Logic Programming Agents Playing Games

Marina De Vos
Department of Computer Science, University of Bath
Bath, UK

Dirk Vermeir
Department of Computer Science, Vrije Universiteit Brussel (VUB)
Brussels, Belgium

Abstract

We present systems of logic programming agents (LPAS) to model the interactions between decision-makers while evolving to a conclusion. Such a system consists of a number of agents connected by means of unidirectional communication channels. Agents communicate with each other by passing answer sets obtained by updating the information received from connected agents with their own private information. As an application, we show how extensive games with perfect information can be conveniently represented as logic programming agent systems, where each agent embodies the reasoning of a game player, such that the equilibria of the game correspond with the semantics agreed upon by the agents in the LPAS.

1 Introduction

In this paper we present a formalism for systems of logic programming agents. Such systems are useful for modeling decision-problems, not just the solutions of the problem at hand but also the evolution of the beliefs of and the interactions between the agents.

A system of logic programming agents consists of a set of agents connected by means of unidirectional communication channels. Each agent contains an ordered choice logic program [3] representing her personal information and reasoning skills. Agents use information received from their incoming channels as input for their reasoning, where received information may be overridden by other concerns represented in their program. The resulting model is communicated to the agents listening on the outgoing channels. The semantics of the whole system corresponds to a stable situation where no agent needs to change its output.

To model a single agent's reasoning, we use ordered choice logic programs[3], an extension of logic programming which provides facilities for the direct representation of preference between rules and dynamic choice between alternatives.

Game theory [5] makes contributions to many different fields. In particular, there is a natural connection with multi-agent systems. In this paper we illustrate the use of logic programming agent systems as convenient executable representations of games, where each player corresponds with exactly one agent. We concentrate on so-called extensive games with perfect information: a sequential communication structure of players taking decisions, based on full knowledge of the past. We demonstrate that

such games have a constructive and intuitive translation to logic programming agent systems where the agents/players are connected in a cyclic communication structure. The game's equilibria (Nash or subgame perfect, depending on the transformation used to construct the corresponding system) can then be retrieved as the system's answer set semantics. Moreover, the fixpoint computation of the answer sets closely mirrors the actual reasoning of the players in reaching a conclusion corresponding to an equilibrium.

2 Choice Logic Programming

Choice logic programs [1] represent decisions by interpreting the head of a rule as an exclusive choice between alternatives.

Formally, a *Choice Logic Program* [1], CLP for short, is a finite set of rules of the form $A \leftarrow B$ where A and B are finite sets of ground atoms. Intuitively, atoms in A are assumed to be xor'ed together while B is read as a conjunction (note that A may be empty, i.e. constraints are allowed). The set A is called the head of the rule r, denoted H_r, while B is its body, denoted B_r. In examples, we often use "$\oplus$" to denote exclusive or, while "," is used to denote conjunction.

The *Herbrand base* of a CLP P, denoted $\mathcal{B}_P$, is the set of all atoms that appear P. An *interpretation* is a consistent[1] subset of $\mathcal{B}_P \cup \neg\mathcal{B}_P$. For an interpretation I, we use I^+ to denote its positive part, i.e. $I^+ = I \cap \mathcal{B}_P$. Similarly, we use I^- to denote the negative part of I, i.e. $I^- = \neg(I \cap \neg\mathcal{B}_P)$. An atom a is *true* (resp. *false*) w.r.t. to an interpretation I for a CLP P if $a \in I^+$ (resp. $a \in I^-$). An interpretation is *total* iff $I^+ \cup I^- = \mathcal{B}_P$. The positive complement of an interpretation I, denoted $\overline{I}$, equals $\mathcal{B}_P \setminus I^+$.

A rule r in a CLP is said to be *applicable* w.r.t. an interpretation I if $B_r \subseteq I$. Since we are modeling choice, we have that r is *applied* when r is applicable and $|H_r \cap I| = 1$[2]. A rule is *satisfied* if it is applied or not applicable. A *model* is defined in the usual way as a total interpretation that satisfies every rule. A model M is said to be *minimal* if there does not exist a model N such that $N^+ \subset M^+$.

The *Gelfond-Lifschitz transformation* for a CLP P w.r.t. an interpretation I is the positive logic program P^I obtained from P by first removing all false atoms from the head of each choice rule (e.g. a rule with more than one head atom). Afterwards, all remaining choice rules r are replaced with the constraints $\leftarrow H_r, B_r$. These constraints force P^I to reject interpretations that contain two true atoms that are both in the head of an applicable choice rule. A total interpretation M is a *stable model* for P iff M is a minimal model for P^M. For choice logic programs the stable model and the minimal model semantics coincide[1].

The following example is a variation on the well-known prisoner's dilemma [5].

Example 1 (Eternal Enemies) *One day two eternal enemies, the lions and the hyaena's, meet on the African plains. Today's cause of argument is a juicy piece of meat. Both the lions and the hyaena's are keen on devouring it. To obtain their share there are two*

[1] For a set of literals X, we use $\neg X$ to denote $\{\neg a \mid a \in X\}$, where $\neg\neg a = a$ for any atom a. X is consistent iff $X \cap \neg X = \emptyset$.

