

# A Logic for Modeling Decision Making with Dynamic Preferences

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://tinfn2.vub.ac.be>

**Abstract.** We present a framework for decision making with the possibility to express circumstance-dependent preferences among different alternatives for a decision. This new formalism, Ordered Choice Logic Programs (OCLP), builds upon choice logic programs to define a preference/specialization relation on sets of choice rules. We show that our paradigm is an intuitive extension of both ordered logic and choice logic programming such that decisions can comprise more than two alternatives which become only available when a choice is actually forced. The semantics for OCL programs is based on stable models for which we supply a characterization in terms of assumption sets and a fixpoint algorithm. Furthermore we demonstrate that OCLPs allow an elegant translation of finite extensive games with perfect information such that the stable models of the program correspond, depending on the transformation, to either the Nash equilibria or the subgame perfect equilibria of the game.

## 1 Introduction

Preferences among defaults or alternatives play an important role in nonmonotonic reasoning, especially when modeling the complex way people reason in every day live. In case of conflict, humans prefer the default or alternative which provides more reliable, more specific or more important information.

For the last two decades, a lot of research in the nonmonotonic reasoning community has concentrated on bringing preference into the different paradigms: for example logic programming ([6, 9, 12]), extended logic programming ([3]), extended disjunctive logic programming ([1]) and prioritized circumscription ([7]). We will discuss some of these systems in more detail later on in this paper when we compare them to our approach.

These systems have demonstrated their usage in a wide variety of applications like law, object orientation, model based diagnosis or configuration tasks. They are especially suitable for working with exceptions to defaults.

In this paper we present a formalism that enables us to reason about decisions with

---

\* The author wishes to thank the FWO-Vlaanderen for its support.

more than two alternatives where the preference between alternatives depends on the situation. The systems mentioned above do not support such dynamic preferences: they either use the preferences when the model is already being computed, which means that the decisions are already made, or they only support preferences between rules with opposite consequences, leaving out the possibility to have decisions with more than two alternatives. Another problem of the latter type of systems is that the alternatives (i.e. complementary literals) are fixed even before writing the program. We feel that alternatives should emerge only when a choice between them is required. Let us illustrate this with the following example.

*Example 1 (Tommy's Birthday).* 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 mums 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.

Thus, in the first solution *cake, biscuit* and *candy* are not considered alternatives of which only one has to be selected, while in second they are because Tommy's mother forces him to make this difficult choice.

To allow this kind of reasoning, two things need to be added to logic programming. First of all we need a mechanism to represent the possible decisions. As argued in [4, 5], choice logic programs are an intuitive tool to represent conditional decisions, as the semantics make sure that only one alternative is chosen. Thus, choice logic programs will be the fundamentals on which we build our new formalism. Now only a mechanism for denoting preference/order amongst different alternatives is missing. To this end, we will use a generalization to multiple alternatives of the ideas behind Ordered Logic [6]. Our formalism, called Ordered Choice Logic Programs, defines a partial order amongst choice logic programs, called components. Each component inherits, like in object orientation, the rules of the less specific components. Normal model semantics is used until alternatives for the same decision are in conflict. Then, the most specific alternative is decided upon.

These extensions offer a new view point to the above mentioned application domains. For example it is possible to reason about which method overrides the others in a subclassing chain, where with the previous systems one could only detect whether a method was overridden or not. Also applications in AI & law can be envisaged: e.g. lawyer can work out a whole strategy by taking into account the possible actions of the other parties.

We are also able to add a new application domain to this list: Game Theory<sup>1</sup>[8]. We will show that ordered choice logic programs are capable of naturally representing finite extensive games with perfect information such that the stable models of the former correspond with, depending on the transformation, either the Nash equilibria or the subgame perfect equilibria of the latter.

The outline of the rest of the paper is as follows: In Sect. 2 we introduce ordered choice logic programs. The stable model semantics for such programs is presented in Sect. 3. Sect. 4 is used for discussing an application in game theory while Sect. 5 compares ordered choice logic programs with some alternative approaches.

## 2 Ordered Choice Logic Programs

The basis of Ordered Choice Logic Programs are, as the name already might have indicated, choice logic programs[4, 5].

