Decisions, Agents and Games

Marina De Vos* and Dirk Vermeir

Dept. of Computer Science
Free University of Brussels, VUB
Pleinlaan 2, Brussels 1050, Belgium
Tel: +32 2 6293308
Fax: +32 2 6293525
{marinadv,dvermeir}@vub.ac.be
http://tinf2.vub.ac.be

Abstract. In this paper we present a framework for logic programming agents to take part in games in such a way that stable models of the system, the ones agreed upon by all the members, correspond with the different equilibria of the game. The proposed transformations from games to ordered choice logic program produce a multi-agent system where each agent embodies the reasoning specific to a single player and where the system itself represents the structure of the game. This allows us to monitor the knowledge and beliefs of the agents, i.e. the flow of information between agents/players.

1 Introduction

Game theory [OR96] makes contributions to many different fields. In particular, there is a natural connection with multi-agent systems.
Logic programming was originally conceived to provide a procedural interpretation of logical theories (e.g. sets of horn clauses) that corresponds to its declarative semantics. This combination of specification and execution proved very attractive for many application areas, see e.g. [NBG+01] for a recent example. The addition of "negation as failure" caused the above correspondence to break down, thus giving rise to alternative "declarative" readings of which the so-called answer set semantics is the most prominent, [Lif00].
In this paper we use logic programming to represent agents that exhibit game-theoretic behavior. 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 use an extension of logic programming, called Ordered Choice Logic Programs (OCLPs). The order is used to represent the agent's situation-dependent preferences. Reasoning about appropriate decisions is captured using "choice rules", which are essentially clauses where the disjunction in the head is interpreted as "exclusive or".

* Wishes to thank the FWO for its support.

We show that extensive games with perfect information have a natural formulation as multi-agent systems with a particularly simple information-flow structure between the agents, each of which is an OCLP. The stable model semantics of such a system is shown to coincide with the game's equilibria (Nash or subgame perfect, depending on the transformation used to construct the agents). Moreover, the fixpoint computation of a model closely mirrors the actual reasoning of a player in reaching a conclusion corresponding to an equilibrium.

2 Choice Logic Programming

In this section we introduce choice logic programming and its stable model semantics.

Definition 1. *A **choice logic program**, or CLP for short, is a finite set of rules of the form $A \leftarrow B$ where A and B are finite sets of atoms.*

Intuitively, atoms in A are assumed to be xor'ed together while B is read as a conjunction. The set A is called the *head* of the rules, denoted H_r, while the set B is the *body*, denoted B_r. In examples, we often use $\oplus$ to denote exclusive or, while "," is used to denote conjunction.

Example 1 (Prisoner's Dilemma). The following simple choice logic program models the well-known prisoner's dilemma.

$$
\begin{aligned}
\mathit{silent}_1 \oplus \mathit{confess}_1 &\leftarrow \\
\mathit{silent}_2 \oplus \mathit{confess}_2 &\leftarrow \\
\mathit{confess}_1 &\leftarrow \mathit{silent}_2 \\
\mathit{confess}_1 &\leftarrow \mathit{confess}_2 \\
\mathit{confess}_2 &\leftarrow \mathit{silent}_1 \\
\mathit{confess}_2 &\leftarrow \mathit{confess}_1
\end{aligned}
$$

The set of all atoms that appear in a program P is called its *Herbrand Base*, denoted $\mathcal{B}_P$. An interpretation indicates which atoms are assumed to be true in the current context. Atoms that do not appear in the interpretation are considered false.

Definition 2. *Let P be a CLP. An **interpretation** of P is any subset of $\mathcal{B}_P$.*

From an interpretation we can derive the truth value for the heads and bodies of the rules in a program. The body of a rule is true when all its elements are considered true while its head is true iff it contains exactly one true atom.

Definition 3. *Let I be an interpretation for a CLP P. A rule r is said to be **applicable** w.r.t. I iff $B_r \subseteq I$. A rule r is **applied** w.r.t. I iff r is applicable and $|H_r \cap I|^1 = 1$.*

[1] $|A|$ denotes the number of elements in the set A.

