Intention-based Decision Making for Strategic Scenarios Dynamics via Computational Logic

The Anh Han[1] and Luís Moniz Pereira[2]

[1] AI lab, Computer Science Department, Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium; email: h.anh@ai.vub.ac.be
[2] Centro de Inteligência Artificial (CENTRIA), DI/FCT, Universidade Nova de Lisboa
2829-516 Caparica, Portugal; email: lmp@fct.unl.pt

Abstract. We provide a characterization of different strategic scenarios within our Logic Programming based (previously implemented) evolution prospection system, and describe how to express and enact therein intention-based decision making supported by computational logic. The evolution prospection system had already implemented several kinds of environment triggering constructs—such as different forms of preferences and integrity constraints—and has also been empowered with an ability to take into account intentions of others in the decision making process, by means of an intention recognition system. Herein we generalize, extend, and explore how the system can be further used for decision making in different strategic dynamic scenarios, namely games, where strategies can be conceived as the intentions of co-players.
Keywords. Intention-based decision making, Game Theory, Evolution Prospection, Logic Programming, Intention Recognition.

1 Introduction

In strategic and economic situations as typically modeled using the game theoretical framework [13], the achievement of a goal by an agent usually does not depend uniquely on its own actions, but also on the decisions and actions of others—especially when the possibility of communication is limited [21,17]. The knowledge about the intentions of others in such situations could enable a recognizing agent to plan in advance, either to secure a successful cooperation or to deal with potential hostile behaviours, and thus to take the best advantage of such knowledge [4,22,7]. Additionally, in more realistic settings where deceit may offer additional profits, agents often attempt to hide their real intentions, change or abandon them along the way when needed, make others believe in the bogus ones, and even go to the extent of defaulting on commitments [11,7]. Undoubtedly, in all such dynamic situations an ability to recognize intentions of others and their own dynamics and take them into account when making decisions is crucial, empowering its holders with significant net benefit or evolutionary advantages. In addition, there is a large body of literature on experimental economics that shows the importance of intention-based decision making in diverse kinds of strategic games, such as the Prisoner's Dilemma [5] and the Ultimatum game [18]. Furthermore, computational models show that the taking into account of the ongoing strategic intentions of others is crucial for agents' success in the course of a diversity of dynamic, strategic scenarios [9,10,7].

Relying on our previously implemented Evolution Prospection (EP) system [15], herein we generalize and extend it, and explore different ways in which the system can be used for knowledge representation and reasoning in strategic dynamic scenarios, and, more importantly, how intention-based modelling can be useful therein. EP is an implemented, Logic Programming (LP) based system for decision making [15,17] (described in Section 2). An EP agent can prospectively look ahead a number of steps into the future to choose the best course of action choice evolution that satisfies a goal. This is achieved by designing and implementing several kinds of prior and post preferences on abductive hypotheses generated scenarios [1], and several other useful environment-triggering constructs for decision making. To take into account intentions of others in the decision making process, EP has been extended with an intention recognition system, also implemented in LP, thus providing for a coherent integration [8] (Section 3).

2 Evolution Prospection and Extensions

The implemented Evolution Prospection (EP) system has proven useful for decision making [15,17]. It is implemented on top of ABDUAL, a preliminary implementation of [1], using XSB Prolog [23]. We next describe the constructs of EP, to the extent we use them here. We also describe *novel* extensions for its application in diverse game scenarios, where strategies are recognized as intentions, illustrating with examples. A full account, including its semantics, can be found in [15].

Language Let $\mathcal{L}$ be a first order language. A domain literal in $\mathcal{L}$ is a domain atom A or its default negation $not\ A$. The latter is used to express that the atom is false by default (Closed World Assumption). A domain rule in $\mathcal{L}$ is a rule of the form: $A \leftarrow L_1, \ldots, L_t\ (t \geq 0)$, where A is a domain atom and $L_1, \ldots, L_t$ are domain literals. An integrity constraint (IC) in $\mathcal{L}$ is a rule with an empty head. A (logic) program P over $\mathcal{L}$ is a set of domain rules and integrity constraints, standing for all their ground instances.