We identify these choice logic program with their grounded version, i.e. the set of all ground instances of its clauses. This keeps the program finite as we do not allow function symbols (i.e. we stick to datalog).

**Definition 1 ([4, 5]).** A **Choice Logic Program**, *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 (note that  $A$  may be empty, i.e. constraints are allowed). In examples, we often use " $\oplus$ " to denote exclusive or, while " $,$ " is used to denote conjunction.

The Herbrand Base and interpretations for a choice logic programs are defined in the usual way, except that we will only consider total interpretations in this paper.

**Definition 2 ([4, 5]).** Let  $P$  be a CLP. The **Herbrand Base** of  $P$ , denoted  $\mathcal{B}_P$ , is defined as the set of all atoms appearing in the program. An **interpretation**  $I$  is any subset of the Herbrand Base of  $P$ , i.e.  $I \subseteq \mathcal{B}_P$ . An atom in  $I$  is assumed to be true while an atom in  $\mathcal{B}_P \setminus I$  is considered false. We denote the set of all false atoms wrt  $I$  as  $\bar{I}$ .

**Definition 3.** An **Ordered Choice Logic Program**, or *OCLP*, is a pair  $\langle C, \preceq \rangle$  where  $C$  is a finite set of choice logic programs, called **components**, and " $\preceq$ " is a partial order on  $C$ . In this paper we assume that  $C$  contains a minimal element  $C_\perp$  such that  $C_\perp \preceq X$  for all  $X \in C$ . Furthermore, we assume that a rule appears in at most one component of  $C^2$ .

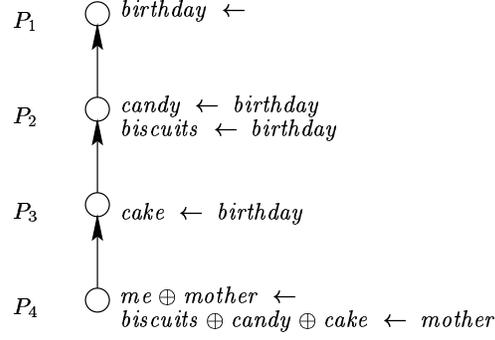
For two components  $C_1, C_2 \in C$ ,  $C_1 \prec C_2$  implies that  $C_2$  contains more general information than  $C_1^3$ . Also  $[A, B]$  is used to denote the set  $\{X \mid A \preceq X \preceq B\}$ . Similarly,  $[A, B[$  denotes the set  $\{X \mid A \preceq X \prec B\}$ .

Throughout the examples, we will often represent an OCLP  $P$  by means of a directed acyclic graph (dag) in which the nodes represent the components and the arcs the relation " $\prec$ ".

<sup>1</sup> Game Theory has proven its usefulness in domains such as economics and computer science.

<sup>2</sup> This is only a technical restriction that considerably simplifies the notation.

<sup>3</sup> As usual, " $\prec$ " denotes the restriction of " $\preceq$ " to all the pairs of distinct components.



**Fig. 1.** Tommy's Birthday Dream.

*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.

As more specific components "inherit" the rules from more general components, we also need, when defining an interpretation, to consider the atoms mentioned in those less specific parts.

**Definition 4.** Given an OCLP  $P$  and a component  $A \in C$  of  $P$ . An **interpretation for  $P$  in  $A$**  is any interpretation of  $A^*$ , where  $A^*$  denotes the CLP  $\{r \in B \in C \text{ and } A \preceq B\}$ . An interpretation for  $P$  is called **global** if it is an interpretation in  $C_\perp$ . We say that a rule  $r$  is **applicable** in  $I$  if  $B_r \subseteq I$ <sup>4</sup> and that  $r$  is **applied** in  $I$  if  $r$  is applicable and  $|H_r \cap I| = 1$ <sup>5</sup>.

We argued in the introduction that choice rules represent a choice between the head elements once the precondition, the body, is satisfied (e.g. the rule is applicable). From that moment on, we can consider those elements as alternatives. With this we can define the alternatives for an atom  $a$  from a viewpoint  $B$  known in a specific component  $A$ , called horizon, as those atoms that appear together with  $a$  in the head of an applicable choice rule in a component  $C$  at least as specific as  $B$  but not more so than  $A$  (e.g.  $C \in [A, B]$ ).