[2] For a a set X, we use $|X|$ do denote its cardinality.

possibilities: they can either divide the piece among the members of both groups or they can fight for it with the risk of getting injured. Knowing the other group's temper it is always best to attack: either both parties are willing to fight or the peace-loving group will be chased away. However, both parties know that they get the most meat without any risk if they are both willing to share. Despite this, no pride is willing to take the risk of losing out on this free lunch. The following simple choice logic program models this eternal feud.

$$
\begin{array}{lcll}
share_{lions} & \oplus & fight_{lions} & \leftarrow \\
share_{hyaenas} & \oplus & fight_{hyaenas} & \leftarrow \\
 & & fight_{lions} & \leftarrow share_{hyaenas} \\
 & & fight_{lions} & \leftarrow fight_{hyaenas} \\
 & & fight_{hyaenas} & \leftarrow share_{lions} \\
 & & fight_{hyaenas} & \leftarrow fight_{lions}
\end{array}
$$

The program from Example 1 has one stable model $\{fight_{lions}, fight_{hyaenas}\}$*, which explains why the two species remain enemies: neither wants to give sharing a try as they fear that the other will take advantage by attacking.*

3 Ordered Choice Logic Programming

An ordered choice logic program (OCLP) [2] is a collection of choice logic programs, called components, each representing a portion of information. The relevance or preciseness of each component with respect to the other components is expressed by a strict pointed partial order[3].

Definition 1 *An **Ordered Choice Logic Program**, or OCLP, is a pair* $P = \langle \mathcal{C}, \prec \rangle$ *where* $\mathcal{C}$ *is a finite set of finite choice logic programs*[4]*, called **components**, and "*$\prec$*" is a strict pointed partial order on* $\mathcal{C}$*. We use* $P^{\cup}$ *to denote the CLP obtained from* P *by joining all components, i.e.* $P^{\cup} = \cup_{c \in \mathcal{C}} c$*. For a rule* $r \in P^{\cup}$*,* $c(r)$ *denotes the component from which the rule was taken (i.e. we assume that rules are labeled by the component)*[5]*. The Herbrand base of an OCLP* P *is defined by* $\mathcal{B}_P = \mathcal{B}_{P^{\cup}}$*. An **interpretation** of* P *is an interpretation of* $P^{\cup}$*. A rule* r *in an OCLP* P *is applicable, resp. applied, w.r.t. an interpretation* I*, if it is applicable, resp. applied in* $P^{\cup}$*, w.r.t.* I*.*

For two components $C_1, C_2 \in \mathcal{C}$, $C_1 \prec C_2$ implies that C_2 contains more general, or less preferred, information than C_1. Throughout the examples, we will often represent an OCLP P by means of a directed acyclic graph in which the nodes represent the components and the arcs the $\prec$-relation.

[3] A relation $<$ on a set A is a strict partial order iff $<$ is anti-reflexive, anti-symmetric and transitive. $<$ is pointed if there is an element $a \in A$ such that $a < b$ for all $b \in A \setminus \{a\}$.

[4] We assume that all rules are grounded; i.e. any rule containing variables has been replaced with all its ground instances.

[5] In fact, the same rule could appear in two components and thus $P^{\cup}$ should be a set of labeled rules. We prefer to work with the present simpler notation and note that all results remain valid in the general case.

Example 2 *This year, the choice for the holidays has been reduced to a city trip to London or a fortnight stay in either Spain or Mexico. A city trip to London is rather short and Mexico is expensive. With a larger budget however, we could have both a holiday in Mexico and a trip to London. Given these considerations, there are two possible outcomes:*

- *we have a small budget and we should opt for Spain, or*
- *with a larger budget, we can combine Mexico and London.*

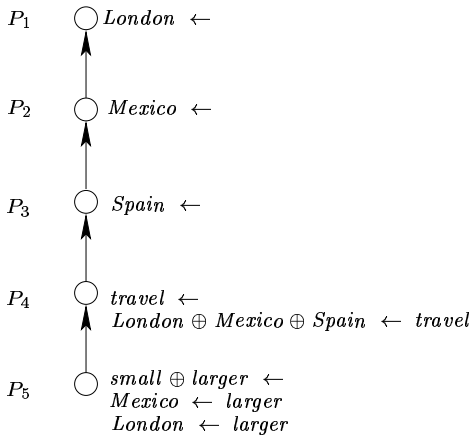


Figure 1 *The travel OCLP of Example 2*

This decision problem can be conveniently represented as an OCLP, as displayed by Figure 1. The rules in the components $P_1 \ldots P_3$, express the preferences in case of a small budget. The rules in P_4 explain that we want to travel and, because of this, we need to make a decision concerning our destination. In component P_5, the first rule states that there is also the possibility of a larger budget. In this case, the two other rules in this component tell us that we can have both London and Mexico.

The sets: $I = \{Mexico, small, \neg Spain\}$, $J = \{travel, Mexico, small, \neg London, \neg Spain, \neg larger\}$, $K = \{travel, Spain, small, \neg larger, \neg London, \neg Mexico\}$, and $L = \{travel, London, Spain, Mexico, larger, \neg small\}$ are all interpretations for this OCLP. The interpretation I makes the rule $small \oplus larger \leftarrow$ applied while the rule $London \leftarrow$ is applicable but not applied. While J, K and L are total, I is not.

