move towards the most probable gold location, or to remain in their position, or to try to pick up the gold (through the $\mathit{pickOrGo}$ procedure). We assume that they can try this maximally n times. Once a joint action of $\mathit{pickOrGo}$ is executed, if one of the two agents $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ holds the gold, then it can start to move towards the base (through the $\mathit{carryToBase}$ procedure).

The agent behavior is partially specified by the procedure $\mathit{schema}(n)$, which is fully instantiated by the program interpreter given the current belief states of the two agents. Given the context in Fig. 3, we now focus on the possible instances of $\mathit{schema}(2)$, that is, the agents can spend two attempts to pick up the object. From $\boldsymbol{a}_1$'s point of view (Fig. 3, left side), a good policy $\pi_{\boldsymbol{a}_1}$ of $\mathit{schema}(2)$ may be to start with the joint action $c = \{\mathit{pickUpS}(\boldsymbol{a}_1, g), \mathit{moveS}(\boldsymbol{a}_2, \mathit{stand})\}$, and then to act depending on $\boldsymbol{a}_1$'s observation:

```
$\pi_{\boldsymbol{a}_1} = c\,;$ **if** $\mathit{obs}_{\boldsymbol{a}_1}(\mathit{success}) \wedge \mathit{obs}_{\boldsymbol{a}_2}(\mathit{success})$ **then** $\pi^1_{\boldsymbol{a}_1}$
            **else if** $\mathit{obs}_{\boldsymbol{a}_1}(\mathit{success}) \wedge \mathit{obs}_{\boldsymbol{a}_2}(\mathit{failure})$ **then** $\pi^2_{\boldsymbol{a}_1}$
                **else if** $\mathit{obs}_{\boldsymbol{a}_1}(\mathit{failure}) \wedge \mathit{obs}_{\boldsymbol{a}_2}(\mathit{success})$ **then** $\pi^3_{\boldsymbol{a}_1}$
                    **else if** $\mathit{obs}_{\boldsymbol{a}_1}(\mathit{failure}) \wedge \mathit{obs}_{\boldsymbol{a}_2}(\mathit{failure})$ **then** $\pi^4_{\boldsymbol{a}_1}$ .
```

Indeed, acting in this way, $\boldsymbol{a}_1$ could try to get its gold and then bring it to the base, avoiding a collision with $\boldsymbol{a}_2$. On the other hand, from $\boldsymbol{a}_2$'s point of view (Fig. 3, right side), a good policy $\pi_{\boldsymbol{a}_2}$ of schema may be to start with the joint action $c = \{moveS(\boldsymbol{a}_1, E), pickUpS(\boldsymbol{a}_2, g)\}$. In fact, in this way, both $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ could try to pick up some gold and then move towards the base. However, the situation here is very complex. Indeed, each action could fail, and each failure determines a different context. The complete policy is generated by the POGTGolog interpreter. Assuming the horizon $H = 5$, an optimal 5-step joint policy π is given by $DT \models OptG([schema(2); nil], (b_{\boldsymbol{a}_1}, b_{\boldsymbol{a}_2}), 5, \pi, (v_1, v_2), (pr_1, pr_2))$. Since both $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ know each others initial belief states, and they are both endowed with the same POGTGolog program $schema$, the interpreter of each of them independently produces the same joint 5-step policy. The obtained policy is a pure strategy that starts with the joint action $c = \{moveS(\boldsymbol{a}_1, stand), moveS(\boldsymbol{a}_2, N)\}$. Indeed, since agent $\boldsymbol{a}_2$ is closer to the base, and the $moveS$ action is not reliable, from the perspectives of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, an optimal joint strategy is to keep $\boldsymbol{a}_1$ idle, letting $\boldsymbol{a}_2$ try to pick up some gold from $(7, 4)$. That is, the produced policy is the following:

```
$\pi = c$ ; **if** $obs_{\boldsymbol{a}_2}(success)$ **then** [
            $\{moveS(\boldsymbol{a}_1, stand), pickUpS(\boldsymbol{a}_2, g)\}$;
            **if** $obs_{\boldsymbol{a}_2}(success)$ **then** [
                $\{moveS(\boldsymbol{a}_1, stand), moveS(\boldsymbol{a}_2, E)\}$;
                **if** $obs_{\boldsymbol{a}_2}(success)$ **then** [
                    $\{moveS(\boldsymbol{a}_1, stand), moveS(\boldsymbol{a}_2, E)\}$;
                    **if** $obs_{\boldsymbol{a}_2}(success)$ **then**
                        $\{moveS(\boldsymbol{a}_1, stand), dropS(\boldsymbol{a}_2, g)\}$
                    **else if** $obs_{\boldsymbol{a}_2}(failure)$ **then** $\pi_1$ ]
                **else if** $obs_{\boldsymbol{a}_2}(failure)$ **then** $\pi_2$ ]
            **else if** $obs_{\boldsymbol{a}_2}(failure)$ **then** $\pi_3$ ]
        **else if** $obs_{\boldsymbol{a}_2}(failure)$ **then** $\pi_4$ .
```