**Definition 5.** Let  $P$  be an OCLP, let  $A, B \in C$  be components of  $P$  and let  $I$  be an interpretation in  $A$ . For any rule  $r \in A^*$ , we use  $c(r)$  to denote its component. The set of all **alternatives for an atom  $a \in B_{A^*}$  in  $[A, B]$ , wrt  $I$** , denoted  $\Omega_{[A, B]}^I(a)$ , is defined as:

$$\Omega_{[A, B]}^I(a) = \{b \mid \exists r \in A^* \cdot c(r) \in [A, B] \wedge B_r \subseteq I \wedge a, b \in H_r \text{ with } a \neq b\} .$$

<sup>4</sup> For a rule  $r \equiv Q \leftarrow R$ , we use  $H_r$  to denote its **head**  $Q$  while  $B_r$  denotes its **body**  $R$ .

<sup>5</sup>  $|A|$  denotes the number of elements in the set  $A$ .

Now we are in a position to demonstrate that OCLPs are really dynamic when considering the alternatives for a decision.

*Example 3.* Reconsider Tommy's Dream OCLP of example 2. Let  $I$  and  $J$  be the following global interpretations:  $I = \{\textit{birthday}, \textit{me}\}$  and  $J = \{\textit{birthday}, \textit{mother}\}$ . The set of alternatives for *biscuit* in  $[C_{\perp}, P_2]$  wrt  $I$  equals:  $\Omega_{[C_{\perp}, P_2]}^I(\textit{biscuit}) = \emptyset$ , while the one wrt  $J$  is  $\Omega_{[C_{\perp}, P_2]}^J(\textit{biscuit}) = \{\textit{cake}, \textit{candy}\}$ . In words, this means that *biscuits* is not part of any decision when considering  $I$ , while it is if you are using  $J$  instead.

Deciding upon different alternatives can vary depending on the one who is making the decision or on the kind of decision. In all cases, when one alternative is preferred over all others, the choice is easily made: you simply take that alternative and leave out the others. But what happens if some alternatives are equally preferred (or incomparable)? One possible way of dealing with this dilemma is just making an objective choice between those alternatives. In this case, one is at least sure that there is a solution to the problem. This is the credulous<sup>6</sup> way of looking at the world.

In this context we say, intuitively, that a rule is defeated if there exist(s) some applied rule(s) containing head alternatives that are not less preferred than the ones defeated in the head of the defeated rule.

**Definition 6.** Let  $P$  be an OCLP, let  $A \in C$  be a component of  $P$  and let  $I$  be an interpretation in  $A$ . A rule  $r \in A^*$  is **defeated in  $A$  wrt  $I$**  iff

$$\forall a \in H_r \cdot \exists r' \in A^* \cdot c(r) \not\prec c(r') \wedge r' \text{ is applied} \wedge H_{r'} \subseteq \Omega_{[A, c(r)]}^I(a).$$

The rules  $r'$  are called **defeaters**.

The following two examples illustrate the two possible ways that a rule can be defeated: a rule can either be defeated by a single rule containing only alternatives for each head element, or by a number of rules containing only alternatives for some of the head elements, but together they offer alternatives for the whole lot.

*Example 4.* Consider the following OCLP  $\langle C, \preceq \rangle$  with:

$$P_1 : r_1 : a \leftarrow \quad P_2 : r_2 : a \oplus b \leftarrow \quad P_3 : r_3 : b \leftarrow$$

such that  $C = \{P_1, P_2, P_3\}$  and  $P_3 \prec P_2 \prec P_1$ . Let  $I = \{b\}$  be an interpretation in  $P_3$ . For this interpretation, the rule  $r_1$  is defeated by the more specific rule  $r_3$  as  $a$  has a more specific alternative  $b$ , due to the more specific rule  $r_3$ .

*Example 5.* Consider the following OCLP  $\langle C, \preceq \rangle$  with:

$$P_1 : r_1 : a \oplus b \leftarrow \quad P_2 : r_2 : a \leftarrow \quad P_3 : r_3 : b \leftarrow$$

---

<sup>6</sup> There exists also a more skeptical way of facing alternatives that are equally preferred or incomparable. Whereas in the credulous approach a choice between the alternatives is acceptable, one remains undecided in the skeptical one. Although most results in this paper also hold for the skeptical semantics, we will only use the credulous approach in this paper.