Here one considers solely Normal Logic Programs (NLPs), those whose heads of rules are positive literals, or empty. One focuses furthermore on abductive logic programs [1], i.e. NLPs allowing for abducibles – user-specified positive literals without rules, whose truth-value is not fixed. Abducibles instances or their default negations may appear in bodies of rules, like any other literal. They stand for hypotheses, each of which may independently be assumed true, in positive literal or default negation form—in order to produce an abductive solution to a query. A detailed definition is in [1].

Active Goals In each cycle of its evolution the agent has a set of active goals or desires. The $on\_observe/1$ predicate is introduced, which is considered as representing active goals or desires that, once triggered by the observations figuring in its rule bodies, cause the agent to attempt their satisfaction by launching all the queries standing for them, or using preferences to select them. The rule for an active goal AG is of the form: $on\_observe(AG) \leftarrow L_1, ..., L_t\ (t \geq 0)$, where L_1,...,L_t are domain literals. When starting a cycle, the agent collects its active goals by finding all the $on\_observe(AG)$ that hold under the initial theory without performing any abduction, then finds abductive solutions for their conjunction.

Preferring Abducibles An abducible A can be assumed only if it is a considered one, i.e. if it is expected in the given situation, and, moreover, there is no expectation to the contrary: $consider(A) \leftarrow expect(A),\ not\ expect\_not(A),\ A$.

The rules about expectations are domain-specific knowledge contained in the theory of the program, and effectively constrain the hypotheses available in a situation. Note that for each abducible a *consider*-rule is added automatically into the EP program.

Handling preferences over abductive logic programs has several advantages, and allows for easier and more concise translation into NLPs than those prescribed by more general and complex rule preference frameworks. The advantages of so proceeding stem largely from avoiding combinatory explosions of abductive solutions, by filtering irrelevant as well as less preferred abducibles [14].

To express preference criteria among abducibles, it is envisaged an extended language $\mathcal{L}^\star$. A preference atom in $\mathcal{L}^\star$ is of the form $a \lhd b$, where a and b are abducibles. A preference rule in $\mathcal{L}^\star$ is of the form: $a \lhd b \leftarrow L_1, ..., L_t\ (t \geq 0)$, where $L_1, ..., L_t$ are domain literals over $\mathcal{L}^\star$. The formal semantics of this rule is given in [14]. *A priori* preferences are used to produce the most interesting or relevant conjectures about possible future states. They are taken into account when generating possible scenarios (abductive solutions), which will subsequently be preferred amongst each other a posteriori.

Example 1 (Choose a move). Let us consider a scenario where one needs to choose a move, say from the set $\{a_1, a_2, a_3\}$. Typically, not all theoretically definable moves or acts are actually available as real behavioral choices. So one needs to decide on the set of available moves. Moreover, if one can assign some cost (e.g. lesser costs for moves that require less resources [19]), then one prefers the move that is less costly. The following EP program enables the decision making process.

```
1. abds([move/1]).
2. on_observe(choose).
   choose <- move(a1).  choose <- move(a2).  choose <- move(a3).
   <- move(a1), move(a2).      <- move(a1), move(a3).
   <- move(a2), move(a3).
3. expect(move(X)).     unavail(move(a2)).
4. %cost(a1,1).  cost(a2,3).  cost(a3,5).
5. expect_not(X) <- unavail(X).
6. move(X) <| move(Y) <- cost(X,C1), cost(Y,C2), C1 < C2.
```

This program has three abducibles, $move(a_1)$, $move(a_2)$, and $move(a_3)$ (line 2), and all of them are expected (line 3). But the move a_2 is not available, and therefore not expected (line 5). Hence, the program has two abductive solutions, one with $move(a_1)$ the other with $move(a_3)$.

If one can assign cost to each move (uncomment line 4), the *a prior* preference rule in line 6 is activated, which defeats the solution where only $move(a_3)$ is present, due to the impossibility of simultaneously abducing $move(a_1)$ (line 2).

A Posteriori Preferences Having computed possible scenarios, represented by abductive solutions, more favorable scenarios can be preferred a posteriori. Typically, *a posteriori* preferences are performed by evaluating consequences of abducibles in abductive

solutions. An *a posteriori* preference has the form:

$$A_i \ll A_j \leftarrow holds\_given(L_i, A_i),\ holds\_given(L_j, A_j)$$

where A_i, A_j are abductive solutions and L_i, L_j are domain literals. This means that A_i is preferred to A_j a posteriori if L_i and L_j are true as the side-effects of abductive solutions A_i and A_j, respectively, without any further abduction when testing for the side-effects. Optionally, in the body of the preference rule there can be any Prolog predicate used to quantitatively compare the consequences of the two abductive solutions.

