Logic Programming Agents and Game Theory

Marina De Vos* and **Dirk Vermeir**
Dept. of Computer Science,
Free University of Brussels, VUB,
Pleinlaan 2, Brussels 1050, Belgium,
email:[marinadv,dvermeir]@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 of a 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.

Introduction

Game theory (Osborne & Rubinstein 1996) makes contributions to many different fields. In particular, there is a natural connection with multi-agent systems.
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".

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 a 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.

The continuation of this paper is organized as follows: in Sect. 2, we give a brief overview of OCLPs which model the reasoning capabilities of our individual agents. In Sect. 3 we illustrate that OCLPs can be usefull to simulate a wide class of general logic program, such that the answer set semantics of the former corresponds with our stable model semantics. Section 4 is devoted to extensive games with perfect information and their different equilibria. Our multi-agent systems, capable of producing the equilibria of extensive games with perfect information, are introduced in Sec. 5. We end this paper with a section on relationships to other approaches and directions for future research.

*The author wishes to thank to FWO for its support.

Ordered Choice Logic Programs

In this section we will give a brief informal overview of Ordered Choice Logic Programming (De Vos & Vermeir 2000), or OCLP for short. OCLP 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. We will explain the basics[1] of OCLP by means of Tommy's Birthday Dream.

Example 1 *Today it is Tommy's birthday. Six years old, time goes fast. To celebrate this, his mother agreed to invite some of his friends over for a party. Sitting in his room he is dreaming about his own private party: "A huge birthday cake with lots of candles, of course not forgetting the icing. Lots of candy and biscuits. We just have to make sure that there is plenty, you can never have enough treats. But no matter what, there definitely has to be that big cake. Hopefully my mum will let me decide, that way I can have everything my heart desires. I know that if she starts interfering, she will force me to choose. That is what moms always do." Intuitively, one would expect two possible outcomes for this party:*

- *Tommy's Birthday, Tommy is planning, Tommy and his friends having cake, biscuits and candy.*
- *Tommy's Birthday, Tommy's mother does the planning, Tommy and his friends only having cake.*

OCLP combines Choice Logic Programs (De Vos & Vermeir 1999a; 1999b), for providing the rules to represent the decisions, and Ordered Logic Programs (Gabbay, Laenens, & Vermeir 1991), as the basic concept for the semantics.

[1] We restrict to global interpretation and we give a simplified notion of defeating which is sufficient for the main part of this paper. Full details can be found in (De Vos & Vermeir 2000).

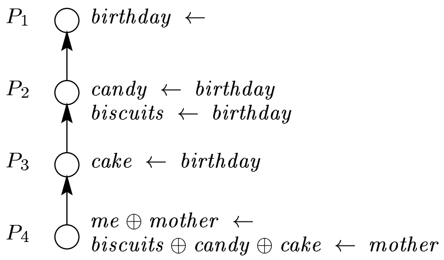

Figure 1: Tommy's Birthday Dream.

Formally, an OCLP P is a pair $\langle \mathcal{C}, \preccurlyeq \rangle$ where C is a collection of CLP's. Thus each component is a finite set of rules of the form $A \leftarrow B$ where A and B are finite sets of atoms. Such 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 $\preccurlyeq$. Another frequently used notion for OCLPs, especially in examples, is the directed acyclic graph (dag) in which the nodes are the components and the arcs represent the relation "$\prec$" (the strict version of $\preccurlyeq$).

Example 2 *Tommy's Birthday dream can easily be translated into the OCLP depicted in Fig. 1, where the choice rules in P_4 correspond with Tommy specifically knowing that either he or his mother will do the organization and that in case his mother will be in charge, he will be forced to choose between all the goodies. The order, together with the rules of P_2 and P_3, expresses that Tommy is more in favor of cake than any of the other treats. Finally P_1 introduces the general fact that it is Tommy's birthday.*

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 finally 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 a more specific applicable choice rule.

Formally, for $C \in \mathcal{C}$, a an atom and I an interpretation, the alternatives for a in C w.r.t. I are defined by

$$\Omega_C^I(a) = \{b \mid \begin{array}{l} b \neq a \wedge \exists r \in D \in \mathcal{C} \cdot D \preccurlyeq C \wedge \\ B_r \subseteq I \wedge \{a, b\} \subseteq H_r \} \end{array}$$

Example 3 *Reconsider Tommy's Dream OCLP of example 2. Let I and J be the following interpretations:*

$$I = \{birthday, me\} \text{ and } J = \{birthday, mother\} \ .$$

The set of alternatives for biscuit in P_2 wrt I equals:

$$\Omega_{P_2}^I(biscuit) = \emptyset \ ,$$

while the one wrt J is:

$$\Omega_{P_2}^I(biscuit) = \{cake, candy\} \ .$$

In words, this means that biscuits is not part of any decision when concerning I, while it is if you are using J instead.