Here, every π_n is a sequence of n joint idle actions, that is, $\pi_n = \{moveS(\boldsymbol{a}_1, stand), moveS(\boldsymbol{a}_2, stand)\}$; π_{n-1}. The rewards of the obtained strategy π are 0.146 and 0.186 for $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$, respectively. Note that there are two other Nash equilibria of $schema(2)$: A pure strategy π_1, which starts with $c = \{moveS(\boldsymbol{a}_1, E), moveS(\boldsymbol{a}_2, stand)\}$, and a mixed strategy π_2, which starts with $\pi_{\boldsymbol{a}_1} \cdot \pi_{\boldsymbol{a}_2}$, where $\pi_{\boldsymbol{a}_1} = \{(moveS(\boldsymbol{a}_1, stand), 0.4), (moveS(\boldsymbol{a}_1, E), 0.6)\}$ and $\pi_{\boldsymbol{a}_2} = \{(moveS(\boldsymbol{a}_2, N), 0.52), (moveS(\boldsymbol{a}_2, stand), 0.48)\}$ (both of them associated with a lower reward for both agents). The Nash equilibrium selected by the agents depends on the Nash selection function embedded in the POGTGolog interpreter.

6 Related Work

In the situation calculus literature, we can find other frameworks for reasoning about actions in multi-agent contexts. In [44], the authors provide a situation calculus based framework suitable for reasoning about multi-agent beliefs, abilities, and multi-agent communicative actions. Here, the agents' mental states are explicitly represented deploying a possible world semantics and introducing accessibility relations between situations (following [43]). In this setting, a Golog-based agent programming language is proposed for the specification and verification of complex multi-agent systems. In contrast, our main concern here is to use a relational representation inspired by partially observable stochastic games and functional abstractions for policy synthesis. Given this aim, our treatment of beliefs is different from the one in [44, 43, 1]. In this paper, we are not interested in explicit representation of beliefs as modalities for reasoning about mental states of the agents. Instead, we use belief states for state abstraction, that is, belief states are outside the object language as a set of state formulas generalizing the belief states in POSGs.

More closely related to our framework are recent extensions of DTGolog [6, 10, 9] to the multi-agent setting. In [6], Lakemeyer and his group present ICPGolog, a multi-agent Golog framework for team playing. ICPGolog integrates different features like concurrency, exogenous actions, continuous change, and the possibility to project into the future. The framework is used in the robotic soccer domain. Here, multi-agent coordination is achieved without communication by assuming that the world models of the agents do not differ too much. Differently from POGTGolog, the setting is fully observable and no game-theoretic mechanism is used.

Another closely related work is Poole's independent choice logic (ICL) [37], which is a representation and reasoning formalism for single- and multi-agent systems that is based on acyclic logic programs under different "choices". Poole's ICL can be used as a formalism for logically encoding games in extensive and normal form. Differently from POGTGolog, Poole's ICL aims more at representing games and generalized strategies, while the problem of policy synthesis is not addressed. Furthermore, partially observable stochastic games are also not treated.

