Credulous vs. Sceptical Semantics for Ordered Logic Programs

D. Gabbay
Dept. of Computing
Imperial College
180 Queen's Gate
London SW72BZ, UK

E. Laenens
Dept. of Math. and Computer Science
University of Antwerp, UIA
Universiteitsplein 1
B2610 Wilrijk, Belgium

D. Vermeir
Dept. of Math. and Computer Science
University of Antwerp, UIA
Universiteitsplein 1
B2610 Wilrijk, Belgium

Abstract

We present a semantic approach to the characterization of credulous and sceptical reasoning mechanisms, within the framework of ordered logic. One of the advantages of our approach is that it integrates thightly with "conservative" ordered logic semantics which is known to generalize "classical" (stable and well-founded) logic programming semantics. This allows us to compare the conservative and credulous approaches, thus providing insight in the fundamental properties of both reasoning paradigms. It turns out that maximal credulous models are extensions of conservative stable models while the (unique) minimal credulous model, called the sceptical model, is a restriction of the well-founded model.

1. MOTIVATION

Consider a logic program $C_1=\{\neg q \rightarrow p\}$. The computation of this program assumes that, since q is not a head of any clause, $\neg q$ is part of the data. Suppose we relinquish this principle and adopt the principle of asking an *advisor* what to do with $\neg q$. The adviser might say that $\neg q$ succeeds or might say that $\neg q$ fails. The adviser might have his own program to consult. If his program is C_2, he might run the goal q (or $\neg q$), look at what he gets and then advise. To make the situation symmetrical and general we must allow for Horn progams to have rules with both q and $\neg q$ (i.e. literals) in heads and bodies and have any number of negotiating advisers. Thus we can have $C_2=\{\neg q\}$, $C_{1=}\{\neg q \rightarrow p\}$ and C_1 depends on C_2. Ordered logic FIXOL EXLOP geerts nute 1989 computational intelligence develops and studies various aspects of such an advisor system which is modeled as a partially ordered set of theories. Such a logic is useful, e.g. for multi-expert systems where we want to represent the knowledge of several experts in a single system. Experts may then be ordered according to an "advisory" or a relative preference relation.

A problem to consider is what happens when we have several advisers that are in conflict. For example, C_1 depends on C_2 and C_1 depends on C_3. The two advisers, C_2 and C_3, may be in conflict. One may advise $\neg q$, the other q. How to decide? There are several options: in a *conservative* approach, one would not make an overall conclusion re. q, while a *credulous* approach could yield two models: one with q, the other with $\neg q$. In this work, we present a semantic approach to the characterization of credulous and sceptical reasoning mechanisms, within the framework of ordered logic . FIXOL One of the advantages of our approach is that it integrates thightly with "conservative" ordered logic semantics which is known EXLOP to generalize "classical" (stable and well-founded) logic programming semantics. This allows us to compare the conservative and credulous approaches, thus providing insight in the fundamental properties of both reasoning paradigms. It turns out that our formalization of credulous reasoning extends the well-founded - stable hierarchy of (partial) models in that stable models can be extended to credulous models while the (unique) well-founded model is an extension of the (unique) minimal credulous model, which is called the sceptical model. The present work also sheds light on the underlying mechanisms that may cause a theory to have multiple "best" models: in the conservative approach, multiplicity (of stable models) is caused by making "choices" which are not based on "assumptions", while the credulous approach allows further extensions of a (stable) model by making (conservative) assumptions.

We start off by presenting a summary of the relevant definitions and results on ordered logic.

The proofs of the results presented in this paper can be found in Chapter 8 of . laenens phd

2. ORDERED LOGIC PROGRAMS

In this section, we introduce the basic concepts and notations of ordered logic. For a more detainled description with examples, we refer to EXLOP

The basic tokens of the language are terms, formulas and literals where a *term* is defined as a variable or a constant . Lloyd foundations 1987 An *atomic formula* or *atom* of the language is of the form $p(t)$, where p is a predicate symbol with arity n ($n \geq 0$) and t is a sequence of n terms (*arguments of the atom*). A *literal* is either an atom (*positive literal*) or its negation (*negative literal*). A term, predicate or literal is *ground* if it is variable-free.