Implementing Maximin and Minimax Decision Rules These decision rules do not take uncertainty into account. They are widely used in decision theory, game theory, statistics and philosophy for maximizing the minimum gain (maximin) or minimizing the maximum possible loss (minimax) [13].

One now shows how to implement these rules, extending the power of the Evolution Prospection system [15]. The *maximin* (*minimax*) orders abductive solutions by their worst-case (best-case, respectively) consequences, i.e. the consequences with smallest (greatest, respectively) utility. The *a posteriori* preference using *maximin* as decision rule has the form: $A_i \ll A_j \leftarrow min\_utility(A_i, U_i),\ min\_utility(A_j, U_j),\ U_i > U_j$, which says that A_i is preferred to A_j a posteriori if a consequence of A_i with minimal utility has greater utility than a consequence of A_j with minimal utility.

Respectively, the *a posteriori* preference using *minimax* as decision rule has the form: $A_i \ll A_j \leftarrow max\_utility(A_i, U_i),\ max\_utility(A_j, U_j),\ U_i < U_j$. It means that A_i is preferred to A_j a posteriori if a consequence with maximal utility of A_i has smaller utility than a consequence with maximal utility of A_j. Predicates $min\_utility/2$ and $max\_utility/2$ are built-in predicates of our system.

Example 2 (Maximin Rule). The following example of a zero-sum game [13], where two players A and B make simultaneous moves, illustrates maximin solutions. Suppose A has three choices (a_1, a_2 and a_3) and B has two choices (b_1 and b_2). The payoff matrix for B is described below in line 7. Then, the maximin choice for B is b_2 since B gains at least 1. The EP program encoding the maximin choice of B follows

```
1. abds([b1/0, b2/0]).
   expect(b1). expect(b2).
2. on_observe(b_choose).
   b_choose <- b1.   b_choose <- b2.
3. c(b1,a1) <- b1.   c(b1,a2) <- b1.   c(b1,a3) <- b1.
4. c(b2,a1) <- b2.   c(b2,a2) <- b2.   c(b2,a3) <- b2.
5. Ai << Aj <- min_utility(Ai, V1),
               min_utility(Aj, V2), V1 > V2.

beginProlog.    % beginning of just Prolog code
6. consequences([c(b1,a1), c(b1,a2), c(b1,a3),
                 c(b2,a1), c(b2,a2), c(b2,a3)]).
7. util(c(b1,a1),3). util(c(b1,a2),-1). util(c(b1,a3),6).
   util(c(b2,a1),3). util(c(b2,a2), 5). util(c(b2,a3),1).
endProlog.      % end of just Prolog code
```

There are two abducibles, b_1 and b_2, both are expected and have no counter-expectation. Thus, there are two possible abductive solutions for the only active goal b_choose. Next, the *a posteriori* preference in line 5 is taken into account, ruling out the one having greater worst-case utility consequence, obtained using the built-in predicate $min\_utility/2$. Thus, it is easily seen that b_1 is ruled out by this preference. In short, the final (maximin) choice of B is b_2.

Note that usual (XSB) Prolog code can be embedded in an EP system by putting it between two reserved predicates *beginProlog/0* and *endProlog/0* (lines 6-7). In line 6, the reserved predicate $consequences/1$ is employed to declare the list of relevant consequences being evaluated. Another reserved predicate, $util/2$, defines the utility of each consequence in that list (line 7).

Generalized A Posteriori Preferences. *A posteriori* preferences between two abductive solutions is enacted by comparing a pair of consequences of each abductive solution. However, more often than not, one abductive solution might have several relevant consequences that contribute to make it either more or less preferred than the other abductive solution it is being compared to. All those relevant consequences are needed to be taken into account for decision making. This is similar to the problem addressed in standard decision theory [6] which is to decide between two actions by evaluating each action's relevant consequences based on an utility function mapping consequences to utilities, typically assumed to be real valued, and a given decision rule.