The model 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. We call this mechanism of choice according to specificity 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 $C \not\preccurlyeq D$[2] for which $H_{r'} \subseteq \Omega_C^I(a)$ holds.

Example 4 *Recall the interpretation J given for Tommy's Dream OCLP in example 3. Both rules in P_2 are defeated wrt J, as their head atoms, candy and biscuits, have an alternative, cake, which is the head atom of the more specific applicable rule in P_3.*

Now we are ready to define the model 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 the following procedure:

1. Remove all rules that are defeated w.r.t. M.
2. Remove all atoms that are false in M from the heads of the remaining choice rules (i.e. from those rules that have more than one head atom).
3. Replace each remaining choice rule r by the set of constraints[3]
$$\{\leftarrow B_r, a, b \mid \{a, b\} \subseteq H_r\} \ .$$
4. Consider all remaining rules as a single positive logic program P^M.

A stable model is then an interpretation which is a minimal model of the transformed program.

Example 5 *Tommy's Dream program of example 2 has two models which are also stable, namely:*

$$M_1 = \{birthday, candy, biscuits, cake, me\} \ ., \text{ and}$$
$$M_2 = \{birthday, cake, mother\} \ .$$

[2]This notion of defeat is called *credulous*. An alternative *skeptical* approach is obtained by demanding that a competing rule $r' \in D \in \mathcal{C}$ satisfies $D \prec C$.

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

These stable models correspond exactly to the intuition given in example 1.

In (De Vos & Vermeir 2000) a simple algorithm for the computation of stable models is given.

OCLP's and Answer Sets for Generalized Logic Programs

In (De Vos & Vermeir 1999b) it was shown that choice logic programs, which do not have negation, can simulate a wide class of seminegative logic programs. Specifically, for every semi-negative positive-acyclic[4] datalog program P, there exists a choice logic program P_c such that the stable models of P and P_c coincide.

In this section we generalize this result by showing an equivalence between generalized logic programs, i.e. disjunctive logic programs that allow for negation as failure in the head, and OCLP's. See e.g. (Lifschitz 2000) for a definition of the *answer set semantics* of such programs.

Given a generalized logic program P, we define a OCLP $\Gamma(P) = \langle\{N, R, E\}, \preccurlyeq\rangle$ where $E \prec R \prec N$. The rules in the 3 components are defined as follows (note that in $\Gamma(P)$, we consider $not\ a$ as an atom):

- For each atom a appearing in P, add a " negation-as-default" rule $not\ a \leftarrow$ to N.
- For each rule $A \leftarrow B$ from P, add the set of "shifted" rules
$$\{a \leftarrow B, not\ (A \setminus \{a\}) \mid a \in A\}$$
to R.
- For each atom a appearing in P, add a choice rule $a \oplus not\ a \leftarrow$ to E.

Theorem 1 *Let P be a positive-acyclic generalized logic program. M is an answer set of P iff $M \cup \{not\ a \mid a \notin M\}$ is a stable model of $\Gamma(P)$.*

Extensive Games with Perfect Information

In this section we give a brief and informal overview of extensive games with perfect information (Osborne & Rubinstein 1996).

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 is 4-tuple, denoted $\langle N, H, P, (\geq_i)_{i\in N}\rangle$, 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 number 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, ...).

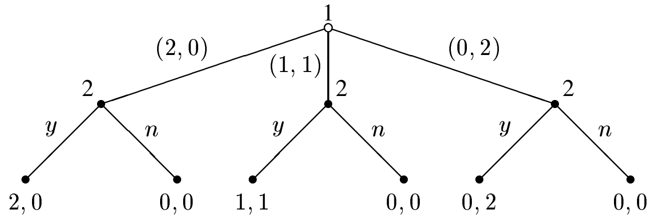

Figure 2: The Sharing-an-Object game of example 6.

Example 6 *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, $\langle N, H, P, (\geq_i)_{i\in N}\rangle$, that models the individuals' predicament is shown in its alternative representation in 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[5].

Example 7 *The extensive game with perfect information of example 6 has nine Nash equilibria [6]:*

[4]A semi-negative program P is called positive-acyclic if there is an assignment of positive integers to all the atoms appearing in P such that the number of the head of any rule is greater than any of the numbers assigned to positive atoms appearing in its body.