When it is clear from the context that only total interpretations are considered, we will omit the negative part. If necessary it can be retrieved by means of the positive complement.

A decision involves a choice between several alternatives. In a CLP, decisions are generated by so-called *choice rules*, i.e. rules with multiple head atoms. Thus, for an interpretation I of a CLP, a and b are alternatives if they appear together in the head of an applicable rule. For an ordered program, we will use a similar notion which takes the preference order into account. Intuitively, for a component C, a and b are alternatives w.r.t. an interpretation I if there is an applicable choice rule containing a and b in the head, in a component that is at least as preferred as C.

Definition 2 *Let $P = \langle \mathcal{C}, \prec \rangle$ be an OCLP, let I be an interpretation and let $C \in \mathcal{C}$. The set of **alternatives** in C for an atom $a \in \mathcal{B}_P$ w.r.t. I, denoted $\Omega_C^I(a)$, is defined as*[6]*: $\Omega_C^I(a) = \{b \mid \exists r \in P^{\cup} \cdot c(r) \preccurlyeq C \ \wedge \ B_r \subseteq I \ \wedge \ a, b \in H_r \text{ with } a \neq b\}$.*

[6] $\preccurlyeq$ is the reflexive closure of $\prec$.

Example 3 *Reconsider the interpretations I and J from Example 2. The alternatives for $Mexico$ in P_2 w.r.t. J are $\Omega^J_{P_2}(Mexico) = \{Spain, London\}$. With respect to I we obtain $\Omega^I_{P_2}(Mexico) = \emptyset$, since the choice rule in P_3 is not applicable. When we take P_5 instead of P_2, we obtain w.r.t. J: $\Omega^J_{P_5}(Mexico) = \emptyset$.*

Atoms that are each others' alternative w.r.t. a certain interpretation I will continue to be so in any extension $J \supseteq I$. In this sense, Ω_P is a monotonic operator.

Although rules do not contain negations, they can still conflict. E.g. one rule could force a choice between a and b while other rules could force a and b separately. More generally, a conflict exists for a rule r, which is applicable w.r.t. an interpretation I, if for all $a \in H_r$, there exists another rule r_a such that $H_{r_a} \subseteq \Omega^I_{c(r)}(a)$.

As in [4], we use the preference relation among the components to ignore rules that are *defeated* by more preferred rules forcing different alternatives.

Definition 3 *Let I be an interpretation for an OCLP P. A rule $r \in P^\cup$ is **defeated** w.r.t. I iff*

$$\forall a \in H_r \cdot \exists r' \in P^\cup \cdot c(r) \not\prec c(r') \ \wedge \ r' \text{ is applied w.r.t. } I \ \wedge \ H_{r'} \subseteq \Omega^I_{c(r)}(a) \ .$$

*The rule r' is called a **defeater** w.r.t. I. I is a **model** of P iff every rule in $P^\cup$ is either not applicable, applied or defeated w.r.t. I. A model M is **minimal** iff its positive part is minimal according to set inclusion, i.e. no model N of P exists such that $N^+ \subset M^+$.*

The above definition defines a approach where a rule can be defeated by applied rules that are not less preferred as the rule at hand. This approach can be seen as credulous, as a random choice is made between two equally or unrelated alternatives. A more skeptical approach would demand that the rules are related and the defeater(s) is (are) strictly more preferred.

Example 4 *Reconsider the interpretations J and L defined in Example 2. The rule $London \leftarrow$ is defeated w.r.t. J by the rule $Mexico \leftarrow$. The combination of the rules $Mexico \leftarrow larger$ and $London \leftarrow larger$ defeats the rule $London \oplus Mexico \oplus Spain \leftarrow$ is w.r.t. L. Only K and L are models. Model L is not minimal due to the smaller model $Z = \{travel, larger, Mexico, London, travel, \neg Spain, \neg small\}$. The minimal models K and Z correspond to the intuitive outcomes of the problem.*

For ordered programs, the minimal semantics sometimes yields unintuitive results, as demonstrated in the following example.

Example 5 *Consider the program $P = \langle\{c_1, c_2, c_3\}, \prec\rangle$ where $c_1 = \{a \leftarrow\}$, $c_2 = \{b \leftarrow\}$, $c_3 = \{a \oplus b \leftarrow c\}$ and $c_3 \prec c_2 \prec c_1$. The minimal models are $\{a,b\}$, where no choice between a and b is forced, and $\{c,b\}$. The latter is not intuitive due to the gratuitous assumption of c.*

Unwarranted assumptions as in Example 5 can be avoided by adopting an answer set semantics, where we use a variant of the Gelfond-Lifschitz transformation to map an OCLP to an unordered CLP.