In [30], high-level agent programming in the FLUX framework is considered in a multi-agent setting, modeling communicative actions among the agents. Here, the agents can reason about the other agents' knowledge and communication skills. Also in this case, the specification is based on modalities, while decision- and game-theoretic problems are not treated. Logic-based multi-agent programming is also investigated in the BDI framework, where the main focus is the specification and formal verification [3, 2] of BDI systems using BDI logical languages [48, 23].

Several authors have investigated graphical representations of games [25, 26, 46], where each player's reward function depends on a subset of players described in a graph structure, which exploit the locality of the interactions to obtain compact models and efficient algorithms. Here, an n-player normal form game is explicitly described by an undirected graph on n vertices, representing the n players, and a set of n matrices, each representing a local subgame (involving only some of the players). In our system, the interaction structure is explicitly encoded, both in the action theory and in POGTGolog procedures. Hence, local dependencies among the players are also available. The multi-agent influence diagrams [46] permit a structured and compact representation of extensive form games, involving time and information, using graphical models. Like in [37], the framework extends influence diagrams and Bayesian networks to the multi-agent setting. However, the focus of these works is very different from ours. In fact, their main concern is computational, and the structured representation is used to reduce the computational cost of finding equilibria. Instead, we propose an agent programming language suitable for multi-agent settings, integrating declarative and procedural features.

Other related works deal with multi-agent decision-theoretic planning in extensions of MDPs [35, 39, 21, 7]. In particular, decentralized POMDPs (Dec-POMDPs) [35, 21] have been explored, which are multi-agent POMDPs where the dynamic system is controlled by multiple distributed agents with common payoff, each with possibly different information about the current state of the world. Another approach to multi-agent POMDPs are communicative multi-agent team decision problems [39], which allow to subsume and analyze many existing models of multi-agent cooperative systems. Interestingly, both logic-based and decision-theoretic approaches can be embedded and assessed in this framework. The free communication model is also defined and analyzed in [39]. Closely related are Dec-POMDPs with communication [21], which allow for studying the tradeoff between the cost and the value of the information acquired in the communication process and its influence on the joint utility of the agents. Dec-POMDPs with free communication are investigated in [42], where a free communication model is used at planning time to simplify the policy generation, while the problem of communication cost and limited resources is handled at the execution time. Differently from our free communication model, the agents in [39, 42] have a unique reward function.

Identical payoff stochastic games [35] represent cooperative games by restricting each agent of the game to a single payoff representing the team reward. An algorithm that approximates POSGs as a series of smaller Bayesian games is proposed in [7]. Interactive POMDPs [20] are a control paradigm that complements and generalizes the traditional (Nash) equilibrium approach. Like in our work, the main focus is on policy synthesis for agent control, but [20] addresses directly the synthesis problem in the space of states, while we focus on the representational problem aiming at providing a tool for specifying abstract domains and suitable for balancing the tradeoff between procedural programming and planning.

From the representational perspective, further related works focus on relational and first-order extensions of MDPs [4, 49, 29, 19], multi-agent MDPs [22], and Markov games [12]. In all these works, partial observability is not addressed.

7 Conclusion

We have presented the agent programming language POGTGolog, which combines explicit agent programming in Golog with game-theoretic multi-agent planning in partially observable stochastic games. It allows for modeling one team of cooperative agents under partial observability, where the agents may also have different initial belief states and not necessarily the same rewards. POGTGolog allows to encode partial control programs in a high-level logical language, which are then completed by an interpreter in an optimal way. 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 also shown that POGTGolog programs faithfully extend partially observable stochastic games. We have illustrated the usefulness of this approach along several examples.

An interesting topic of future research is to generalize POGTGolog in the direction of weakening the free communication assumption. In particular, we are currently exploring mechanisms for implicit communication between the agents [8]. Alternatively, one may also allow for explicit communication between the agents (for example, along the lines of [39, 21]) assuming a cost associated with communication actions. Another direction of future research is to generalize POGTGolog to two competing teams of cooperative agents under partially observability.