Models and minimal models are defined in the usual way.

Definition 4. *Let P be a CLP. An interpretation that makes each applicable in P applied is called a* ***model*** *of P.*
A model M is called ***minimal*** *when M is minimal[2] according to set inclusion.*

Most logic programming paradigms have some kind of stable model semantics which is based on a Gelfond-Lifschitz transformation. Choice logic programs also have a stable model semantics.

Definition 5. *Let P be a CLP and let I be an interpretation for it. The Gelfond-Lifschitz transformation from P w.r.t. I is the positive logic program P^I, obtained from P in the following way:*

- *remove all false atoms from the head of each rule with more than one head atom; and*
- *replace all remaining rules r with $|H_r| > 1$ (after deletion) with the following constraint[3]:*

$$\leftarrow B_r, H_r \ .$$

Just as for the original definition for logic programs with negation as failure (see [GL88]), we have that stable models are those total interpretations which are the minimal models, or Herbrand models, for their Gelfond Lifschitz transformation.

Definition 6. *Let M be a total interpretation for a CLP P. M is a stable model for P iff M is the minimal model for P^M.*

Thus, stable models are fixpoints: the program is transformed using the stable model and the minimal model of this new program is the original input.

Example 2. The program of example 1 has a single stable model, namely:

$$\{confess_1, confess_2\} \ .$$

Neither prisoner is willing to take the change of being convicted for a higher sentence by remaining silent about the crimes they have committed. This stable model corresponds exactly to the Nash equilibrium of this strategic game[4].

Theorem 1 ([DVV99]). *Let M be a total interpretation for a CLP P. M is a stable model for P iff M is a minimal model for P.*

The above result provides a more efficient procedure to check the minimality of a model M: instead of computing the other models, it suffices to check P^M.

[2] M is minimal iff there does not exist a model N of P such that $N \subset M$.
[3] A constraint is a rule with an empty head.
[4] See [DVV99] for more information about the relationship between strategic games and choice logic programming.

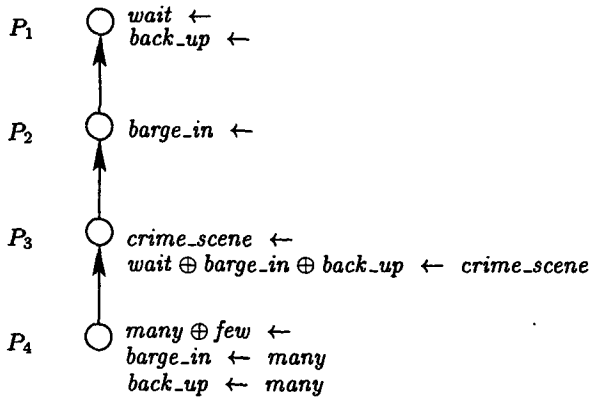

Fig. 1. Police tactics.

3 Ordered Choice Logic Programs

Ordered Choice Logic Programming [DVV00], or OCLP for short, finds its origin in decision making and knowledge representation. This formalism allows programmers to explicitly express decisions, i.e. exclusive choices between multiple alternatives, and to work with situation dependent preferences among the different alternatives of a decision. OCLP combines Choice Logic Programs [DVV99], for providing the rules to represent the decisions, and Ordered Logic Programs [GLV91], as the basic concept for the semantics.
Formally, an OCLP P is a pair $\langle C, \preccurlyeq \rangle$ where C is a collection of component CLPs. A component represents knowledge, including choices, with a certain amount of generality/preference. The generality/preference among components is expressed by the partial order relation $\preccurlyeq$. OCLPs are often represented using directed acyclic graphs in which the nodes are the components and the arcs represent the relation "$\prec$" (the strict version of $\preccurlyeq$).

Example 3. A police patrol discovers that a transfer of narcotics is taking place in an isolated warehouse. In these circumstances the police officers have three plausible options: they can wait until the gangsters come out, they can call for back-up or they can barge in. Given the situation it is the best to catch the dealers red-handed, so the third possibility, barging in, is preferred. This course of action is only safe if there are only a few gangsters. If not, the officers better call for back up as well as performing the roundup immediately. Intuitively, one would expect two possible scenarios for this roundup:

- there are only a few narcotic traders, so that it is sufficient that the police barges in to clear the crime scene; or
- there are simply too many gangsters, and the officers should call for back-up before they enter the crime scene.