Definition 4 *Let M be a total interpretation for an OCLP P. The **reduct** for P w.r.t. M, denoted P^M, is the choice logic program obtained from $P^{\cup}$ by removing all defeated rules. M is called an **answer set** for P iff M is a stable model for P^M.*

Example 6 *The program P from Example 5 does not admit $N = \{a, b\}$ as an answer set, since $P^N = \{b \leftarrow,\ a \oplus b \leftarrow c\}$ which has only $\{b\} \neq N$ as a stable model. The minimal models K and Z of Example 4 are both answer sets.*

4 Logic Programming Agents

In this section we consider systems of communicating agents where each agent is represented by an OCLP that contains its knowledge about itself and other agents.

Agents communicate via unidirectional communication channels through which the conclusions derived by the agent at the source of the channel are passed on to the agents at the other end.

Definition 5 *A **logic programming agent system**, or LPAS, is a pair $F = \langle \mathbf{A}, \mathbf{C} \rangle$ where $\mathbf{A}$ is a set of agents a and $C \subset \mathbf{A} \times \mathbf{A}$ is an anti-reflexive relation representing the communication channels between agents. Moreover, each agent $a \in \mathbf{A}$ is associated with an ordered choice logic program $F_a = \langle \mathcal{C}_a, \prec_a \rangle$.*

We will use a more convenient graph-like notation in our examples.

Example 7 *Two witnesses discover a body lying in the park. The first witness tells the local police that she saw hair near the victim and that she did not see any blood. The second witness testifies that she saw blood and that the victim had strange bite marks. The sheriff states that this situation is a clear case of murder and passes it to the FBI. Because of the strange appearance of bite-marks and hair, the FBI passes the case to the special X-cell. In addition, the FBI states that, if the X-cell reports that a werewolf is involved, the case should be classified. Given the evidence, the X-file team has no choice but to decide that the killing was indeed done by a werewolf. This situation is represented by the LPAS depicted in Figure 2.*

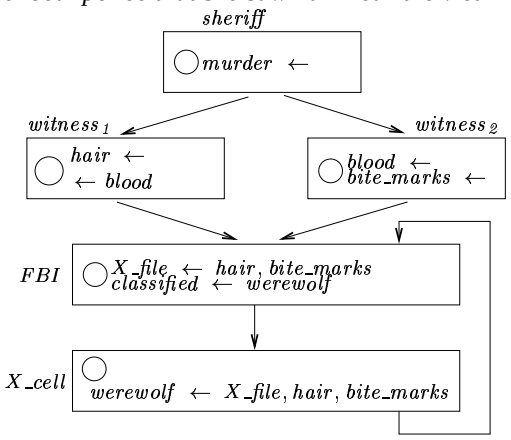


Figure 2 *The werewolf-killing of Example 7*

The Herbrand base of a LPAS is the union of all the Herbrand bases of the ordered choice logic programs used by the agents. An interpretation assigns a set of literals to each agent in the system. These literals may be concluded by the agent itself, based on

input received through an input channel, or they may simply be accepted from other agents via an input channel.

Definition 6 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be an LPAS. The **Herbrand base** of F, denoted $\mathcal{B}_F$, equals $\mathcal{B}_F = \bigcup_{A \in \mathbf{A}} \mathcal{B}_A$. An **interpretation** of F is a function $I : \mathbf{A} \rightarrow \mathbf{2}^{(\mathcal{B}_\mathbf{F} \cup \neg \mathcal{B}_\mathbf{F})}$ that associates a consistent set of literals (beliefs) to each agent.*
*Given an interpretation I, the **inputs** and **outputs** of each agent are defined by $In_I(a) = cons(\cup_{(b,a) \in C} I(b))$ and $Out_I(a) = I(a)$, respectively, where $cons(X) = X^+ \setminus (X^+ \cap X^-)$, i.e. the maximal positive consistent part of X.*

Thus, an agent sends its full set of beliefs over all outgoing communication channels. On the other hand, an agent receives as input, the beliefs of all agents connected to its incoming channels. If two agents send conflicting information to a receiving agent, the conflicts are removed.

Example 8 *Consider the Werewolf LPAS F of Example 7. We define the interpertation I of F as:*

$$\begin{array}{lcl} I(sheriff) & = & \{murder\} \\ I(witness_1) & = & \{murder, hair, \neg blood\} \\ I(witness_2) & = & \{murder, blood, bite\_marks\} \\ I(FBI) & = & \{murder, hair, bite\_marks, werewolf.X\_file, classified\} \\ I(X\_cell) & = & \{murder, hair, bite\_marks, werewolf, X\_file, classified\} \end{array}$$

The input of agent FBI w.r.t. I equals $In_I(FBI) = \{murder, hair, bite\_marks\}$. The output produced by the X_cell-agent w.r.t. I_2 is $Out_I(X\_cell) = \{murder, hair, bite\_marks, werewolf.X\_file, classified\}$.

An agent reasons on the basis of positive information that is received from other agents (its input) and its own program that may be used to draw further conclusions, possibly overriding incoming information. Hence, agents attach a higher preference to their own rules rather than to suggestions coming from outside.

This can be conveniently modeled by extending an agent's ordered program with an extra "top" component containing the information gathered from its colleagues. This way, the OCLP semantics will automatically allow for defeat of incoming information that does not fit an agent's own program.