Appendix A: Proofs for Section 4

Proof of Theorem 4.1. Let $DT = (AT, ST, OT)$ be a domain theory, let p be a POGTGolog program relative to DT, let b be a belief state, and let $H \geqslant 0$ be a horizon. Observe first that $DT \models \mathit{OptG}(\widehat{p}, b, H, \pi, v, pr)$ implies $DT \models G(\widehat{p}, b, H, \pi, v, pr)$. Hence, if $DT \models \mathit{OptG}(\widehat{p}, b, H, \pi, v, pr)$, then π is a H-step policy of p in b relative to DT, and $\mathit{utility}(v_i, pr_i)$ is its expected H-step utility to agent $i \in I$. Thus, it only remains to prove the following statement: $(\star)$ if $DT \models \mathit{OptG}(\widehat{p}, b, H, \pi, v, pr)$, then π is an H-step Nash equilibrium of p in b relative to DT. We give a proof by induction on the structure of OptG.

Basis: The statement $(\star)$ trivially holds for the null program and zero horizon cases. Indeed, in these cases, OptG generates only the policy $\pi = \mathit{nil}$.

Induction: For every program construct that involves no action choice of one of the two agents, the statement $(\star)$ holds by the induction hypothesis. We now prove the statement $(\star)$ for the remaining constructs:

(1) *Nondeterministic action choice of agent i:* Let $\widehat{p} = [\mathbf{choice}(i\colon a_1|\cdots|a_n)\,;p']$, and let π be the H-step policy associated with p via OptG. By the induction hypothesis, for every $k \in \{1,\ldots,n\}$, it holds

that $DT \models OptG([a_k; p'], b, H, a_k; \pi_k, v_k,\, pr_k)$ implies that the policy $a_k; \pi_k$ is an H-step Nash equilibrium of the program $[a_k; p']$ in b. By construction, π is the policy with the maximal expected H-step utility among the $a_k; \pi_k$'s. Hence, any different action selection a_k would not be better for i, that is, $U_i(H, b, a_q; \pi_q) \leqslant U_i(H, b, \pi)$ for all $q \in \{1, \ldots, n\}$. That is, any first action deviation from π would not better for the agent i. Moreover, since each $a_k; \pi_k$ is an H-step Nash equilibrium of $[a_k; p']$ in b, also any following deviation from π would not be better for i. In summary, this shows that $U_i(H, b, \pi') \leqslant U_i(H, b, \pi)$ for every H-step policy π' of p in b relative to DT obtained from π by modifying only actions of agent i. Also for any agent $j \neq i$, any unilateral deviation π' from π cannot be better. In fact, since j is not involved in the first action choice, j can deviate from π only after i's selection of $a_k; \pi_k$, but this would not be better for j by the induction hypothesis. Hence, $U_j(H, b, \pi') \leqslant U_j(H, b, \pi)$ for every H-step policy π' of p in b relative to DT obtained from π by modifying only actions of agent j.

(2) *Nondeterministic joint action choice:* Let $\widehat{p} = [\,\|_{j\in J}$**choice**$(j\colon\, a_{j,1}|\cdots|a_{j,n_j}); p']$, and let π be the H-step policy that is associated with p via $OptG$. By the induction hypothesis, $DT \models OptG([\bigcup_{j\in J} a_j; p'],$ $b,\, H,\, \bigcup_{j\in J} a_j; \pi_a, v_a, pr_a)$ implies that each $\bigcup_{j\in J} a_j; \pi_a$ is an H-step Nash equilibrium of $[\bigcup_{j\in J} a_j; p']$ in b. We now prove that π is an H-step Nash equilibrium of p in b. Observe first that, by construction, π is of the form $\Pi_{j\in J}\pi_j; \pi'$, where $(\pi_j)_{j\in J}$ is a Nash equilibrium (computed via the Nash selection function $selectNash$) of the matrix game consisting of all $r_a = utility(v_a, pr_a)|_J$ such that $a{\in}A$. Thus, if agent j deviates from π_j with π'_j, it would not do better, that is, $U_j(H, b, \pi') \leqslant U_j(H, b, \pi)$, where π' is obtained from π by replacing π_j by π'_j. That is, any first action deviation from π would not be better for agent j. Moreover, by the induction hypothesis, also any following deviation from π' would not be better for j. In summary, this shows that $U_j(H, b, \pi') \leqslant U_j(H, b, \pi)$ for every H-step policy π' of p in b relative to DT that is obtained from π by modifying only actions of agent j.