Two literals are complementary if they are of the form A and $\neg A$, for some atom A. In general, given a literal A and a set of ground literals X, $\neg A$ denotes the complement of A and $\neg X$ denotes the set of literals $\{\neg B \mid B \in X\}$. Moreover, $pos(X)$ (resp. $neg(X)$) denotes the set of all positive (resp. negative) literals in X.

A *negative rule* (or simply, a *rule*) is a formula of the language represented as follows:

$$Q_1, \ldots, Q_m \rightarrow Q_0$$

where $Q_0, \ldots, Q_m$ are literals, Q_0 is the *head* of the rule, and $Q_1, \ldots, Q_m$ is the body of the rule. If Q_0 is positive then the rule is a *seminegative rule*; moreover, if also $Q_1, \ldots, Q_m$ are all positive then the rule is a *positive rule* (or Horn clause). Given a rule r, $H(r)$ denotes the head of r and $B(r)$ denotes the set of all literals in the body of r. A rule is *ground* if it is variable-free. A ground rule is a *fact* if it has an empty body.

A *negative program* is a set of rules. If all rules are seminegative (resp. positive) then the program is called a *seminegative program* (resp. *positive program*).

Let P be a negative program. The Herbrand Universe of P (denoted by H_P) is the set of all possible ground terms. The Herbrand Base of P (denoted by B_P) is the set of all possible ground atoms whose predicate symbols occur in P and whose arguments are elements of H_P. A *ground instance* of a rule r in P is a rule obtained from r by replacing every variable X in r by $\phi(X)$, where ϕ is a mapping from the set of all variables occurring in P to H_P. The set of all ground instances of all rules in P is denoted by $ground(P)$. Any subset of $B_P \cup \neg B_P$ is called an *interpretation* for P. We say that an interpretation X is *consistent* if there are no two literals A and B in X such that $A = \neg B$. Finally, we denote by $\hat{X}$ the set $\{\overline{p} \in B_P \cup \neg B_p \mid \text{neither } p \text{ nor } \neg p \text{ is in } X\}$ and by X the set $(B_P \cup \neg B_P) - X$.

We now formally define the notion of ordered logic program.

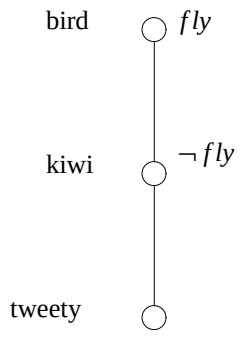
Definition 1.

An **ordered logic program** is a finite partially-ordered set of negative programs, called **components**, where " $\leq$ " is the partial order. □

The "<" relation is a sort of isa hierarchy for the components and provides the ground for inheritance.

(As usual, " $<$ " denotes the restriction of " $\leq$ " to all pairs of distinct components.)

Throughout all examples of this work, an ordered logic program P is represented as a pair $<C, L>$ where C is the set of components and L is the relation "<", or as a directed acyclic graph (dag) in which the nodes represent the components and the arcs the relation "<". As an example consider the ordered logic version P_1 of the well known "tweety" example: P_1 has components $bird = \{fly\}$, $kiwi = \{\neg fly\}$ and $tweety = \emptyset$ ordered by $tweety < kiwi < bird$.



P_1

It turns out that an ordered logic program has several meanings, one for each of its components. Moreover, the knowledge defined at a component, i.e. the local knowledge, does not constitute the entire knowledge about that component as a component inherits information from other components. Therefore, the meaning of a particular component in an ordered logic program is defined by all rules that are visible to this component, i.e. all local rules (defined in this component) as well as all global rules defined in components that are higher up in the component-hierarchy induced by the partial order.

Definition 2.

Given an ordered logic program P and a component C of P. An **interpretation** for P in C is any interpretation of C^*, where C^* denotes the negative program $\{r \mid r \in C' \text{ and } C \leq C'\}$. □

In the next section, we briefly discuss the conservative semantics for ordered logic.