Definition 7 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be a LPAS. The **updated version** of an agent $a \in \mathbf{A}$, with program $F_a = \langle \mathcal{C}_a, \prec_a \rangle$, w.r.t. a set of atoms $U \subseteq \mathcal{B}_F$, denoted a^U, is defined by $a^U = \langle \mathcal{C}_a \cup \{c_U\}, \prec_a \cup \{c < c_U \mid c \in \mathcal{C}_a\} \rangle$ with $c_U = \{l \leftarrow \mid l \in U\}$.*

For an interpretation to be a model, it suffices that each agent produces a local model (output) that is consistent with its input.

Definition 8 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be a LPAS. An interpretation I of F is a **model** iff $\forall a \in \mathbf{A} \cdot Out_I(a)$ is an answer set of $a^{In_I(a)}$.*

Example 9 *Reconsider the Werewolf LPAS of Example 7 and its interpretation I from Example 8. It is easy to see that I is a model. Even more, it is the only model.*

For systems without cycles the above model semantics will generate rational solutions for the represented decision-problems. The next example demonstrates that systems that do have cycles may have models that contain too much information, because assumptions made by one agent may become justified by another agent.

Example 10 *Two children have been listening to a scary story about vampires and zombies. Suddenly, they think something moved in the room and they start fantasizing about the story they just heard. The first child says that, in order to have a real zombie or vampire, the creature should be dead and be walking around. The second child agrees that vampires and zombies are both dead but still able to walk around.*

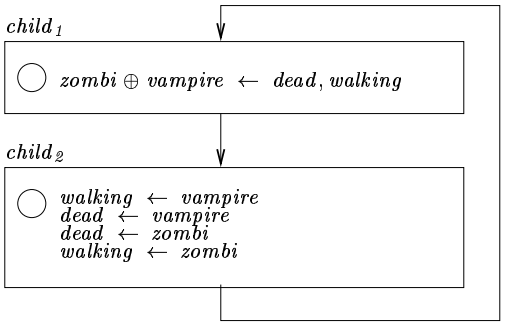


Figure 3 *The imagination LPAS of Example 10*

The situation is represented by the LPAS in Figure 3. This system has three models with $M_1(child_1) = M_1(child_2) = \{\neg vampire, \neg zombi, \neg walking, \neg dead\}$, *and* $M_2(child_1) = M_2(child_2) = \{zombi, walking, dead, \neg vampire\}$ *; and* $M_3(child_1) = M_3(child_2) = \{vampire, walking, dead, \neg zombi\}$ *. The last two models are not realistic, since the children are just giving a description.*

To avoid such self-sustaining propagation of assumptions, we will demand that a model be the result of a fixpoint procedure which mimics the evolution of the belief set of the agents over time.

Definition 9 *Let* $F = \langle \mathbf{A}, \mathbf{C} \rangle$ *be a LPAS. A sequence of interpretations* $I_0 \ldots I_n$ *is an* ***evolution*** *of* F *iff for any* $i \geq 0$, $a \in \mathbf{A}$, $I_{i+1}(a)$ *is a model of* $a^{In_{I_i}(a)}$. *An* ***evolutionary fixpoint*** *of an interpretation* I_0 *is any interpretation* I *that can be repeated forever in an evolution* $I_0 \ldots I_n = I,\ I_{n+1} = I$. *An* ***answer set*** *of* F *is any c-evolutionary fixpoint of* $I_\emptyset$.

Thus, in an evolution, the agents evolve as more information becomes available: at each phase of the evolution, an agent updates her program to reflect input from the last phase and computes a new set of beliefs. An evolution thus corresponds to the way decision-makers try to get a feeling about the other participants. The process of reaching a fixpoint boils down to trying to get an answer to the question "if I do this, how would the other agents react", while trying to establish a stable compromise. Note that the notion of evolution is nondeterministic since an agent may have several local models. For a fixpoint, it suffices that each agent can maintain the same set of beliefs as in the previous stage.

Example 11 *Consider the Werewolf LPAS of Example 7. The interpretation I described in Example 8 is an answer set of the LPAS. The vampire-zombie LPAS of Example 10 has one answer set I, where*

$$I(child_1) = I(child_2) = \{\neg zombie, \neg vampire, \neg walking, \neg dead\} \ .$$

Theorem 1 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be a LPAS. An interpretation I is a model for F iff it is an evolutionary fixpoint of F.*

Corollary 1 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be a LPAS. Every answer set of F is a (model of F.*

The reverse of the above corollary does not hold in general. A counter example is given in Example 10. However, for acyclic LPAS a one-to-one mapping does exist.

Theorem 2 *Let $F = \langle \mathbf{A}, \mathbf{C} \rangle$ be a LPAS without cycles. An interpretation M is a stable model (resp. answer set) iff I is a model (resp. c-model) of F.*

5 LPAS and Game Theory

In this section we demonstrate that extensive games with perfect information have a natural formalization as logic programming agent systems. The equilibria of such games can be obtained as the answers sets of the system, where each agent represents a player, and the evolution mimics the mechanism players can use in order to come to a decision.