(3) *Nondeterministic choice of two programs:* The line of argumentation is similar to the one in the case of nondeterministic action choice of agent i above. $\Box$

Proof of Theorem 4.2. Suppose that $G = (I, Z, (A_i)_{i\in I}, (O_i)_{i\in I}, P, (R_i)_{i\in I})$ is a partially observable stochastic game. Without loss of generality, let the A_i's be pairwise disjoint. We now construct a domain theory $DT = (AT, ST, OT)$, a set of situation constants $\{S_z \mid z \in Z\}$, a function g mapping any belief state b of G into the belief state B_b of DT, and a set of POGTGolog programs $\{p^h \mid h \in \{0, \ldots, H\}\}$ relative to DT such that $\sigma = (\sigma_i)_{i\in I}$ is an H-step Nash equilibrium of G, where every $\sigma_i(b, h) = \pi_i$, $i \in I$, is given by $DT \models OptG(\widehat{p}^h,\, B_b,\, h+1,\, \Pi_{i\in I}\pi_i; \pi',\, v, pr)$, for all b and $h \in \{0, \ldots, H\}$, and the expected H-step reward of σ in b to agent i is given by $utility(v_i, pr_i)$, where $DT \models OptG(\widehat{p}^H, B_b, H+1, \pi, v, pr)$.

The basic action theory AT comprises a situation constant S_z for every state $z \in Z$ and a fluent $state(z, s)$ that associates with every situation s a state $z \in Z$ such that $state(z, S_z)$ for all $z \in Z$. Here, every state $z \in Z$ is represented by a constant, and different states are interpreted in a different way. Informally, the set of all situations is given by the set of all situations that are reachable from the situations S_z with $z \in Z$ (and thus we neglect the situation S_0), and Z partitions the set of all situations into equivalence classes (one for each $z \in Z$) via the fluent $state(z, s)$. Given the situation constants S_z and the fluent $state(z, S_z)$, we map any belief state $b = (b_i)_{i\in I}$ of G to a belief state $B_b = (g(b_i))_{i\in I}$ through the function g such that every $(z, \mu) \in b_i$ is mapped to $(S_z, \mu) \in g(b_i)$. The basic action theory AT also comprises a deterministic action $n_{a,z}$ for every joint action $a \in A = \bigtimes_{i\in I} A_i$ and $z \in Z$, which performs a transition into the situation S_z, that is, $state(z, do(n_{a,z}, s))$ for all states $z \in Z$ and situations s. The actions $n_{a,z}$ are executable in every situation s, that is, $Poss(n_{a,z}, s) \equiv \top$ for all states $z \in Z$ and situations s.

The stochastic theory ST comprises the stochastic action $\{a_i \mid i \in I\}$ for every joint action $a \in A$ along with all axioms $stochastic(\{a_i \mid i \in I\}, s, \{n_{a,z'}\}, o, P(z', o|z, a))$ such that (i) $z, z' \in Z$, (ii) s is a situation that satisfies $state(z, s)$ and that contains at most $H + 1$ actions, and (iii) $o \in O = \times_{i \in I} O_i$. Informally, the stochastic theory ST represents the transition probabilities encoded in P.

The optimization theory OT comprises all axioms $reward(i, \{n_{a,z'}\},\ s) = R_i(a, z)$ such that (i) $i \in I$, (ii) $a \in A$, (iii) $z, z' \in Z$, and (iv) s is a situation that satisfies $state(z, s)$ with at most $H + 1$ actions. Let $f = selectNash$ be a Nash selection function for normal form games $M = (I, (A_i)_{i \in I}, (S_i)_{i \in I})$, and let the expected reward to agent $i \in I$ under the Nash equilibrium $f(M)$ be denoted by $v^i_f(M)$.