The above considerations can be translated into the OCLP depicted in Fig. 1, where the choice rule in P_3 corresponds with the decision that the police officers need to make. The other rules in P_1,P_2, and P_3 give the preferences of the police officers in a normal situation. The choice rule in P_4 makes it clear that there might be many or just a few dealers. The other rules in P_4 give the police guidelines for a roundup with a large number of offenders.

An *interpretation* is a set of atoms that are assumed to be true (atoms not in the interpretation are false). Given an interpretation, we call a rule $r \equiv (A \leftarrow B)$ *applicable* when the precondition B, called the body B_r, is true (i.e. $B \subseteq I$). A rule $r \equiv (A \leftarrow B)$ is said to be *applied* iff it is applicable and the consequence A, called the head H_r, contains exactly one true atom. The latter condition is reasonable as rules with more than one element in their head represent decisions of which only one alternative may be chosen.
Now that we have interpretations, we can distinguish the different alternatives, as we only consider two atoms to be alternatives if a choice between them is forced, i.e. there exists a more specific and applicable choice rule with a head containing (at least) the two atoms. So given an atom a in a component C, we can define the *alternatives* of a in that component C, with respect to an interpretation, as those atoms that appear together with a in the head of an at least as specific applicable choice rule.
Formally, the alternatives for an atom a in $C \in \mathcal{C}$ w.r.t. an interpretation I are defined by $\Omega_C^I(a) = \{b \mid b \neq a \wedge \exists r \in D \in \mathcal{C} \cdot D \preccurlyeq C \wedge B_r \subseteq I \wedge \{a, b\} \subseteq H_r\}$.

Example 4. Reconsider the Police Tactics OCLP of example 3. Assume the following two interpretations: $I = \{crime\_scene\}$ and $J = \emptyset$. The set of alternatives for *wait* in P_1 w.r.t. I equals $\Omega_{P_1}^I(wait) = \{barge\_in, back\_up\}$. In P_4 this would be $\Omega_{P_4}^I(wait) = \emptyset$. This means that *wait* is no longer part of any decision; it should be considered separately. When considering J instead of I we obtain $\Omega_{P_1}^J(wait) = \emptyset$. Interpretation J does not force any decision, so there should not be any alternatives either.

The semantics for choice logic programs is fairly simple: an interpretation is a model iff every rule is either not applicable (i.e. the body is false) or applied (i.e. the body is true and the head contains exactly one head atom). For OCLPs something extra is required to cover the cases in which two or more alternatives of a decision are triggered. In such situations the most specific alternative should be chosen. In the event that some alternatives are equally specific, a random choice between them is justified. This mechanism of choice according to specificity is called "defeating".
Formally, a rule $r \in C \in \mathcal{C}$ is *defeated* w.r.t. an interpretation I iff for each head literal $a \in H_r$, there is an applied *competing* rule $r' \in D \in \mathcal{C}$ where $C \not\preccurlyeq D$ for which $H_{r'} \subseteq \Omega_C^I(a)$ holds.

Example 5. Recall interpretation I for the Police Tactics OCLP in example 4. The rule *back_up* $\leftarrow$ is defeated w.r.t. I because of the applied rule *wait* $\leftarrow$.

The rule $wait \oplus barge\_in \oplus back\_up \leftarrow crime\_scene$ is defeated in P_4 w.r.t. I thanks to the rules $back\_up \leftarrow many$ and $barge\_in \leftarrow many$.

Now we are ready to define the semantics for OCLPs. An interpretation is a *model* if every rule appearing in one of the components is either not applicable, applied or defeated. Stable models are introduced to filter out models that assume too much or which are unintuitive. Just as for standard logic programs, they are based on a Gelfond-Lifschitz transformation. The transformed logic program P^M, where M is an interpretation, is obtained by first removing all the defeated rules. Then, all the false atoms are deleted from the heads of the remaining choice rules. If there are still choice rules left then they are transformed into constraints[5] preventing two head atoms to become true when the rule is applicable. A stable model is then an interpretation which is a minimal model of the transformed program. In [DVV00] a simple algorithm for the computation of stable models of ordered choice logic programs is given.