5.1 Extensive Games with Perfect Information

An extensive game is a detailed description of a sequential structure representing the decision problems encountered by agents (called *players*) in strategic decision making (agents are capable to reason about their actions in a rational manner). The agents in the game are perfectly informed of all events that previously occurred. Thus, they can decide upon their action(s) using information about the actions which have already taken place. This is done by means of passing *histories* of previous actions to the deciding agents. *Terminal histories* are obtained when all the agents/players have made their decision(s). Players have a preference for certain outcomes over others. Often, preferences are indirectly modeled using the concept of *payoff* where players are assumed to prefer outcomes where they receive a higher payoff.
Summarizing, an extensive game with perfect information, [5]), is 4-tuple, denoted $\langle N, H, P, (\succcurlyeq_i)_{i \in N} \rangle$, containing the players N of the game, the histories H, a player function P telling who's turn it is after a certain history and a preference relation $\leq_i$ for each player i over the set of terminal histories.
For examples, we use a more convenient representation: a tree. The small circle at the top represents the initial history. Each path starting at the top represents a history. The terminal histories are the paths ending in the leafs. The numbers next to nodes represent the players while the labels of the arcs represent an action. The numbers below the terminal histories are payoffs representing the players' preferences (The first number is the payoff of the first player, the second number is the payoff of the second player, ...).

Example 12 *The game depicted in Figure 4 models an individuals' predicament in the following situation: two ladies have decided that they want fruit cake for dessert. There are two possibilities: they either bake a cake or they buy one. At the bakery shop one can choose between strawberry and cherry cake. For strawberry cake there is the possibility to have whipped cream on top. They agree that the first lady will decide on how to get the cake and, if necessary, whether a topping is wanted or not. The second lady will be picking the type of fruit cake.*

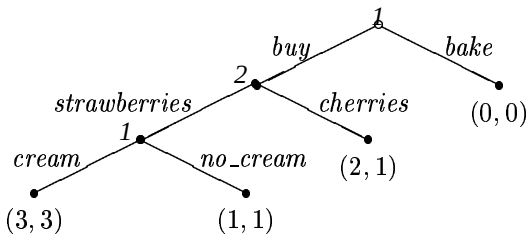


Figure 4 *The Cake-game of Example 12*

A *strategy* of a player in an extensive game is a plan that specifies the actions chosen by the player for every history after which it is her turn to move. A *strategy profile* contains a strategy for each player.
The first solution concept for an extensive game with perfect information ignores the sequential structure of the game; it treats the strategies as choices that are made once and for all before the actual game starts. A strategy profile is a *Nash equilibrium* if no player can unilaterally improve upon his choice. Put in another way, given the other players' strategies, the strategy stated for the player is the best this player can do[7].

Example 13 *The game of Example 12 has two Nash equilibria:*

$$\{\{buy, cream\}, \{strawberries\}\} \text{ and } \{\{buy, no\_cream\}, \{cherries\}\} \ .$$

Although the Nash equilibria for an extensive game with perfect information are intuitive, they have, in some situations, undesirable properties due to not exploiting the sequential structure of the game. These undesirable properties are illustrated by the next example.

Example 14 *Being a parent can sometime be hard. Especially when your child asks for a pet. His two favorite animals are cats and spiders and you really hate spiders. However, your son prefers the cat since it is more affectionate. The game corresponding to this situation is depicted in Figure 5. This game has three Nash equilibria* $\{\{no\_pet\}, \{cat\}\}$, $\{\{no\_pet\}, \{spider\}\}$ *and* $\{\{pet\}, \{cat\}\}$.
The strategy profile $\{\{no\_pet, spider\}\}$ *is an unintuitive Nash equilibrium since it is sustained by the threat that the child would opt for the spider when a pet was allowed. However, the child would never go for a spider (payoff 1) since a cat is more playful (payoff 2).*

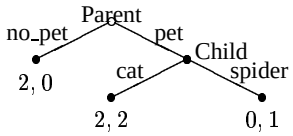


Figure 5 *The Spider Threat of Example 14*

[7] Note that the strategies of the other players are not actually known to i, as the choice of strategy has been made before the play starts. As stated before, no advantage is drawn from the sequential structure.

Because players are informed about the previous actions they only need to reason about actions taken in the future. This philosophy is represented by subgames. A *subgame* is created by pruning the tree in the upwards direction. So, intuitively, a subgame represent a stage in the decision making process where irrelevant and already known information is removed. Instead of just demanding that the strategy profile is optimal at the beginning of the game, we require that for a *subgame perfect equilibrium* the strategy is optimal after every history. In other words, for every subgame, the strategy profile, restricted to this subgame, needs to be a Nash equilibrium. This can be interpreted as if the players revise their strategy after every choice made by them or another player. Therefor, subgame perfect equilibria eliminate Nash equilibria in which the players' threats are not credible.

Example 15 *Reconsider the game of Example 14:* $\{\{pet\},\{cat\}$ *and* $\{\{no\_pet\},\{cat\}$ *are the only subgame perfect equilibria. The unintuitive Nash equilibrium* $\{\{no\_pet\},$ $\{spider\}$ *is no longer accepted. The Cake-game of Example 12 admits only one subgame perfect equilibrium:* $\{\{buy, cream\},\{strawberries\}\}$.