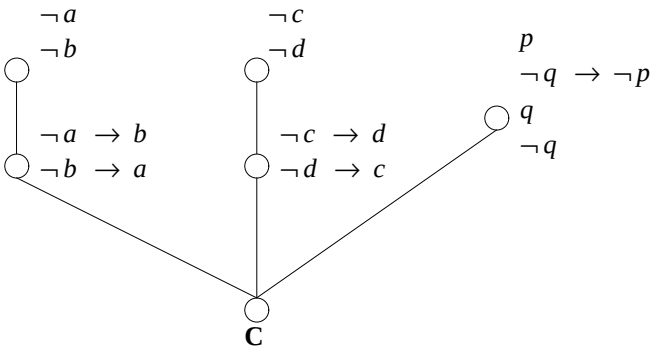
3. PARTIAL MODEL SEMANTICS

For a detailed discussion of the well-founded and stable partial model semantics, we refer to . laenens phd As was explained in the previous section, an ordered logic program has several meanings, one for each component. In a given component, its semantics

depends on the rules that are visible to it, i.e. all local rules (defined in this component) as well as all global rules defined in components that are higher up in the component-hierarchy induced by the partial order. Hence, an interpretation reflects the meaning of a component if it satisfies all rules that are visible to this component. In other words, for a consistent interpretation M to be a model for a component, it suffices that it satisfies all local and global rules of this component. Classically, this would amount to demanding that each visible rule r in $ground(C^*)$ is either non-applicable in M, i.e. $B(r) \not\subseteq M$, or applied in M, i.e. $B(r) \subseteq M$ and $H(r) \in M$. In the case of an ordered logic program, this is not enough laenens vermeir assumption-free report relationship laenens vermeir fixpoint semantics journal computation as it cannot deal with a situation where several competing rules (i.e. rules with complementary heads p and $\neg p$) are applicable. So we add a third possibility, saying that a rule is also satisfied if it is **defeated** by a competitor, i.e. it has an applicable **competitor** where a competitor is a visible rule with complementary head that is defined in a component that is not strictly higher in the component hierarchy. Let $c(r)$ denote the component in which the rule r is defined. For a given ordered logic program P and a component of it C, the set of competitors of a rule r in $ground(C^*)$ is

$$comp_{P\,|\,C}(r) = \{\hat{r} \in ground(C^*) \mid H(\hat{r}) = \neg H(r) \text{ and } c(\hat{r}) \not> c(r)\}$$

r is defeated in M by a rule $\hat{r}$ in $comp_{P\,|\,C}(r)$ if $\hat{r}$ is applicable in M. Such a competitor will be called a **defeater** of r. Summarizing, an interpretation M is a **model** for an ordered logic program in a component of it if each rule that is visible from this component is either non-applicable or applied or defeated. As an example, the interpretation $\{\neg fly.\}$ is a model of P_1 since the rule *fly.* at *bird* is defeated by the applied rule $\neg fly$ at *kiwi*.As another example, consider P_2.



P_2

Both $\{\neg a, b, \neg c, d, p\}$ and $\{a, \neg b, c, \neg d, p\}$ are models of P_2, as can easily be verified.

Those interpretations that are not models can be thought of as interpretations that are 'too poor', in the sense that they are not deductively closed. Some ordered logic programs such as P_3 have no models:

P_3

Programs like P_3 are called "contradictory" in . FIX-REP

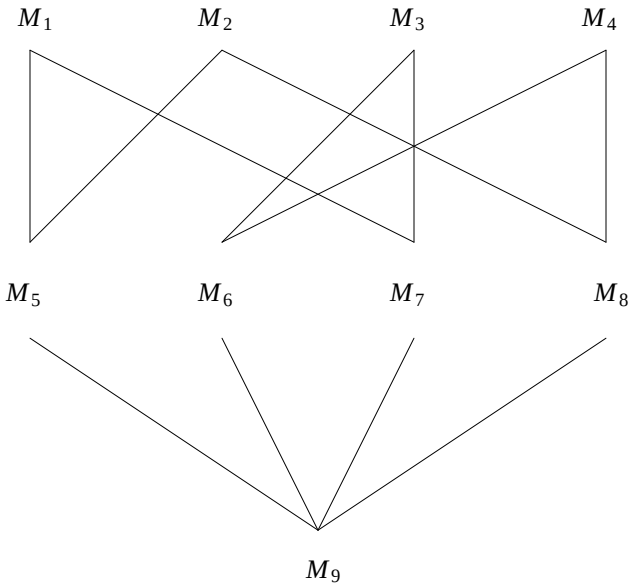
In SOLPARTIAL the weaker notion of "partial model" was introduced in order to be able to give a meaningful interpretation to any ordered logic program. Intuitively, the notion of partial model is obtained by relaxing the definition of defeater: instead of demanding that a rule's competitor be applicable, we will require that its body not be "false", i.e. it is either true (in which case the competitor is applicable) or undefined . Fitting Ben-Jacob 1988 Przymusinski 1989 three-valued non-monotonic you yuan needed Reconsidering the example program P_3, the empty set is a partial model as the literal $\neg p$ is undefined and therefore the (non-applicable) rule $\neg p \rightarrow p$ acts as a defeater of $\neg p$.