2. remove all false atoms from the head of the remaining rules with more than one atom in the head,
3. replace all rules  $r$  with more than one head atom with constraint rules: for each such rule  $r$  where  $a, b \in H_r$  and  $a \neq b$ , we add a constraint

$$\leftarrow B_r, a, b .$$

The introduction of constraints is necessary to assure that a non-defeated applicable choice rule with more than one head atom will be properly satisfied (i.e. only one head atom must be considered true).

Stable models for a program are then minimal models of the program obtained from applying the Gelfond-Lifschitz transformation.

**Definition 9.** Let  $M$  be an interpretation for an OCLP  $P$  in a component  $A$ .  $M$  is called a **stable model** for  $P$  in  $A$  iff  $M$  is a minimal model for the positive logic program  $P_A^M$ .

In example 6, both  $M_1$  and  $M_2$  are stable.

The next theorem confirms our earlier claim that the stable model semantics restricts the minimal model semantics.

**Theorem 1.** Let  $M$  be a stable model for an OCLP  $P$  in a component  $A$ . Then,  $M$  is minimal model for  $P$  in  $A$ .

The reverse is not true, as illustrated by the following example.

*Example 8.* Consider the program  $P$  from example 7 which has a unique minimal model  $M = \{a, b\}$  in  $P_2$ . Applying the Gelfond-Lifschitz transformation on  $P$  in  $P_2$  yields

$$P_{P_2}^M : \begin{array}{l} a \leftarrow b \\ b \leftarrow a \end{array}$$

This program has as a minimal model  $\emptyset \neq M$ , so  $M$  is not stable.

Looking back on example 7, we note that, for the minimal model  $M = \{a, b\}$ , at least one atom must have been produced only by a defeated rule. Intuitively, such atoms can be considered assumptions, because they lack a proper motivating rule to introduce them. The following definition makes this intuition more precise.

**Definition 10.** Let  $I$  be an interpretation for an OCLP  $P$  in a component  $A$ . A set  $X \subseteq \mathcal{B}_{A^*}$  is called an **assumption set** wrt  $I$  iff for each  $a \in X$  one of the following conditions is satisfied:

1.  $\exists r \equiv (a \oplus A \leftarrow B) \in A^* \cdot B \subseteq I \wedge A \cap I \neq \emptyset \wedge r$  is not defeated in  $A$  wrt  $I$ ; or
2.  $\exists r \equiv (\leftarrow B, a) \cdot B \subseteq I$ ; or
3.  $\forall r \in A^*$  where  $a \in H_r$ , one of the following conditions holds:
  - (a)  $B_r \not\subseteq I$ ; or
  - (b)  $B_r \cap X \neq \emptyset$ ; or
  - (c)  $r$  is defeated in  $A$  wrt  $I$ ; or
  - (d)  $H_r \cap B_r \neq \emptyset$ .

The set of all assumption sets for  $P$  in  $A$  wrt  $I$  is denoted  $\mathcal{A}_{P|A}(I)$ . The **greatest assumption set** for  $P$  in  $A$  wrt  $I$ , denoted  $\mathcal{GAS}_{P|A}(I)$ , is the union of all assumption sets for  $P$  in  $A$  wrt  $I$ .

The first condition in Definition 10 expresses that, if there exists a non-defeated applicable rule with already a true atom in the head, then the interpretation does not need  $a$  to become/maintain a model. The second condition says that, if a constraint contains, besides the element one is considering, only true atoms, one should not assume that element to be true as well. The last condition states that if every rule with  $a$  in the head is either not applicable, defeated, containing assumptions in the body or sharing atoms both in the head and the body, then we know that the atom  $a$  is not involved in making the interpretation into a model.

The greatest assumption set is an assumption set.

**Proposition 2.** *Let  $I$  be an interpretation for an OCLP  $P$  in a component  $A$ . Then,  $\mathcal{GAS}_{P|A}(I) \in \mathcal{A}_{P|A}(I)$ .*

Assumption sets can be used to eliminate candidate models.