Example 6. The preferred tactics of the police patrol in case of a drug transfer correspond exactly with the stable models of our Police Tactics OCLP of example 3. Our program has two models: $M_1 = \{crime\_scene, few, barge\_in\}$ and $M_2 = \{crime\_scene, many, barge\_in, back\_up\}$ which are also the only stable models of our program.

4 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, [OR96]), is 4-tuple, denoted $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$, 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 $\preceq_i$ 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

[5] Rules with an empty head are called constraints.

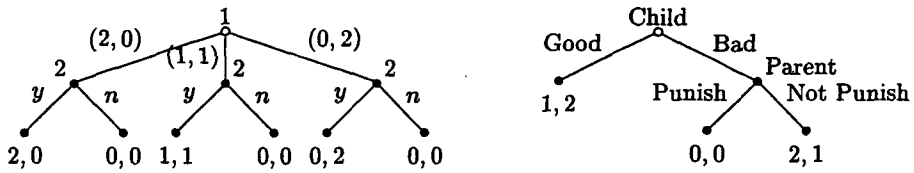

Fig. 2. The Sharing-an-Object game of Example 7 (left) and The Child-Parent game of Example 9 (right).

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 7. Two people use the following procedure to share two desirable identical objects. One of them proposes an allocation, which the other either accepts or rejects. In the event of rejection, neither person receives either of the objects. A game, $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$, that models the individuals' predicament is shown in its alternative representation on the left side of Fig. 2.

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[6].

Example 8. The extensive game with perfect information of example 7 has nine Nash equilibria $((2,0), yyy)$, $((2,0), yyn)$, $((2,0), yny)$, $((2,0), ynn)$, $((1,1), nyy)$, $((1,1), nyn)$, $((0,2), nny)$, $((2,0), nny)$, $((2,0), nnn)$.

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 9. The game on the right side of Fig. 2 has two Nash equilibria: (Good, Punish) and (Bad, Not Punish), with payoff profiles (1,2) and (2,1). The strategy profile (Good, Punish) is an unintuitive Nash equilibrium because given

[6] 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.

that the Parent chooses Punish after history Bad, it is optimal for the Child to choose Good at the start of the game. So the Nash equilibrium is sustained by the "threat" of the Parent to choose Punish if the Child is Bad. However, this threat is not credible since the Parent has no way to commit herself to this choice. Thus the Child can be confident that the Parent will Not Punish him in case he is Bad; since the Child prefers the outcome (Bad, Not Punish) to the Nash equilibrium (Good, Punish), he has thus the incentive to deviate from the equilibrium and choose Bad. We will see that the notion of a subgame perfect equilibrium captures these considerations.

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.

Example 10. The Child-Parent game of example 9 has one subgame perfect equilibrium, (Bad, Not Punish), corresponding to the non-credible threat of the Parent. The Object-sharing game of example 7 has two subgame perfect equilibria: $((2,0), yyy)$ and $((1,1), nyy)$.

When players decide on their actions in an equilibrium, either Nash or subgame perfect, they examine their personal preference relationship on strategies. They try to find the most favorable actions they need to take when considering the information they already know and the assumptions they have about the other players' actions. For each of the actions they choose, there are alternatives which are at the most as preferable as the action they selected. Since the stable model semantics for ordered choice logic programming works along the same lines, it seems reasonable that it should be possible to transform a game into an OCLP such that the Nash or subgame perfect equilibria can be retrieved.

5 Agents and Game Theory

In this section we demonstrate how OCLPs can be used to represent extensive games with perfect information in such a way that the stable models of the program correspond with, depending on the transformation, the Nash equilibria or the subgame perfect equilibria. In the first part we repeat the results of [DVV00] and argue that their transformations do not take the individual players and the structure of the game into account. In the second part, we introduce

two new transformations with the same capabilities as the previous ones but with consideration for the individuality of the players and to some extent the game's structure. In the last part we create separate OCLPs for each player and combine them in a multi-agent system of which the stable models correspond to the Nash or subgame perfect equilibria.