In order to formalize this intuition, we need to characterize what it means for a literal to be "false" w.r.t. a given interpretation. We cannot adopt the traditional logic programming definition saying that a ground literal is false w.r.t. a given interpretation if its negation is true w.r.t. this interpretation, since this traditional definition of falsity makes only sense in the context of some closed world assumption where the negation of each literal that is considered false w.r.t. the interpretation is added to this interpretation. In other words, a literal is not false because its negation is in the interpretation, but the negation of a literal is in the interpretation because the literal is false. An excellent illustration of this is the fixpoint computation of the well-founded model in gelder ross principles where each iteration adds besides the positive literals as inferred using the immediate consequence transformation, also the negation of each literal in the unfounded set which is exactly the set of ground literals that are already known to be false. Since ordered logic makes no use of a closed world assumption, it follows that we need to know the ordered-logic equivalent of the unfounded set in order to obtain a correct notion of falsity. To this end, we generalize the concept of **unfounded set** as defined in unfounded to ordered logic programs. As in the classical case, the **greatest unfounded set** w.r.t. an interpretation is the set of literals that are **false** w.r.t. this interpretation. Intuitively, a rule whose body is false, i.e. it contains

unfounded literals, is out of order and should not be accepted as a defeater of some other rules; only rules of which the body is either false or undefined can act as defeaters. We say that a rule r in $ground(C^*)$ is **defeasible** if it has a competitor $\hat{r}$ in $comp_{P|C}(r)$ whose body is not false, i.e. the intersection of $B(\hat{r})$ with the greatest unfounded set is empty.

Hence the following definition of partial model, which weakens the definition of model. An interpretation M is a **partial model** for an ordered logic program in a component of it if each rule $C \rightarrow p$ that is visible from this component is either non-applicable or applied or defeasible. Intuitively, a partial model M guarantees that there are no 'pending' rules, i.e. rules that definitely should, and are not applied in M. In other words, those interpretations that are not partial models can be thought of as interpretations that are 'too poor', in the sense that they are not deductively closed. On the other hand however, partial models can still be 'too rich' in that they may contain literals that cannot be actually inferred. We call these literals, that are not justified by an applicable non-defeated and non-defeasible rule, assumptions. An **assumption set** of a partial model M is a subset X of this partial model such that for each literal p of X, each rule r in $ground(C^*)$ with $H(r)=p$ is either non-applicable in M or defeated in M or defeasible in M or $B(r)$ contains assumptions ($B(r) \cap X \neq \emptyset$). The last possibility avoids circularities where a literal l_1 motivates another literal l_2 and vice versa. It is easily verified that assumption sets are closed under union, so we can talk about the **greatest assumption set** of a partial model M. It is evident that our interest goes to **assumption-free partial models,** i.e. partial models that have an empty greatest assumption set, as they are fully inferrable as well as deductively closed, thus representing the exact meaning of the ordered logic program. It can be shown that each ordered logic program has a partial model. In particular, the least fixpoint $W_{P|C}^{\infty}(\emptyset)$ of the (monotone) *ordered immediate consequence transformation,* $W_{P|C}$, defined as $W_{P|C}(I)=\{p \mid$there exists a rule $C \rightarrow p$ which is applicable but not defeasible in $I\}$, is an assumption-free partial model.

The set of all assumption-free partial models of an ordered logic program P form a partial order under the subset relation. The maximal elements of this partial order are called *stable partial models* while the minimal elements are called *well-founded partial models.* As an example, the partial-model hierarchy corresponding to the ordered logic program P_2 is depicted in below. Note that there are four stable partial models, M_1 .. M_4, and one well-founded partial model $M_9=\{p\}$, where $M_5=\{a, \neg b, p\}$, $M_6=\{\neg a, b, p\}$, $M_7=\{c, \neg d, p\}$, $M_8=\{\neg c, d, p\}$, $M_1=M_5 \cup M_7$, $M_2=M_5 \cup M_8$, $M_3=M_6 \cup M_7$, and $M_4=M_6 \cup M_8$. In SOLPARTIAL it was shown that there is but a unique well-founded partial model, which is equal to $W_{P|C}^{\infty}(\emptyset)$. Another interesting result is that the intersection of all stable partial models is exactly the well-founded partial model in all cases but a very special type of (contradictory) ordered logic programs. Together with the proof theory (procedural semantics) for ordered logic developed in , geerts tubingen this gives a soundness and completeness result for ordered logic. For the relationship between partial models and "full" models, we refer to . SOLPARTIAL