**Proposition 3.** *Let  $M$  be a model for an OCLP  $P$  in a component  $A$ . Then  $\overline{M}$  is an assumption set, i.e.  $\overline{M} \in \mathcal{A}_{P|A}(M)$ .*

Checking the assumption-free property can be quite time consuming when one needs to verify every subset of  $\mathcal{B}_{A^*}$ . The following proposition implies that there is an easier way.

**Proposition 4.** *Let  $I$  be an interpretation for an OCLP  $P$  in a component  $A$ .  $I$  is assumption-free, i.e.  $I \cap \mathcal{GAS}_{P|A}(I) = \emptyset$ , iff no non-empty subset of  $I$  is an assumption set for  $P$  in  $A$  wrt  $I$ .*

Assumption sets characterize stable models.

**Theorem 2.** *Let  $M$  be a model for an OCLP  $P$  in a component  $A$ . Then,  $M$  is stable iff  $M$  is assumption-free for  $P$  in  $A$  wrt  $M$ , i.e.  $M \cap \mathcal{GAS}_{P|A}(M) = \emptyset$ .*

For choice logic programs we have that minimal models are unfounded-free, which equals assumption-free when the interpretation is total. For OCLP, this can no longer be maintained. A counter example was presented in example 7: the minimal model  $\{a, b\}$  is not assumption-free (i.e.,  $\{a, b\} \in \mathcal{A}_P(\{a, b\})$ ).

Assumption sets are also useful to compute stable models: Fig. 2 contains a sketch of a backtracking fixpoint procedure  $BF$  such that  $BF(\emptyset)$  generates all stable models (in the component  $A$ ).

## 4 An Application to Finite Extensive Games with Perfect Information

In this section we give a brief and informal overview of extensive games with perfect information ([8]) and demonstrate in more detail how OCLP's can be used to retrieve the games' equilibria from the transformed programs.

```

procedure  $BF(I : \text{set}\langle \text{atom} \rangle)$  {
set $\langle \text{atom} \rangle X = \mathcal{GAS}_{P|A}(I)$ 
if  $(X \cap I) \neq \emptyset$ 
    fail
if  $(X = \bar{I})$ 
     $I$  is a stable model
else {
    set $\langle \text{rule} \rangle R = \{r \mid r \in A^* \text{ applicable and not defeated and } H_r \cap I = \emptyset\}$ 
    set $\langle \text{atom} \rangle J = \{a \mid a \in H_r \wedge r \in R \wedge a \notin X\}$ 
    for each  $a \in J$ 
         $BF(I \cup \{a\})$ 
    }
}

```

**Fig. 2.** Computing stable models

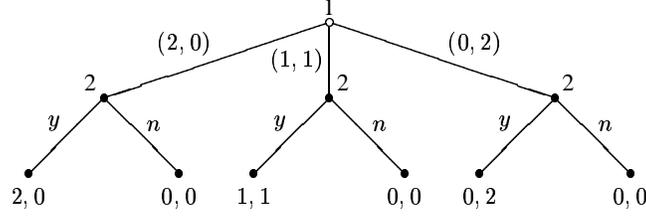
An extensive game is a detailed description of a sequential structure representing the decision problems encountered by agents (called *players*) in strategic decision making (agents are capable to reason about their actions in a rational manner). The agents in the game are perfectly informed of all 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, a game 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  $\geq_i$  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, ...).

*Example 9.* 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.

An extensive game with perfect information,  $\langle N, H, P, (\geq_i)_{i \in N} \rangle$ , that models the individuals' predicament is shown in its alternative representation in Fig. 3.

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 every player. E.g.  $((2, 0), yyy)$  is a strategy profile where the first player intends to take both objects and the second player plans to accept (indicated by "y") any of the three possible proposals from the first player.



**Fig. 3.** The Sharing-an-Object game of example 9.

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 choices. Put in another way, given the other players' strategies, the strategy stated for the player is the best this player can do<sup>7</sup>.