5.1 Phase 1: OCLPs Representing Games

In [DVV00] it was shown that a finite extensive game with perfect information can easily be transformed to an OCLP in such a way that either the Nash equilibria or the subgame perfect equilibria of the game correspond with the stable models of the program.

Let us start with the transformation, called P_n, needed to obtain the Nash equilibria of the game. The set of components consists of a component containing all the decisions that need to be considered and a component for each payoff. The order amongst the components is established according to the corresponding payoff (higher payoffs correspond to more specific components) with the decision component at the bottom of the hierarchy (the most specific component). 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 to reason about both past and future. This is reflected in the rules: each rule in a payoff component is made out of a terminal history (path from top to bottom in the tree) where the head represents the action taken when considering the past and future according to this history. The component of the rule corresponds with the payoff the deciding player would receive in case the history was carried out.

The transformation, called P_s, of extensive games with perfect information needed to obtain the subgame perfect equilibria is quite similar to the one for obtaining the Nash equilibria. The only difference between the two is the creation of rules: since subgame perfect equilibria do take the sequential structure into account, players no longer need to reason about what happened before their decision. They can solely focus on the future.

Example 11. Reconsider the object-sharing game of example 7. The corresponding OCLP P_n is depicted on the left side of Fig. 3[7]. This program P_n has nine stable models which exactly correspond with the nine Nash equilibria of the game. The the second OCLP in Fig. 3 shows P_s. This P_s has the subgame perfect equilibria $(a, y_1y_2y_3)$ and $(b, n_1y_2y_3)$ as its stable models.

Theorem 2. *Let $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ be a finite extensive game with perfect information and let P_n and P_s be its corresponding OCLPs. Then, s^* is a Nash*

[7] To make the graph more readable we renamed the actions $(2,0)$, $(1,1)$ and $(0,2)$ as respectively a, b and c. We also labeled the responses of the second player to make the choices disjoint.

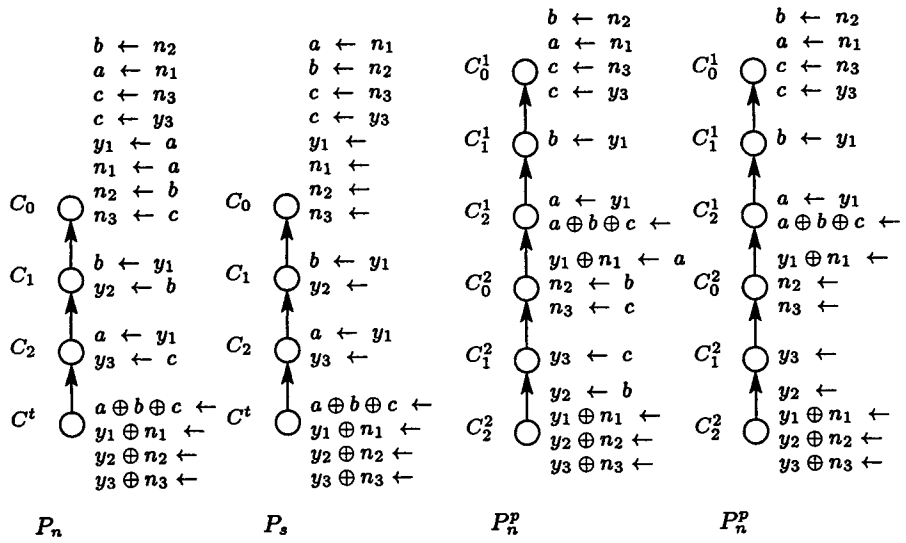

Fig. 3. The corresponding P_n and P_s OCLPs and the phase 2 OCLPs, P_n^p and P_s^p of the extensive game with perfect information of example 7.

equilibrium (resp. subgame perfect equilibrium) for $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ *iff* s^* *is a stable model for* P_n *(resp.* P_s*).*

Looking at Fig. 3 one notices that all information concerning players is totally lost during the transformation process. The structure of the game is broken apart but is still, with lots of effort, retrievable from the programs. In the next two subsections we will focus on bringing the players in the program structure, first in a single OCLP and afterwards in multi-agent system where each agent represents a player of the game.