[5]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.

[6]A profile is represented as a pair of strategies (s_1, s_2) for player 1 and 2 where s_i is written as the the concatenation of the actions corresponding to the different histories (in the order in which

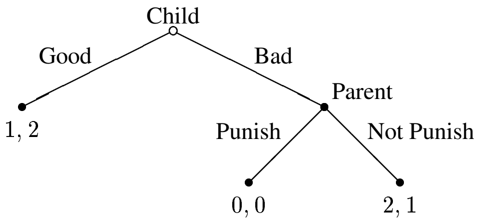

Figure 3: The Child-Parent game of example 8.

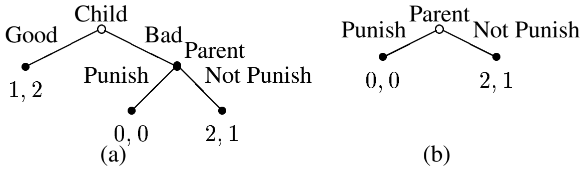

Figure 4: The subgames of the Child-Parent game of example 9.

((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 8 *The game in Fig. 3 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 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.

Example 9 *The two subgames of the game presented in example 8 are depicted in Fig. 4.*

they appear in figure 2). E.g. $((2,0), yyn)$ has player 1 choosing $(2,0)$ while player 2 chooses y for histories $(2,0)$ and $(1,1)$ but n for history $(0,2)$.

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 an other player.

Example 10 *The Child-Parent game of example 8 has one subgame perfect equilibrium, (Bad, Not Punish), corresponding to the non-credible threat of the Parent.*
The Object-sharing game of example 6 has two subgame perfect equilibrium :

$$((2,0), yyy) \textit{ and } ((1,1), nyy) .$$

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 (De Vos & Vermeir 2000) 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.

Phase 1: OCLPs Representing Games

In (De Vos & Vermeir 2000) it was shown that a finite extensive game with perfect information can easily be transformed in a 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 their represented 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

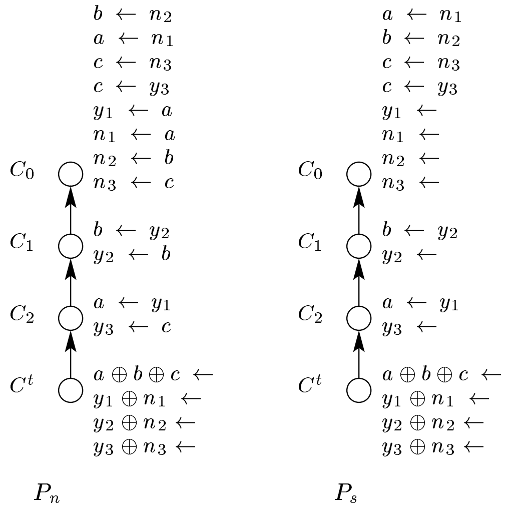

Figure 5: The corresponding P_n and P_s OCLPs of the extensive game with perfect information of example 6.

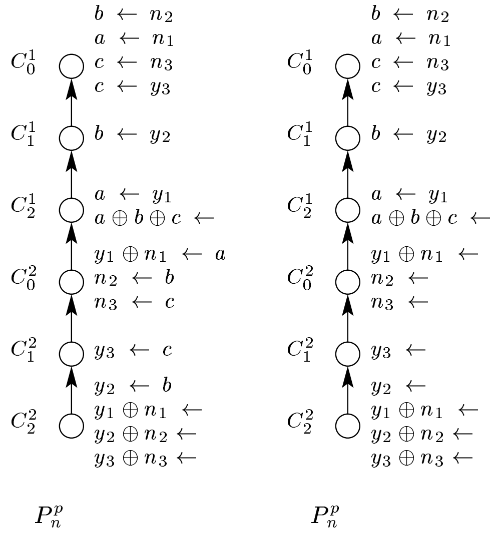

Figure 6: The phase 2 OCLPs of the extensive game with perfect information of example 6.

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 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 6. The corresponding OCLP P_n is depicted on the left side of Fig. 5*[7]*. This program P_n has nine stable models which exactly correspond with the nine Nash equilibria of the game.*
The right side of Fig. 5 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 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. 5 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[8].