*Example 10.* The extensive game with perfect information of example 9 has nine Nash equilibria:  $((2, 0), yyy)$ ,  $((2, 0), yyn)$ ,  $((2, 0), yny)$ ,  $((2, 0), ynn)$ ,  $((1, 1), nyy)$ ,  $((1, 1), nyn)$ ,  $((0, 2), nny)$ ,  $((2, 0), nny)$ ,  $((2, 0), nnn)$ .

The following transformation will be used to retrieve the Nash equilibria from the game as the stable models of the corresponding OCLP.

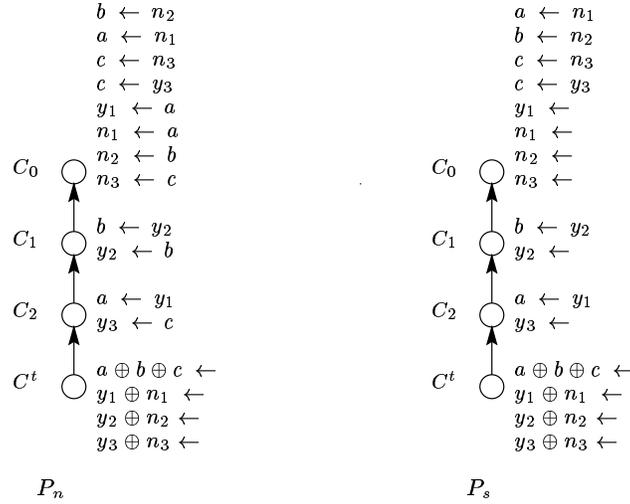
**Definition 11.** Let  $\langle N, H, P, (\geq_i)_{i \in N} \rangle$  be an extensive game with perfect information. The corresponding OCLP  $P_n$  can be constructed in the following way:

- $C = \{C^t\} \cup \{C_u \mid \exists i \in N, h \in Z \cdot u = U_i(h)\};$
- $C^t \prec C_u$  for all  $C_u \in C$ ;
- $\forall C_u, C_w \in C \cdot C_u \prec C_w$  iff  $u > w$ ;
- $\forall h \in (H \setminus Z) \cdot (\{a \mid ha \in H\} \leftarrow) \in C^t$ ;
- $\forall h = h_1ah_2 \in Z \cdot a \leftarrow B \in C_u$  with  $B = \{b \in [h]^8 \mid h = h_3bh_4, P(h_3) \neq i\}$  and  $u = U_{P(h_1)}(h)$ .

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

<sup>7</sup> 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.

<sup>8</sup> We use  $[h]$  to denote the set of actions appearing in a sequence  $h$ .



**Fig. 4.** The corresponding  $P_n$  and  $P_s$  OCLPs of the extensive game with perfect information of example 9.

past and future created by the other players 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.

*Example 11.* Reconsider the Object-sharing game of example 9. The corresponding OCLP  $P_n$  is depicted on the left side of Fig. 4<sup>9</sup>. This program  $P_n$  has nine stable models which exactly correspond with the nine Nash equilibria of the game.

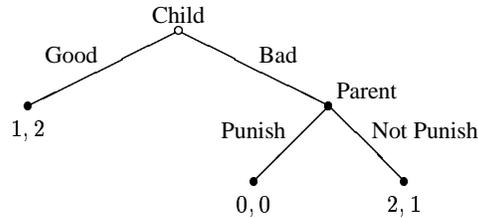
In the next theorem we show that there is indeed a correspondence between Nash equilibria and stable models.

**Theorem 3.** *Let  $G = \langle N, H, P, (\geq_i)_{i \in N} \rangle$  be a finite extensive game with perfect information and let  $P_n$  be its corresponding OCLP. Then,  $s^*$  is a Nash equilibrium for  $G$  iff  $s^*$  is a global stable model for  $P_n$ .*

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 12.* The game in Fig. 5 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

<sup>9</sup> 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.

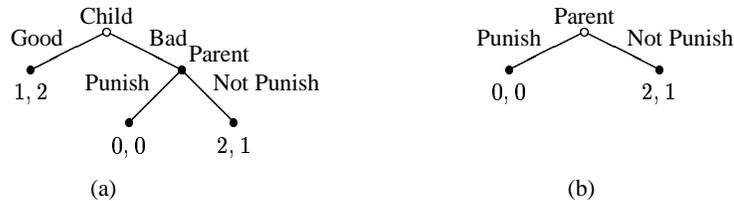


**Fig. 5.** The Child-Parent game of example 12.

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 13.* The two subgames of the game presented in example 12 are depicted in Fig. 6.