There have been many decision rules studied in the literature. The best-known one is *expected utility maximization*, which we adopt here to *extend* the EP system for making decision under uncertainty [6]. In general, as for any type of decision rules, it has a set of relevant consequences plus a real-valued utility function mapping those consequences to real numbers [6]. This decision rule also requires a probability measure that characterizes the decision maker's uncertainty with respect to the consequences of a hypothetical abductive solution. It orders the abductive solutions according to the expected utility[1] of their consequences given the probability measure, having the form:

$$A_i \ll A_j \leftarrow expected\_utility(A_i, U_i),\ expected\_utility(A_j, U_j),\ U_i > U_j$$

where A_i, A_j are abductive solutions. This means that A_i is preferred to A_j a posteriori if the expected utility of relevant consequences of A_i is greater than the expected utility of the ones of A_j.

We modify the previous example (Example 2) to illustrate how to derive a decision that maximizes expected payoff.

Example 3 (Choose a move based of maximizing expected utility). Suppose one can assign a probability distribution over the choice of the opponent (i.e. player A's), that is, the probability with which the opponent chooses a particular move. Let

```
8.  pr(a1, 0.1).   pr(a2, 0.4).   pr(a3, 0.5).
9.  prc(c(_,X), P) :- pr(X, P).
```

[1] The expected utility of a set of consequences C given a probability measure Pr mapping the consequences to probability values, i.e. $Pr : C \rightarrow [0, 1]$, and given a utility function U mapping consequences to real-value utilities, i.e. $U : C \rightarrow \mathbb{R}$, is obtained by the formula: $E(C, Pr, U) = \sum_{X \in C} Pr(X)U(X)$.

These facts and rule continue the EP program in Example 2. The probability distribution over the choices of player A is given in line 8 (as Prolog code). The probability of the consequences in line 6 are derived using the Prolog rule in line 9. Their occurrence probability is captured using the reserved predicate $prc/2$, the first argument of which is some consequence being instantiated during the computation of the built-in predicate $expected\_utility/2$, and the second argument the corresponding probability value.

The *a posteriori* preference in line 5 from the previous example is replaced with

```
5. Ai << Aj <- expected_utility(Ai, U1),
               expected_utility(Aj, U2), U1 > U2.
```

Now the second abductive solution with b_2 is ruled out because it leads to smaller expected payoff (2.8) than that of the one with b_1 (2.9).

As will be seen later on, the probability distribution over the possible choices of a co-player can be given by an intention recognition system. In contradistinction to the usual approach of simply counting prior occurrences of each choice (for instance in the repeated interaction setting) [13], using an intention recognition method to derive the probability distribution of the choices can account for the reasons why the co-player chose such choices. For instance, in the course of the iterated PD, if our co-player defected it might be because we defected in the past [9,10]), and he does not trust us much. This is an interesting combination of intention recognition and decision making under uncertainty.

Evolution Result A Posteriori Preference While looking ahead a number of steps into the future, the agent is confronted with the problem of having several different possible courses of evolution. It needs to be able to prefer amongst them to determine the best courses from its present state (and any state in general). The *a posteriori* preferences are no longer appropriate, since they can be used to evaluate only one-step-far consequences of a commitment. *A posteriori* preference is generalized to prefer between two evolutions (i.e. courses of actions). An *evolution result a posteriori* preference is performed by evaluating consequences of following some evolutions [15]. It is extended here to account for multiple relevant consequences of evolutions under uncertainty, by maximizing expected utility using the rule

$$E_i \lll E_j \leftarrow expected\_utility\_evol(E_i, U_i),\ expected\_utility\_evol(E_j, U_j),\ U_i > U_j$$

where E_i, E_j are two distinct evolutions. This preference implies that E_i is preferred to E_j if the expected utility of relevant consequences of pursuing E_i is greater than the expected utility of the ones when pursuing E_j.