5.2 Playing Games

We demonstrate that extensive games with perfect information have a natural formulation as multi-agent systems with a particularly simple information-flow structure between the agents. For our mapping, we assume that an action can only appear once[8]. This is not really a restriction, since one can simply use different names for these actions since they are not related. This will just have an effect on the syntax of the game and not on its semantics.

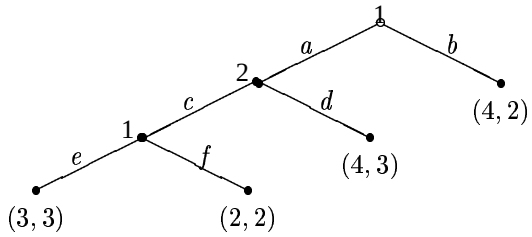


Figure 6 *The game of Example 16*

Example 16 *Consider the extensive game depicted in Figure 6. This game has six Nash equilibria:* $\{\{b,e\},\{c\}\}, \{\{b,f\},\{c\}\}, \{\{b,e\},\{d\}\}, \{\{b,f\},\{d\}\}, \{\{a,e\},$ $\{d\}\}, \{\{a,f\},\{d\}\}$. *Three of these Nash equilibria are also subgame perfect equilibria:* $\{\{b,e\},\{c\}\}, \{\{b,e\},\{d\}\}, \{\{a,e\},\{d\}\}$.

The following transformations will be used to retrieve the Nash equilibria and subgame perfect equilibria from the game as the answer sets of the corresponding OCLP.

Definition 10 *Let* $\langle N, H, P, (\succcurlyeq_i)_{i\in N}\rangle$ *be a finite extensive game with perfect information. The corresponding Nash LPAS* $S^n = \langle\{A^i \mid i \in N\}, C\rangle$ *with* $S^n_{A^i} = \langle\{\mathcal{C}_{A^i}\}, \prec_i\rangle$ *constructed as follows:*

1. $\mathcal{C}_{A^i} = \{C^i_u \mid \exists h \in Z \cdot u = U_i(h)\}$;

2. $\forall C^i_u, C^i_w \in \mathcal{C}_{A^i} \cdot C^i_u \prec_i C^i_w$ *iff* $u > w$;

3. $\forall h \in (H \setminus Z), P(h) = i \cdot (A(h) \leftarrow) \in C^i_w, \forall C^i_n, n \neq w \cdot C^i_w \prec_i C^i_n$;

4. $\forall h = h_1 a h_2 \in Z, P(h_1) = i \cdot a \leftarrow B \in C^i_u$ *with*
$B = \{b \in [h]^9 \mid h = h_3 b h_4, P(h_3) \neq i\}$ *and* $u = U_{P(h_1)}(h)$,

5. $C(A^i) = \{A^{i+1}\}$ *for* $i \in N, i < \max N$,

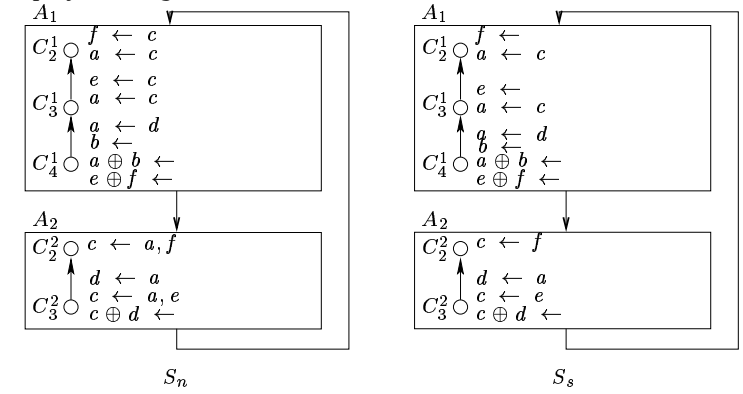
6. $C(A^{\max N}) = \{A^1\}$.

The corresponding subgame LPAS $S^s = \langle \{A^i \mid i \in N\}, C\rangle$ *where* $S^s_{A^i} = \langle \{\mathcal{C}_{A^i}\}, \prec_i\rangle$ *is defined in the same way as the Nash variant apart from step 4 which becomes:*

$4'$. $\forall h = h_1 a h_2 \in Z, P(h_1) = 1 \cdot (a \leftarrow B) \in C^i_u$ *with*
$B = \{b \in [h2] \mid h = h_3 b h_4, P(h_3) \neq i\}$ *and* $u = U_{P(h_1)}(h)$.

Intuitively an agents is created for each player in the game. The OCLP of such an agents contains as many component as the represented player has payoffs (step 1.). The order among the components follows the expected payoff, higher payoffs correspond to more specific components (step 2). The various actions a player can choose from a certain stage of the game are turned into a choice rule which is placed in the most specific component of the agent modelling the player making the decision (step 3). Since Nash equilibria do not take into account the sequential structure of the game, players have to decide upon their strategy before starting the game, leaving them, for each decision, to reason about both past and future. This is reflected in the rules (step 4): each non-choice rule is made out of a terminal history (path from top to bottom in the tree) where the head represents the action taken by the player/agent, when considering the past and future created by the other players according to this history. The component of the rule corresponds to the payoff the deciding player would receive in case the history was actually followed. When considering subgame perfect equilibria, we know that they do take previous actions into account, making unnecessary to reason about the past. This is reflected in step 4'. Steps 5 and 6 establish the communication between the agents in the system.