5.2 Phase 2: Player Based OCLPs

It is fairly simple to adapt P_n and P_s to take into account the player structure. Instead of having a payoff component for every payoff in the game, we introduce payoff components corresponding to the distinct payoffs of each player (e.g. if two players i and j have a payoff 0 then we now have two components C_0^i and C_0^j instead of the single component C_0). Rules made out of a terminal history are now put in the component corresponding to the player taking the associated decision and her perceived payoff. The decision component is no longer necessary as we put the decision rule(s) of a player in the component with the highest payoff corresponding to this player. The order among the components is established, first among components of the same player, according to their payoff (lower payoff is more general) and, secondly, according to the structure of the game (the first deciding player is less specific). The transformation for obtaining the

Nash equilibria is denoted as P_n^p, while P_s^p is used to obtain the subgame perfect equilibria.

The next example illustrates the proposed transformations.

Example 12. Reconsider the object-sharing game of example 7. The transformations P_n^p and P_s^p are depicted on the right side of Fig. 3. The program P_n^p (resp. P_s^p) has precisely the Nash equilibria (resp. subgame perfect equilibria) as its stable models.

Theorem 2 is still valid for the new transformations.

Theorem 3. *Let $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ be a finite extensive game with perfect information and let P_n^p and P_s^p be its corresponding OCLPs. Then, s^* is a Nash equilibrium (resp. subgame perfect equilibrium) for $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ iff s^* is a stable model for P_n^p (resp. P_s^p).*

5.3 Phase 3: Multi-agent Systems

The multi-agent systems that we propose for our games consist of a finite number of agents which are connected in the form of a loop by uni-directional communication channels . Each agent uses an OCLP for the representation of its reasoning skills. A convenient representation for our multi-agent system S is a sequence $A_1 A_2 \ldots A_n$ of OCLPs, where the chain of communication starts at A_1 and ends at A_n to go back to A_1. The models of the system as a whole are those individual models that are accepted (as a model) by every agent in the system. A minimal model (according to set inclusion) is called stable. An elegant fixpoint characterization of stable models can be defined as follows: the first agent P_0 in the chain proposes (one of) her stable model(s) M_0 to her successor P_1. Since M_0 may not be a model for P_1, P_1 will adapt it, within the boundaries of her capabilities, to a new model M_1 for P_1. Then M_1 is forwarded to P_2 etc. The model M_n produced by the last player P_n is then sent back to P_0. The computation stops if a model can flow through the chain without alterations. This method corresponds nicely with the way the players reason in an extensive game. Each player thinks along the lines of "what will the other players/agents after me do in case I would decide this". After her model has passed the whole chain she receives a model reflecting the actions of the other players. According to her strategy she then can reconsider her actions. So the whole sequence of models one obtains before reaching the model which everybody approves of reflects the reasoning process of the entire population of players and the way they respond to each others actions.

The transformations P_n^p and P_s^p can be easily adapted to yield the agents' programs in a multi-agent system: if suffices to "cut" the partial order between components corresponding to different players. S_n and S_s denote the multi-agent versions of resp. P_n and P_s.