Finally, every POGTGolog program p^h is a sequence of $h{+}1$ nondeterministic joint action choices of the form $\|_{i \in I}$**choice**$(i\colon a_{i,1}| \cdots |a_{i,n_i})$, where $a_{i,1}, \ldots, a_{i,n_i}$ are all the singleton subsets of A_i (representing all the actions in A_i) for all $i \in I$.

Observe first that $pr = 1$ for every success probability pr computed in $OptG$ for such programs p^h (since the preconditions of all actions are always satisfied). Since $utility(v, pr) = v \cdot pr$, it then follows that $utility(v, pr) = v$ for every expected reward v and success probability pr computed in $OptG$ for the programs p^h.

We now prove the statement of the theorem by induction on the horizon $H \geqslant 0$. For every belief state $b = (b_i)_{i \in I}$ of G and every $h \in \{0, \ldots, H\}$, let the normal form game $G[b, h] = (I, (A_i)_{i \in I}, (Q_i[b, h])_{i \in I})$ be defined by $(Q_i[b, h])_{i \in I}(a) = v_a$, where $DT \models OptG([\{a_i \mid i \in I\};\ \widehat{p}^{h-1}], B_b, h{+}1, \pi_a, v_a, pr_a)$. By induction on the horizon $H \geqslant 0$, we now prove that for all $i \in I$:

($\star$) (i) $Q_i[b, 0](a) = R_i(b_i, a)$ for every belief state b of G, and
(ii) $Q_i[b, h](a) = R_i(b_i, a) + \sum_{o \in O} v^i_f(G[(b^{a,o}_i)_{i \in I}, h{-}1]) \cdot P_b(b^{a,o}_i | b_i, a)$ for every belief state b of G and $h \in \{1, \ldots, H\}$.

This then implies that $(v^i_f(G[b, h]))_{i \in I} = v$ and $f(G[b, h]) = (\pi_i)_{i \in I}$ are given by $DT \models OptG(\widehat{p}^h, B_b, h{+}1, \Pi_{i \in I}\pi_i; \pi', v, pr)$, for every b and every $h \in \{0, \ldots, H\}$. Furthermore, by finite-horizon value iteration [24], the mixed policy $\sigma = (\sigma_i)_{i \in I}$ that is defined by $(\sigma_i(b, h))_{i \in I} = f(G[b, h])$, for every b and every $h \in \{0, \ldots, H\}$, is a H-step Nash equilibrium of G, and it holds that $G_i(H, b, \sigma) = v^i_f(G[b, H])$ for every $i \in I$. This then proves the theorem. Hence, it only remains to show by induction on the horizon $H \geqslant 0$ that the statement ($\star$) holds, which is done as follows:

Basis: Let $H = 0$, and thus we only have to consider the case $h = 0$. Let $DT \models OptG([\widetilde{a}; nil], B_b, 1, \pi_a, v_a, pr_a)$, with $\widetilde{a} = \{a_i \mid i \in I\}$. Using the definition of $OptG$ for the case of stochastic first program action $\widetilde{a}$, we then obtain $v_a = \sum_{o \in O} v_{a,o} \cdot prob(\widetilde{a}, B_b, o)$, with $DT \models OptG([\widetilde{a}_o\,; nil],\ B_b,\ 1,\ \widetilde{a}_o\,; \pi_{a,o},\ v_{a,o},\ pr_{a,o})$. Using the definition of $OptG$ for the case of stochastic first program action with observation, we then obtain $v_{a,o} = v'_{a,o} + reward(\widetilde{a}_o, B_b) = 0 + (R_i(b_i, a))_{i \in I}$. It thus follows that $v_a = (R_i(b_i, a))_{i \in I} \cdot \sum_{o \in O} prob(\widetilde{a}, B_b, o) = (R_i(b_i, a))_{i \in I}$.