Example 4 (Playing games with limited resource [19]). Acting on an environment, an agent may engage in different kinds of games (situations or affairs). With a limited resource, it might not participate in all the game rounds. It thus needs to be able to assess which game round is more important, possibly leading to greater benefits, and may need to forego the immediate interaction (despite its immediate benefit) in order to save the necessary resource for investing in future interactions. The *evolution result a posteriori preferences* can help deal with this kind of games. Let us consider the following EP example for illustration.

```
1. abds([invest_now/0, refuse_now/0,
         invest_later/0, refuse_later/0]).
2. expect(invest_now).              expect(refuse_now).
3. on_observe(choose) <- decide_invest_now.
   choose <- invest_now.            choose <- refuse_now.
4. gain1(100) <- invest_now.        gain1(0) <- refuse_now.
   lose_resource  <- invest_now.    save_resource <- refuse_now.
5. Ai << Aj <- holds_given(gain1(Gi), Ai),
               holds_given(gain1(Gj), Aj), Gi > Gj.

6. on_observe(decide)  <-  invest_opportunity.
   decide <- invest_later.        decide <- refuse_later.
7. expect(invest_later) <- save_resource.
   expect(refuse_later).
8. gain2(1000,0.6) <- invest_later.  gain2(0,1) <- refuse_later.
9. Ei <<< Ej <- expected_utility_evol(Ui, Ei),
                expected_utility_evol(Uj, Ej), Ui > Uj.

begingProlog.
10. consequences([gain1(_), gain2(_,_)]).
11. util(gain1(U), U).        util(gain2(U,_), U).
12. prc(gain1(_), 1).         prc(gain2(_, P), P).
endProlog.
```

In the first cycle of evolution, there are two abducibles, *invest_now* and *refuse_now*, declared in line 1, to solve the active goal *choose*.

In the case the agent is not capable of looking further ahead into the future, it would choose to invest its resource in the current game round since it would lead to immediate benefit (100 vs. 0), i.e. the *a posteriori* preference in line 5 is taken into account immediately. Then, in the next cycle, it would not be able to take advantage of a more beneficial interaction, since the necessary resource is not saved (line 7), and that leads to the outcome with lower expected utility.

But if the agent is able to look further ahead, i.e. it can see that on the next (game) cycle, if it has enough resource, it will have a good opportunity to make a more beneficial investment (line 6). The *evolution result a posteriori* preference is employed in line 9 to prefer the evolution with greater expected utility ($0 \times 1 + 1000 \times 0.6 = 600 > 100 = 100 \times 1 + 0 \times 1$), which is to refuse to invest in the first game round in order to save the resource for the investment in the second game round.

3 Intention-based Decision Making in Game Theory

There are several ways an EP agent can benefit from the ability to recognize intentions of other agents, both in friendly and hostile settings. Knowing the intention of an agent is a means to predict what it will do next or might have done before [4,7]. The recognizing agent can then plan in advance to take the best advantage of the prediction, or act to take remedial action. Technically, in the EP system, this new kind of knowledge may impinge on the body of any EP constructs, such as active goals, expectation and counter-expectation rules, preference rules, integrity constraints, etc., providing for a

new kind of trigger [8]. Proposing commitments to cooperate is a form of intention recognizing too [11,7].

3.1 Intention Recognition with Bayesian Networks

In [16,17], a general Bayesian Network (BN) model for intention recognition is presented. Basically, the BN consists of three layers: cause/reason nodes in the first layer (called *pre-intentional*), connecting to intention nodes in the second one (called *intentional*), in turn connecting to action nodes in the third (called *activity*) (see [16] for details). In general, intention recognition consists in computing the probabilities of each conceivable intention conditional on the current observations, including the observed actions in the third layer, and some of the causes/reasons in the first layer. The prediction of what is the intention of the observed agent can simply be the intention with the greatest conditional probability, possibly above some minimum threshold. Sometimes it is also useful to predict what are the N ($N \geq 2$) most likely intentions given the current observations [3]. The BN inference for intention recognition is implemented using P-log, a probabilistic logic system [12,2], enabling a coherent integration into EP [16,17].

Fig. 1: Bayesian Network for Intention Recognition in Repeated Social Dilemmas.

Example 5 (Intention recognition in repeated games). As an example, consider the three-layered BN for intention recognition in repeated interaction settings (Figure 1), provided in [9]. The pre-intentional layer has one node, *oTrust (Tr)*, the other player's trust in the player, and receives boolean values, t $(true)$ or f $(false)$. The intentional layer has a single node, *Intention (I)*, receiving value C or D. It is causally affected by *oTrust*. The activity layer has one node, *pastObs (O)*, causally affected by *Intention* node. Its value is a pair (n_C, n_D) where n_C and n_D are the number of times the recognized player cooperated and defected, respectively, in the recent M (memory size) steps. *pastObs* is the only observed node. More details can be found in [9].