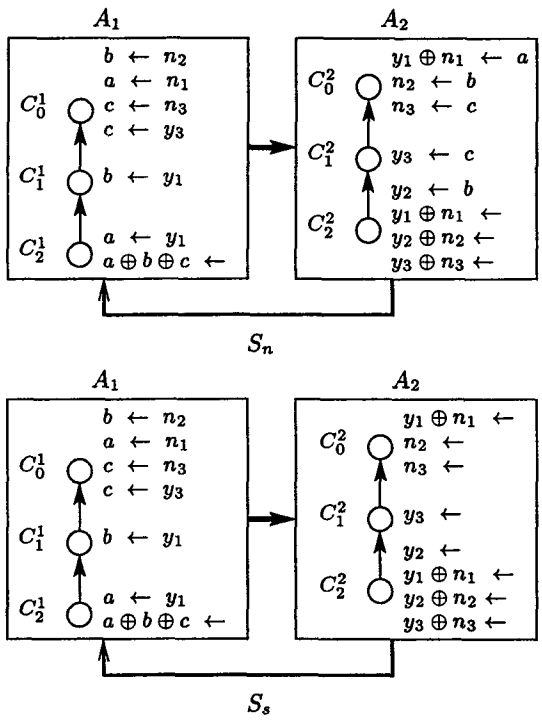

Fig. 4. The multi-agent systems to obtain the Nash equilibria and subgame perfect equilibria of the extensive game with perfect information of example 7.

Note that, while a simple linear structure suffices to recover extensive games with perfect information, our multi-agent systems allow for much more complex communication structures to model e.g. limited awareness of other players etc.

Example 13. Fig. 4 depicts the two multi-agent systems corresponding to the Object-sharing game of example 7 to obtain either the Nash equilibria (S_n on the left side) or the subgame perfect equilibria (S_s on the right).

Theorem 4. *Let $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ be a finite extensive game with perfect information and let S_n and S_s be its multi-agent. Then, s^* is a Nash equilibrium (resp. subgame perfect equilibrium) for $\langle N, H, P, (\succeq_i)_{i \in N} \rangle$ iff s^* is a stable model for S_n (resp. S_s).*

Furthermore we can show that in both systems a fixpoint is obtained no matter what the initial model was and this with no more iterations than the number of agents. This is mainly due to the guaranteed existence of a subgame perfect equilibrium which is also a Nash equilibrium.

Given such a system of agents it fairly easy to reconstruct the original game, since the transformation stores all relevant information either in the rules or in

the structure of the system.
Instead of making the preference of each player known to the whole population of players, the fixpoint characterization allows a player to learn about the preferences of the other participants. In the end, when every possibility is examined, the result will be the same, everybody will know each other preferences. This makes it possible to reuse players in similar games without making any changes. This setting also allows experiments with other forms of information hiding. For example, agents can decide not to pass all their knowledge. Another aspect of this hiding of internal state is the possibility to program agents to deceive other players or to secretly take advantage of the others.

6 Relationship with 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 [DVV00].

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 Police Tactics OCLP (example 3) 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 police patrol). In this case the equilibrium would be $\{barge\_in\}$ 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 [vB]. 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 [Poo97], 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, [RZ94] investigates methods to prevent agents exploiting game theoretic properties of negotiations. [Poo97] 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

[DVV99] 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)*, volume 1730 of *Lecture Notes in Artificial Intelligence*, pages 236–246, El Paso, Texas, USA, 1999. Springer Verslag.

[DVV00] 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*, number 1999 in Lecture Notes in Artificial Intelligence, pages 391–406, Malaga, Spain, 2000. Springer Verslag.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of fifth logic programming symposium*, pages 1070–1080. MIT PRESS, 1988.

[GLV91] 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, Cambridge, Mass, 1991. Morgan Kaufmann.

[Lif00] Vladimir Lifschitz. Answer set programming and plan generation. *Journal of Artificial Intelligence*, page to appear, 2000.

[NBG+01] M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, and M. Barry. A A-Prolog Decision Support System for the Space Shuttle. In *Answer Set Programming: Towards Efficient and Scalable Knowledge Represenation and Reasoning*. American Association for Artificial Intelligence Press, Stanford (Palo Alto), California, US, March 2001.

[OR96] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusets, London, Engeland, third edition, 1996.

[Poo97] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):7–56, 1997.

[RZ94] Jeffrey S. Rosenschein and Gilad Zlotkin. *Rules of Encounter. Designing Conventions for Automated Negotiation among Computers*. The MIT Press, 1994.

[vB] Johan van Benthem. Logic and games. Online course notes of 1999, Stanford University. Available at http://turing.wins.uva.nl/ johan/Phil.298.html.