Induction: Let $H > 0$. By the induction hypothesis, it holds that (i) $Q_i[b, 0](a) = R_i(b_i, a)$ for every belief state b reachable from b_0, and (ii) $Q_i[b, h](a) = R_i(b_i, a) + \sum_{o \in O} v^i_f(G[(b^{a,o}_i)_{i \in I}, h{-}1]) \cdot P_b(b^{a,o}_i | b_i, a)$ for every belief state b reachable from b_0 and every $h \in \{1, \ldots, H - 1\}$. Furthermore, as argued above, $(v^i_f(G[b, h]))_{i \in I} = v$ and $f(G[b, h]) = (\pi_i)_{i \in I}$ are given by $DT \models OptG(\widehat{p}^h, B_b, h{+}1, \Pi_{i \in I}\pi_i; \pi', v, pr)$ for every b that is reachable from b_0, and every $h \in \{0, \ldots, H - 1\}$. Suppose now that $DT \models OptG([\widetilde{a}\,; \widehat{p}^{h-1}], B_b, h + 1, \pi_a, v_a, pr_a)$, with $\widetilde{a} = \{a_i \mid i \in I\}$. Using the definition of $OptG$ for the case of stochastic first program action $\widetilde{a}$, we then obtain that $v_a = \sum_{o \in O} v_{a,o} \cdot prob(\widetilde{a}, B_b, o)$, where $prob(\widetilde{a}, B_b, o) = (P_b(b^{a,o}_i | b_i, a))_{i \in I}$, and every $v_{a,o}$ is given by $DT \models OptG([\widetilde{a}_o\,; \widehat{p}^{h-1}], B_b, h + 1, \widetilde{a}_o\,; \pi_{a,o}, v_{a,o}, pr_{a,o})$. Using the definition of $OptG$ for the case of stochastic first program action with observation, we then obtain $v_{a,o} = v'_{a,o} + reward(\widetilde{a}_o,\ B_b) = v'_{a,o} + (R_i(b_i, a))_{i \in I}$. By the induction hypothesis, we have that $v'_{a,o} = (v^i_f(G[(b^{a,o}_i)_{i \in I}, h{-}1]))_{i \in I}$. We thus conclude that $v_{a,i} = R_i(b_i, a) + \sum_{o \in O} v^i_f(G[(b^{a,o}_i)_{i \in I}, h{-}1]) \cdot P_b(b^{a,o}_i | b_i, a)$. $\Box$

References

[1] F. Bacchus, J. Y. Halpern, and H. J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artif. Intell.*, 111(1–2):171–208, 1999.

[2] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Model checking rational agents. *IEEE Intelligent Systems*, 19(5):46–52, 2004.

[3] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[4] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDPs. In *Proceedings IJCAI-2001*, pp. 690–700. Morgan Kaufmann, 2001.

[5] 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. AAAI Press/MIT Press, 2000.

[6] F. Dylla, A. Ferrein, and G. Lakemeyer. Specifying multirobot coordination in ICPGolog – From simulation towards real robots. In *Proceedings AOS-2003*, 2003.

[7] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings AAMAS-2004*, pp. 136–143. IEEE Computer Society, 2004.

[8] A. Farinelli, A. Finzi, and T. Lukasiewicz. Team programming in Golog under partial observability. In *Proceedings IJCAI-2007*. 2007. In press.

[9] A. Ferrein, C. Fritz, and G. Lakemeyer. On-line decision-theoretic Golog for unpredictable domains. In *Proceedings KI-2004*, volume 3238 of *LNCS/LNAI*, pp. 322–336. Springer, 2004.

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

[11] A. Finzi and T. Lukasiewicz. Game-theoretic agent programming in Golog. In *Proceedings ECAI-2004*, pp. 23–27. IOS Press, 2004.

[12] A. Finzi and T. Lukasiewicz. Relational Markov games. In *Proceedings JELIA-2004*, volume 3229 of *LNCS/LNAI*, pp. 320–333. Springer, 2004.

[13] A. Finzi and T. Lukasiewicz. Game-theoretic agent programming in Golog under partial observability In *Proceedings GTDT-2005*, 2005.