**Fig. 6.** The subgames of the Child-Parent game of example 13.

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 14.* The Child-Parent game of example 12 has one subgame perfect equilibrium, (Bad, Not Punish), corresponding to the non-credible threat of the Parent.

The Object-sharing game of example 9 has two subgame perfect equilibrium :  $((2, 0), yyy)$  and  $((1, 1), nyy)$ .

The following transformation makes sure that subgame perfect equilibria correspond with the stable models of an OCLP.

**Definition 12.** Let  $\langle N, H, P, (\geq_i)_{i \in N} \rangle$  be an extensive game with perfect information. The corresponding OCLP  $P_s$  can be constructed as follows:

- $C = \{C^t\} \cup \{C_u \mid \exists i \in N, h \in Z \cdot u = U_i(h)\};$
- $C^t \prec C_u$  for all  $C_u \in C;$
- $\forall C_u, C_w \in C \cdot C_u \prec C_w$  iff  $u > w;$
- $\forall h \in (H \setminus Z) \cdot (\{a \mid ha \in H\} \leftarrow) \in C^t;$
- $\forall h = h_1 a h_2 \in Z : P(h_1) = i \cdot (a \leftarrow B) \in C_u$  with  $B = \{b \in [h_2] \mid h = h_3 b h_4, P(h_3) \neq i\}$  and  $u = U_{P(h_1)}(h)$ .

This transformation is quite similar to the one for obtaining the Nash equilibria. The only difference between the two is the creation of history-dependent 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 15.* Consider once more the object-sharing game of example 9. The corresponding OCLP  $P_s$  is show on the right side of Fig. 4. This  $P_s$  has the subgame perfect equilibria  $(a, y_1 y_2 y_3)$  and  $(b, n_1 y_2 y_3)$  as its stable models.

**Theorem 4.** Let  $G = \langle N, H, P, (\geq_i)_{i \in N} \rangle$  be a extensive game with perfect information and let  $P_s$  be its corresponding OCLP. Then,  $s^*$  is a subgame perfect equilibrium of  $G$  iff  $s^*$  is a global stable model for  $P$ .

Note that [10] proposes an alternative formalism to model strategic games using an extension of logic programming. However, in [10], the specification of choices is external to the program while, in our approach, we rely on nondeterminism (and priority) to represent alternatives and on the properties of the stable model semantics to obtain equilibria.

## 5 Relationships to Other Approaches

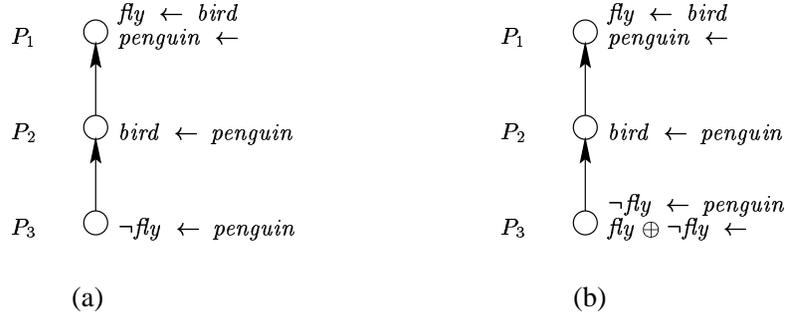
### 5.1 Ordered Logic ([6])

Ordered logic programs are a special, also semantically, case of OCLP's: all choices are restricted to 2 alternatives  $a$  and  $\neg a$ . This is confirmed by the following.

**Proposition 5.** Let  $P = \langle C, \preceq \rangle$  be an ordered logic program in the sense of [6] and let  $A \in C$  be a component for it. The corresponding OCLP  $P_A$  with respect to  $A$  equals  $\langle C', \preceq \rangle$  where:

$$C' = \{B \in C \mid B \neq A\} \cup \{A \cup \{a \oplus \neg a \leftarrow \mid a, \neg a \in \mathcal{B}_{A^*}\}\} .$$

An interpretation  $I$  in  $A$  is a model for  $P$  in  $A$  iff  $I$  is a model for  $P_A$  in  $A$ .