Example 17 *For the game of Example 16, the corresponding LPAS's* S^n *and* S^s *are displayed in Figure 7.*



[9] We use $[h]$ to denote the set of actions appearing in a sequence h.

Figure 7 *The LPAS's of Example 17*

Notice that the answer sets for S^n match exactly the Nash equilibria of the game, while the subgame perfect equilibria can be retrieved using the answer sets of S^s.

Theorem 3 *Let $\langle N, H, P, (\succcurlyeq_i)_{i \in N} \rangle$ be a finite extensive game with perfect information and let S^n and S^s be the corresponding LPAS's, according to Definition 10. Then, s^* is a Nash equilibrium (resp. subgame perfect equilibrium) for $\langle N, H, P, (\succcurlyeq_i)_{i \in N} \rangle$ iff the interpretation I with $I(a) = s^*$ for every $a \in \mathbf{A}$ is an answer set for S^n (resp. S^s).*

For the proof of the above theorem, we demonstrate that every c-evolutionary fixpoint of $I_\emptyset$ can be constructed in n iterations, with n the number of players in the game. Of course, this only happens when players or agents know which actions will lead to an equilibrium state. In practice, it might take more iterations in order to find a fixpoint. Such a fixpoint computation can easily be seen as the players trying to obtain actions belonging to an equilibrium state. At first, she picks an action and sees how the other players respond to this. With this information she can update her actions. This process is carried on until a equilibrium is reached.

6 Relationship to Other Approaches

In this section we investigate the relationship of our approach with other formalisms. We restrict to the relationship with Game Theory, logic and agents. For a comparison with other preference based systems we refer to [2].

In the previous section we have demonstrated that OCLPs and multi-agent systems based on OCLPs provide a way to represent extensive games with perfect information in such a way that, depending on the transformation, either the Nash or subgame perfect equilibria can be retrieved as the stable models of the system. With the algorithm for the stable model computation of an OCLP, we immediately have an implementation for the equilibria of the game that the OCLP is representing. At the end of the previous section we already mentioned certain extensions to our proposed formalism in order to investigate other topics relevant to game theory, like for example information hiding and cheating. Probably the most important benefit for using logic programming for such research is its immediate return in the form of algorithms which allow for an efficient monitoring tool for the effects of the changes made.

Another aspect of logic programming that we already mentioned in the introduction, is its capability to represent more complex games. Take the Travel OCLP (Example 2) for example. If we just consider the first three components (P_1, P_2 and P_3), we see the representation of a very simple strategic or extensive game with a single player (the person who wants to travel). In this case the equilibrium would be $\{Spain\}$ which corresponds to the situation in which there are few dealers. From the moment that the dealers outnumber the officers two alternatives instead of just one are needed, which is impossible in game theory.

There are two main ways of relating logic and games: logic games and game logics. The former uses the games for the purpose of logic, while the latter uses logic

for the purpose of game theory. Detailed information about their history can be found in [8]. Our research belongs the category of game logic and, as far as we know, we are the only ones that look at game theory in the context of logic programming. The only exception might be [6], but he simply puts game theoretic features on top of his language. We, on the other hand, do not go outside the realm of logic programming to retrieve equilibria.

Some research has already been done in the area of agents and games, although with different viewpoints. For example, [7] investigates methods to prevent agents exploiting game theoretic properties of negotiations. [6] incorporates the players of the game directly into its logic programming formalism for strategic games in order to obtain mixed strategy Nash equilibria. We, on the other hand, are interested in multi-agent systems that are able to represent, in an intuitive way, games such that agents correspond with players and models with the equilibria.

References

[1] Marina De Vos and Dirk Vermeir. On the Role of Negation in Choice Logic Programs. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Logic Programming and Non-Monotonic Reasoning Conference (LPNMR'99), Lecture Notes in Artificial Intelligence*, pages 236–246, Springer Verslag, 1999.

[2] Marina De Vos and Dirk Vermeir. A Logic for Modelling Decision Making with Dynamic Preferences. In *Proceedings of the Logic in Artificial Intelligence (Jelia2000) workshop*, Lecture Notes in Artificial Intelligence, pages 391–406, Malaga, Spain, 2000. Springer Verslag, 2000.

[3] Marina De Vos and Dirk Vermeir. Logic Programming Agents and Game Theory. In *Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*, pages 27–33, American Association for Artificial Intelligence Pres, 2001.

[4] D. Gabbay, E. Laenens, and D. Vermeir. Credulous vs. Sceptical Semantics for Ordered Logic Programs. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 208–217, Morgan Kaufmann, 1991.

[5] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, third edition, 1996.

[6] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):7–56, 1997.

[7] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter. Designing Conventions for Automated Negotiation among Computers*. The MIT Press, 1994.

[8] Johan van Benthem. Logic and games. Online course notes of 1999, Stanford University.