In order to account for intentions of other agents in decision making with EP, we provide a built-in predicate, $has\_intention(Ag, I)$, stating that an agent *Ag* has the intention I. The truth-value of this predicate is evaluated by the intention recognition system. Whenever this predicate is called in an EP program, the intention recognition system is employed to check if *Ag* has intention I, i.e. I is the most likely conceivable intention at that moment. We also provide predicate $has\_intention(Ag, I, Pr)$, stating that agent *Ag* has intention I with probability Pr. Hence, one can express, for example, the situation where one needs to be more, or less, cautious.

In the sequel we draw closer attention to some EP constructs, exemplifying how to take into account intentions of others for decision making enhancement.

3.2 Intentions Triggering Active Goals

Recall that an active goal has the form: $on\_observe(AG) \leftarrow L_1, ..., L_t$ $(t \geq 0)$, where L_1,...,L_t are domain literals. At the beginning of each cycle of evolution, those literals

are checked with respect to the current evolving knowledge base and trigger the active goal if they all hold. For intention triggering active goals, the domain literals in the body can be in the form of $has\_intention$ predicates, accounting for other agents' intentions.

This way, any intention recognition system can be used as the goal producer for decision making systems, the inputs of which are used to trigger (active) goals to be solved (see for instance [17]).

It is easily seen that intention triggering active goals are ubiquitous. New goals often appear when one recognizes some intentions in others. We might want to help friends to achieve their intentions, while it is desirable to prevent our foes from achieving theirs. Or, we simply want to plan in advance to take advantage of the hypothetical future obtained when the intending agent employs the plan that achieves his intention.

Example 6 (Intention triggering active goals in games). In the previous example, we may say that an opportunity for a good investment can be inferred from recognizing the intention of others who, for instance, intend to put forward a new company.

```
on_observe(decide_invest) <-
          person(P), has_intention(P, open_new_company).
```

Furthermore, one can similarly apply this to any coordination game [13,20], where it is crucial to recognize co-players' intentions to achieve a smooth coordination or cooperation, providing help when necessary.

3.3 Intention Triggering Preferences

Having recognized an intention of another agent, the recognizing agent may either favor or disfavor an abducible (*a priori* preferences), an abductive solution (*a posteriori* preferences) or an evolution (*evolution result a posteriori* preferences) with respect to another, respectively, depending on the setting they are in. If they are in a friendly setting (e.g. players in one same coalition in cooperative games [13]), the one which provides more support to achieve the intention is favored; in contrast, in a hostile setting (e.g. players in different coalitions in cooperative games [13]), the one providing more support is disfavored. The recognizing agent may also favor one which takes better advantage of the recognized intention.

For illustration, let us consider some examples of intention-based decision making within an evolving population of agents interacting using the well-known Prisoner's Dilemma (PD) [13,20], where in each interaction a player needs to choose a move, either to cooperate ('c') or to defect ('d'). In a one-shot PD interaction, one is always better off choosing to defect, but cooperation might be favorable if the PD is repeated (called iterated PD), that is, there is a good chance that players will play the same PD with each other again. Several successful strategies have been provided in the context of the iterated PD (see a survey in [20, Ch. 3]), most famously amongst them are tit-for-tat (*tft*) and win-stay-lose-shift (*wsls*).

The following two strategies (each denoted by IR), operating upon intention-based decision making in the course of iterated PD, have been shown to be better than those two famous strategies [9,10]—they can dynamically assess the ongoing strategic intentions of others as the population evolves. In the sequel we show how they can be modeled within our framework, thereby demonstrating its applicability as a modeling framework of strategies in dynamic game situations [20].

Example 7 (Intention-based decision making rule, in [9,7]). Prefer to cooperate if the co-player intends to cooperate, and prefer to defect otherwise.

1. $abds([move/1])$.
2. $on\_observed(decide) \leftarrow new\_interaction$.
3. $decide \leftarrow move(c)$. $\quad decide \leftarrow move(d)$. $\quad \leftarrow move(c),\ move(d)$.
4. $expect(move(X))$.
5. $move(c) \ \triangleleft \ move(d) \leftarrow has\_intention(co\_player, c)$.
   $move(d) \ \triangleleft \ move(c) \leftarrow has\_intention(co\_player, d)$.