**Fig. 7.** a) The Ordered logic version of the Penguin problem. b) The corresponding Penguin OCLP  $P_{P_3}$  wrt component  $P_3$ .

We illustrate this construction with the following well-known example:

*Example 16 (Tweety, the penguin).* The left side of Fig. 7 depicts the ordered logic program for the problem. The right hand side gives the corresponding OCLP wrt to component  $P_3$ . Both programs have only one model in component  $P_3$ , namely  $M = \{bird, \neg fly, penguin\}$ .

## 5.2 Other Approaches to Preference

Dynamic preference in extended logic programs is introduced in [3] in order to obtain a better suited well-founded semantics. Although preferences are called dynamic they are not dynamic in our sense. Instead of defining a preference relation on subsets of rules, preferences are incorporated as rules in the program. Moreover, a stability criterion may come into play to overrule preference information. Another difference with our approach is that the alternatives are static.

A totally different approach is proposed in [12]. Here the preferences are defined amongst atoms. Given these preferences, one can combine them to obtain preferences for sets of atoms. Defining models in the usual way, the preferences are then used to filter out the less preferred models. That way, this system is not convenient for decision making as the preferences cannot easily be made to depend on the situation.

In [1], preference in extensive disjunctive logic programming is considered. As far as overriding is concerned the technique corresponds rather well with our skeptical defeating, but alternatives are fixed as an atom and its (real) negation.

Outside the context of logic programming, [2] proposes to add priorities to the object language of default logic. Extensions are then required to be compatible with this information. OCLP and [2] support different intuitions on the notion of priority, as shown by the following example<sup>10</sup>:

*Example 17.*

$$\begin{array}{ll}
 P_1 : a \leftarrow & P_2 : \neg c \leftarrow \\
 P_3 : c \leftarrow a & P_4 : c \oplus \neg c \leftarrow
 \end{array}$$

<sup>10</sup> For the ease of notation we simply denote the program in our formalism.

with  $P_4 \prec P_3 \prec P_2 \prec P_1$ . With our approach, we obtain  $\{a, c\}$  as the (stable) model of this program while [2] returns  $\{a, \neg c\}$  as the extension for the default theory. [2] considers the knowledge of  $a$  coming from a more general rule insufficient (the rule from  $P_1$ ) to favor the rule from  $P_4$  over the one from  $P_3$ . We, and also [11], prefer to say that there is no counter evidence for  $a$  so we should exploit this knowledge as much as possible.

## References

1. Francesco Buccafurri, Wolfgang Faber, and Nicola Leone. Disjunctive Logic Programs with Inheritance. In Danny De Schreye, editor, *International Conference on Logic Programming (ICLP)*, pages 79–93, Las Cruces, New Mexico, USA, 1999. The MIT Press.
2. Gerhard Brewka. Reasoning about priorities in default logic. In Barbara Hayes-Roth and Richard Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 940–945, Menlo Park, California, 1994. American Association for Artificial Intelligence, AAAI Press.
3. Gerhard Brewka. Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, 4 (1996) 19–36.
4. Marina De Vos and Dirk Vermeir. Choice Logic Programs and Nash Equilibria in Strategic Games. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 266–276, Madrid, Spain, 1999. Springer Verslag.
5. 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.
6. 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.
7. Vladimir Lifschitz. Computing Circumscription. In *9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, 1985.
8. Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, London, England, third edition, 1996.
9. David Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, 1985.
10. David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1–2):7–56, 1997.
11. Henry Prakken and Giovanni Sartor. A system for defeasible argumentation, with defeasible priorities. In Dov M. Gabbay and Hans Jürgen Ohlbach, editors, *Proceedings of the International Conference on Formal and Applied Practical Reasoning (FAPR-96)*, volume 1085 of *LNAI*, pages 510–524, Berlin, June 3–7 1996. Springer.
12. Chiaki Sakama and Katsumi Inoue. Representing Priorities in Logic Programs. In Michael Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 82–96, Cambridge, September 2–6 1996. MIT Press.