4. CREDULOUS AND SCEPTICAL MODELS

It was discussed before that the need for defeating arises where the classical approach fails, namely when it comes to the satisfaction of applicable competing rules. From this, it follows that an actual defeater, i.e. a rule that causes a rule to be defeated (not just defeasible), should definitely be applicable. The question then arises as to whether a defeater should be applied or not? If in an ordered logic program, every two competing rules are such that one is strictly preferred to the other, then the defeater of an applicable, non-applied rule is automatically applied. So, the answer only depends on whether we want to ignore equivalent (i.e. incomparable in the partial order) conflicting information or whether we want to make a choice in the case of equivalent conflicting information. It turns out that there are two alternative approaches, one taking the view that a defeater is an applicable competitor thus unifying conflicting information with no information, and the other taking the view that a defeater should be an applied competitor thus enforcing a choice. The former *conservative* approach was discussed in the previous section. Our objective now is to develop semantics for the second *credulous* approach and to discuss its relationship with the conservative approach.

Only a few changes to the definitions of the previous section are required: specifically, the new notion of defeating will be reflected in the credulous versions of (the definitions of) unfounded and assumption sets. Surprisingly, it will turn out that the new credulous semantics not only provides us with a more credulous interpretation of ordered logic programs, but also with an extremely sceptical interpretation.

Definition 3.

Let P be an ordered logic program and C a component of it. Given an interpretation I for P in C, a rule r in $ground(C^*)$ is **c-defeated** in I if there exists a rule $\hat{r}$ in $comp_{P\,|\,C}(r)$ such that $B(\hat{r}) \cup H(\hat{r}) \subseteq I$. □

Definition 4.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C. A subset X of $B_P \cup \neg B_P$ is a **c-unfounded set** of P in C w.r.t. I if for each p in X, every rule r in $ground(C^*)$ with $H(r)=p$ is either c-defeated in I or $B(r) \cap X \neq \emptyset$.

The **greatest c-unfounded set** of P in C w.r.t. I, denoted $Upcc(I)$, is the union of all sets that are c-unfounded sets of P in C w.r.t. I. $U^c_{P\,|\,C}(I)$ is easily seen to be a c-unfounded set. □

Lemma 1.

Let P be an ordered logic program, C a component of it, and I and J interpretations for P in C. If $I \subseteq J$, then $U^c_{P\,|\,C}(I) \subseteq U^c_{P\,|\,C}(J)$. ■

Clearly, $U^c_{P\,|\,C}(I)$ is a subset of $U_{P\,|\,C}(I)$ which fits our intuition that, in the current case, more literals should be true, i.e. more literals should be in I, due to the enforced choice when conflicts arise. From this observation, it follows immediately that, in general, we can expect more c-partial models than partial models as the number of defeasible rules increases (i.e. there are more c-defeasible rules than defeasible rules).

Definition 5.

Let P be an ordered logic program and C a component of it. Given an interpretation I for P in C, a rule r in $ground(C^*)$ is **c-defeasible** in I if there exists a rule $\hat{r}$ in $comp_{P\,|\,C}(r)$ such that $(B(\hat{r}) \cup H(\hat{r})) \cap U^c_{P\,|\,C}(I) = \emptyset$. □

Definition 6.

Let P be an ordered logic program and C a component of it. A consistent interpretation M is a **c-partial model** for P in C if every rule r in $ground(C^*)$ is either applied in M or non-applicable in M or c-defeasible in M. □

The next definition is simpler than the conservative definition of assumption set. This is due to the fact that an applied rule can never be c-defeated w.r.t. an interpretation.

Definition 7.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C. A subset X of I is a **c-assumption set** of P in C w.r.t. I if for each p in X, every rule r in $ground(C^*)$ with $H(r) = p$ satisfies one of the following conditions:

(a) r is non-applicable in I, or

(b) r is c-defeasible in I, or

(c) $B(r) \cap X \neq \emptyset$.