At the start of a new interaction, an IR player needs to choose a move, either cooperate (c) or defect (d) (lines 2-3). Both options are expected, and there are no expectations to the contrary (line 4). There are two *a priori* preferences in line 5, stating that an IR player prefers to cooperate if the co-player's recognized intention is to cooperate, and prefers to defect otherwise. The built-in predicate $has\_intention/2$, when in the body of the preferences, triggers the intention recognition module to validate, on the fly, if the co-player is more likely to have the intention expressed in the second argument.

Now let us see how this strategy behaves when interacting with another strategy, in the course of an iterated PD. Consider an opponent that always defects (denoted by *alld*). In the first interaction, *alld* already defects. Using the BN model in Figure 1, IR predicts that the co-player has intention of defecting in the next round, that is, the second *a priori* preference in line 5 is activated. It results in that the program has only one abductive solution with *move(d)*. So IR will defect in the next round.

If the opponent is an *allc* or *tft*, both of which cooperate in the first round, then IR recognizes them as having intention of cooperating in the next round, that is, the first preference is activated. The EP program has only one abductive solution, containing *move(c)*, i.e. IR will cooperate in the next round.

We next consider a more complex intention-based decision making rule in the iterated PD, where IR recognizes the actual strategy of its co-player, envisaged as the intention to play in a way defined by that strategy.

Example 8 (Intention-based decision making rule, in [10]). Defect if the co-player's recognized intention or rule of behavior is always-cooperate (*allc*) or always-defect (*alld*), cooperate if it is *tft*; and if it is *wsls*, cooperate if in the last game state both cooperated (denoted by R) or both defected (denoted by P), and defect if the current game state is that IR defected and the co-player cooperated (denoted by T) or vice versa (denoted by S). Furthermore, IR also has the option of opting out when it is not confident about its co-player's intention [7]. This rule of behavior is learnt using a dataset collected from prior interactions with those strategies [10].

1. $abds([move/1])$.
2. $on\_observed(decide) \leftarrow new\_interaction$.
3. $decide \leftarrow move(c)$. $\quad decide \leftarrow move(d)$. $\quad \leftarrow move(c),\ move(d)$.
4. $expect(move(X)) \ \leftarrow \ has\_intention(co\_player, I, Pr),\ Pr > 0.7$.
5. $move(d) \ \triangleleft \ move(c) \leftarrow has\_intention(co\_player, allc)$.
   $move(d) \ \triangleleft \ move(c) \leftarrow has\_intention(co\_player, alld)$.
   $move(c) \ \triangleleft \ move(d) \leftarrow has\_intention(co\_player, tft)$
   $move(c) \ \triangleleft \ move(d) \leftarrow has\_intention(co\_player, wsls),$
   $game\_state(s),\ (s = \text{'R'}; s = \text{'P'})$.

$$move(c) \ \triangleleft \ move(d) \leftarrow has\_intention(co\_player, wsls),$$
$$game\_state(s),\ (s = \text{'T'}; s = \text{'S'}).$$

Similarly, at the start of a new interaction, an IR needs to choose a move, either cooperate or defect (lines 2-3). But differently, now any of such moves is expected to be made only if IR is confident enough about its co-player intention, i.e. the probability the most likely intention is greater than a given threshold [3], say 0.7 as in our example (line 4). The *a priori* preferences in line 5 state which move IR prefers to choose, given the recognized intention of the co-player (*allc, alld, tft* or *wsls*) and the current game state (T, R, P or S). More details on how this strategy interacts with others and its performance can be found in [10].

4 Conclusions and Future Work

We have summarized, generalized, and extended our previously LP-based implemented Evolution Prospection system—with several novel extensions for intention-based decision making—and have described several ways in which it can be useful for knowledge representation and reasoning in strategic scenarios, leading to a computational model of the basic dynamics of mental attitudes in such situations. Overall, it can take into account ongoing intentions of others, be triggered by new observations and ongoing recognized intentions of others, in order to attempt new goals and new preferences. A case in point, is to recognize the strategy intention of others by proposing a commitment to cooperate [7]. The presented computational logic based framework is general and expressive, suitable for intention-based decision making in the wide context of dynamic strategic scenarios. The declarative specification of dynamic strategies, such as we have illustrated, can be easily provided in more complex dynamic environments such as those with uncertainty, limited resources, unexpected events, and changing payoffs. Moreover, it can be integrated with other LP tools and functionalities, and provides a declarative rendering of decision making integrated with LP.