[14] A. Finzi and T. Lukasiewicz. Game-theoretic Golog under partial observability. In *Proceedings AAMAS-2005*, pp. 1301–1302. ACM Press, 2005.

[15] A. Finzi and T. Lukasiewicz. Game-theoretic agent programming in Golog under partial observability. In *Proceedings KI-2006*, volume 4314 of *LNCS/LNAI*, pp. 113–127. Springer, 2007.

[16] A. Finzi and T. Lukasiewicz. Adaptive multi-agent programming in GTGolog. In *Proceedings ECAI-2006*, pp. 753–754. IOS Press, 2006.

[17] A. Finzi and T. Lukasiewicz. Adaptive multi-agent programming in GTGolog. In *Proceedings KI-2006*, volume 4314 of *LNCS/LNAI*, pp. 389–403. Springer, 2007.

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

[19] N. H. Gardiol and L. Pack Kaelbling. Envelope-based planning in relational MDPs. In *Proceedings NIPS-2003*. MIT Press, 2003.

[20] P. J. Gmytrasiewicz and P. Doshi. Interactive POMDPs: Properties and preliminary results. In *Proc. AAMAS-2004*, pp. 1374–1375. IEEE Computer Society, 2004.

[21] C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings AAMAS-2003*, pp. 137–144. ACM Press, 2003.

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

[23] A. Herzig and N. Troquard. Knowing how to play: Uniform choices in logics of agency. In *Proceedings AAMAS-2006*, pp. 209–216. ACM Press, 2006.

[24] M. Kearns, Y. Mansour, and S. Singh. Fast planning in stochastic games. In *Proceedings UAI-2000*, pp. 309–316. Morgan Kaufmann, 2000.

[25] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proceedings UAI-2001*, pp. 253–260. Morgan Kaufmann, 2001.

[26] P. La Mura. Game networks. In *Proceedings UAI-2000*, pp. 335–342. Morgan Kaufmann, 2000.

[27] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings ICML-1994*, pp. 157–163. Morgan Kaufmann, 1994.

[28] B. Marthi, S. J. Russell, D. Latham, and C. Guestrin. Concurrent hierarchical reinforcement learning. In *Proceedings IJCAI-2005*, pp. 779–785. 2005.

[29] M. Martin and H. Geffner. Learning generalized policies from planning examples using concept languages. *Appl. Intell.*, 20(1):9–19, 2004.

[30] Y. Martin, I. Narasamdya, and M. Thielscher. Knowledge of other agents and communicative actions in the fluent calculus. In *Proceedings KR-2004*, pp. 623–633. AAAI Press, 2004.

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

[32] 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. Morgan Kaufmann, 2003.

[33] G. Owen. *Game Theory: Second Edition*. Academic Press, 1982.

[34] 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.

[35] L. Peshkin, K.-E. Kim, N. Meuleau, and L. Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings UAI-2000*, pp. 489–496. Morgan Kaufmann, 2000.

[36] J. Pinto. Integrating discrete and continuous change in a logical framework. *Computational Intelligence*, 14(1):39–88, 1998.

[37] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artif. Intell.*, 94(1–2):7–56, 1997.

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

[39] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *J. Artif. Intell. Res.*, 16:389–423, 2002.

[40] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pp. 359–380. Academic Press, 1991.

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

[42] M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings AAMAS-2005*, pp. 786–793. 2005.

[43] R. B. Scherl and H. J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings AAAI-1993*, pp. 689–697. AAAI Press/MIT Press, 1993.

[44] S. Shapiro, Y. Lesperance, and H. J. Levesque. The cognitive agents specification language and verification environment for multiagent systems. In *Proceedings AAMAS-2002*, pp. 19–26. ACM Press, 2002.

[45] J. van der Wal. *Stochastic Dynamic Programming*, volume 139 of *Mathematical Centre Tracts*. Morgan Kaufmann, 1981.

[46] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proceedings AAAI-2002*, pp. 345–351. AAAI Press, 2002.

[47] J. von Neumann and O. Morgenstern. *The Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[48] M. Wooldridge. *Reasoning about rational agents*. MIT Press, Cambridge, MA, 2001.

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