Phase 2: Player Based OCLPs

In the previous subsection we argued that the transformations P_n and P_s of (De Vos & Vermeir 2000) totally ignore the players of the game. However, it is fairly simple to adapt them in such a way that they do take into account those players. Instead of having a payoff component for every payoff in the game, we now 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 6. The transformations P_n^p and P_s^p are depicted in Fig. 6. 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 $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ 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 $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ iff s^* is a stable model for P_n^p (resp. P_s^p).*

Phase 3: Multi-agent Systems

The multi-agents systems that we propose for our games consist of a finite number of agents which are connected in

[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.

[8] It is also possible to put more emphasis on the game's structure by assigning an agent to each decision instead of to each player.

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 systems 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 actions 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.

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. 7 depicts the two multi-agent systems corresponding to the Object-sharing game of example 6 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 $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ 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 $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ 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.

Relationship with other approaches

Some research has already been done in the area of agents and games, although with different viewpoints. For example, (Rosenschein & Zlotkin 1994) investigates methods to prevent agents exploiting game theoretic properties of negotiations. The topic of (Birk 1999) is the evolution of cooperation in N-player iterated prisoner's dilemma and how this process can be speeded up. (Poole 1997) 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.

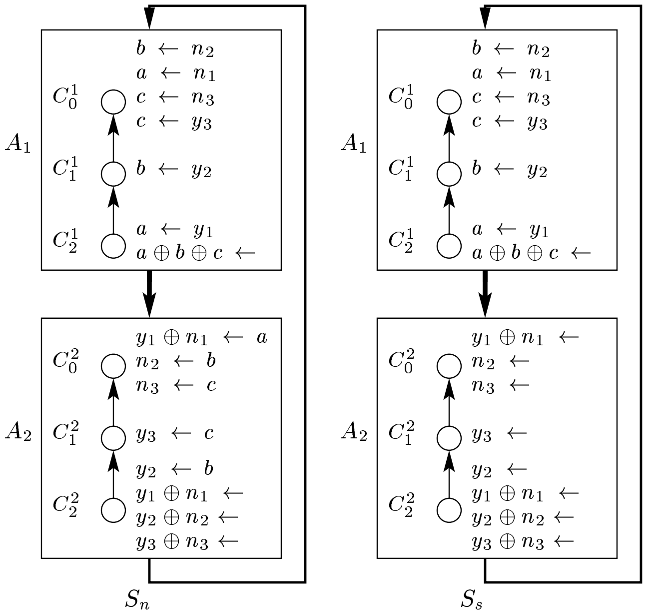

Figure 7: The multi-agent systems to obtain the Nash equilibria and subgame perfect equilibria of the extensive game with perfect information of example 6.

Directions for Future Research

For the future, we plan to extend our multi-agent system to allow more complex ways of communicating. A lot of interesting questions arise once the complexity is raised. What happens in case of multiple incoming channels? Can we model notions as "Common knowledge" and "Distributed knowledge" in such a way that agents can use it advantageously? What happens if an agent is not forced to pass all its knowledge?

Furthermore, we are investigating the possibility of using these multi-agents systems for the representation of repeated games with both finite and infinite horizon.

References

Birk, A. 1999. Trust in an N-Player Iterated Prisoner's Dilemma. In *Agents'99 WC on trust*.

De Vos, M., and Vermeir, D. 1999a. Choice Logic Programs and Nash Equilibria in Strategic Games. In Flum, J., and Rodríguez-Artalejo, M., eds., *Computer Science Logic*

(CSL'99), volume 1683 of *Lecture Notes in Computer Science*, 266–276. Madrid, Spain: Springer Verslag.

De Vos, M., and Vermeir, D. 1999b. On the Role of Negation in Choice Logic Programs. In Gelfond, M.; Leone, N.; and Pfeifer, G., eds., *Logic Programming and Non-Monotonic Reasoning Conference (LPNMR'99)*, volume 1730 of *Lecture Notes in Artificial Intelligence*, 236–246. El Paso, Texas, USA: Springer Verslag.

De Vos, M., and Vermeir, D. 2000. A Logic for Modelling Decision Making with Dynamic Preferences. In *Proceedings of the Logics in Artificial Inteligence (jelia2000) workshop*, volume 1919 of *Lecture Notes in Artificial Intelligence*, 391–406. Springer Verslag.

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

Lifschitz, V. 2000. Answer set programming and plan generation. *Journal of Artificial Intelligence* to appear.

Osborne, M. J., and Rubinstein, A. 1996. *A Course in Game Theory*. Cambridge, Massachusets, London, Engeland: The MIT Press, third edition.

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

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter. Designing Conventions for Automated Negotiation among Computers*. The MIT Press.