For future work, it appears that our diverse kinds of preferences and environment trigger constructs can be generally applied for knowledge representation and reasoning in different kinds of games. For instance, *a posteriori* (including evolution level) preferences can be applied to extensive form games [13], whether with perfect or imperfect information. Accounting for intentions of co-players can further improve the reasoning/decision making in more complex and dynamic game scenarios, such as multiplayer and evolutionary games [20]. We also plan to systematically compare our decision making framework with the existing ones, at least within those game theoretical settings.

Acknowledgements

The Anh Han acknowledges the support provided by the F.W.O. Belgium.

References

1. J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming*, 4(4):383–428, 2004.

2. C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.
3. N. Blaylock and J. Allen. Corpus-based, statistical goal recognition. In *Proceedings of 18th international joint conference on Artificial intelligence*, pages 1303–1308, 2003.
4. M. E. Bratman. *Intention, Plans, and Practical Reason*. The David Hume Series, 1987.
5. Robert H. Frank, Thomas Gilovich, and Dennis T. Regan. The evolution of one-shot cooperation: An experiment. *Ethology and Sociobiology*, 14(4):247 – 256, 1993.
6. J.Y. Halpern. *Reasoning about Uncertainty*. MIT Press, 2005.
7. T. A. Han. *Intention Recognition, Commitments and Their Roles in the Evolution of Cooperation: From Artificial Intelligence Techniques to Evolutionary Game Theory Models*, volume 9 of *SAPERE series*. Springer Berlin / Heidelberg, 4 2013.
8. T. A. Han and L. M. Pereira. Intention-based decision making with evolution prospection. In L. Antunes and H.S. Pinto, editors, *Proceedings of 15th Portuguese Conference on Artificial Intelligence (EPIA'2011)*, pages 254–267. Springer LNAI 7026, 2011.
9. T. A. Han, L. M. Pereira, and F. C. Santos. Intention recognition promotes the emergence of cooperation. *Adaptive Behavior*, 19(3):264–279, 2011.
10. T. A. Han, L. M. Pereira, and F. C. Santos. Corpus-based intention recognition in cooperation dilemmas. *Artificial Life journal*, 18(4):365–383, 2012.
11. T. A. Han, L. M. Pereira, and F. C. Santos. The emergence of commitments and cooperation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'2012)*, pages 559–566. ACM, 2012.
12. T. A. Han, C. K. Ramli, and C. V. Damásio. An implementation of extended P-log using XASP. In *Proceedings of International Conference on Logic Programming (ICLP'08)*, pages 739–743. Springer LNCS 5366, 2008.
13. Martin J. Osborne. *An introduction to game theory*. Oxford University Press, 2004.
14. L. M. Pereira, P. Dell'Acqua, A. M. Pinto, and G. Lopes. Inspecting and preferring abductive models. In *Handbook on Reasoning-based Intelligent Systems*. World Scientific Publishers, 2013.
15. L. M. Pereira and T. A. Han. Evolution prospection in decision making. *Intelligent Decision Technologies*, 3(3):157–171, 2009.
16. L. M. Pereira and T. A. Han. Intention recognition via causal Bayes networks plus plan generation. In *Proceedings of 14th Portuguese International Conference on Artificial Intelligence (EPIA'09)*, pages 138–149. Springer LNAI 5816, October 2009.
17. L. M. Pereira and T. A. Han. Intention recognition with evolution prospection and causal Bayesian networks. In *Computational Intelligence for Engineering Systems 3: Emergent Applications*, pages 1–33. Springer, 2011.
18. Sina Radke, Berna Guroglu, and Ellen R. A. de Bruijn. There's something about a fair split: Intentionality moderates context-based fairness considerations in social decision-making. *PLoS ONE*, 7(2):e31491, 02 2012.
19. Rubén J. Requejo and Juan Camacho. Evolution of cooperation mediated by limiting resources: Connecting resource based models and evolutionary game theory. *Journal of Theoretical Biology*, 272(1):35–41, 2011.
20. Karl Sigmund. *The Calculus of Selfishness*. Princeton University Press, 2010.
21. M. Tomasello. *Origins of Human Communication*. MIT Press, 2008.
22. M. van Hees and O. Roy. Intentions and plans in decision and game theory. In *Reasons and intentions*, pages 207–226. Ashgate Publishers, 2008.
23. XSB. The XSB system version 3.2 vol. 2: Libraries, interfaces and packages, March 2009.