The **greatest c-assumption set** of P in C w.r.t. I, denoted $A^c_{P\,|\,C}(I)$, is the union of all sets that are c-assumption sets of P in C w.r.t. I. $A^c_{P\,|\,C}(I)$ is easily seen to be a c-assumption set. A member of $A^c_{P\,|\,C}(I)$ is called an **c-assumption.** I is said to be **c-assumption free** iff $A^c_{P\,|\,C}(I) = \emptyset$. □

Interpretations that are free from c-assumptions are also free from c-unfounded literals.

Lemma 2.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C. Then $U^c_{P\,|\,C}(I) \cap I \subseteq A^c_{P\,|\,C}(I)$. ■

As for the conservative approach, we can show that the greatest c-assumption set is that part of the interpretation which is non-inferrable. To this end, we define the credulous enabled version which corresponds to the inferrable part of a c-partial model.

Definition 8.

Let P be an ordered logic program, C a component of P and M a c-partial model for P in C. The **credulous enabled version** of $ground(C^*)$ w.r.t. M, denoted by C^c_M, is the program containing all rules of $ground(C^*)$ that are applied in M and not c-defeasible in M. □

Let us now apply the *immediate consequence transformation T* as defined for seminegative and positive programs to C^c_M. Given any interpretation I for P in C, $T_{C^c_M}(I) = \{A \mid$ there exists a rule r in C^c_M such that $H(r)=A$ and $B(r) \subseteq I\}$.

Lemma 3.

Let P be an ordered logic program, C a component of P and M a c-partial model for P in C. Then $T_{C^c_M}$

is monotone and has a least fixpoint, denoted $T^{\infty}_{C^c_M}(\emptyset)$. *Moreover,* $T^{\infty}_{C^c_M}(\emptyset) \subseteq M$. ■

The members of $M - T^{\infty}_{C^c_M}(\emptyset)$ cannot be inferred in C^c_M: they are assumptions of M.

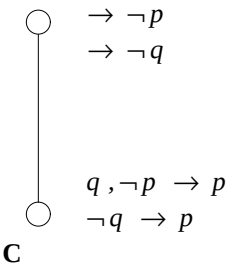
Theorem 1.

Let P be an ordered logic program and C a component of it. A c-partial model M for P in C is c-assumption free if and only if $T^{\infty}_{C^c_M}(\emptyset) = M$. ■

While the condition of c-defeating would be meaningless in the definition of c-assumption set, we next illustrate that the condition of c-defeasible is essential.

Example 1.

Consider the program P_4 depicted below, together with the interpretation $M = \{\neg p\}$.



P_4

Clearly, $U^c_{P_4|C}(M) = \emptyset$, and therefore, the credulous enabled version is also empty as each rule in P_4 is either c-defeasible in M or non-applicable in M. As expected (Theorem 1), we find that M is not assumption free because the rule $\rightarrow \neg p$ is c-defeasible in M. So, condition (b) in the definition of c-assumption set is essential. □

We next prove that a c-assumption-free c-partial model exists for any ordered logic program in any of its components, by means of the following transformation.

Definition 9.

Let P be an ordered logic program and C one of its components. Let $\boldsymbol{I}$ be the family of all interpretations of P in C. The **credulous ordered immediate consequence transformation** for P in C is the function $S_{P|C} : \boldsymbol{I} \rightarrow \boldsymbol{I}$ defined as follows: given an interpretation I, $S_{P|C}(I) = \{p \mid$ there exists a rule r in $ground(C^*)$ such that $H(r) = p$, r is applicable in I and not c-defeasible in $I\}$. □

Lemma 4.

For a given ordered logic program P and a component of it C, the transformation $S_{P|C}$ is monotone and has a least fixpoint. ■

We denote the least fixpoint of $S_{P|C}$ by $S^{\infty}_{P|C}(\emptyset)$.

Lemma 5.

Given an ordered logic program P and a component C of P. $S^{\infty}_{P|C}(\emptyset)$ is a c-assumption-free c-partial model for P in C. ■

We now consider the collection of all c-assumption-free c-partial models of an ordered logic program P in a component C, which is partially ordered by the set-inclusion relation.

Definition 10.

Let P be an ordered logic program, C a component of it and $<\mu, \subseteq>$ the partially ordered set where μ denotes the set of c-assumption-free c-partial models for P in C. A c-partial model M for P in C is called

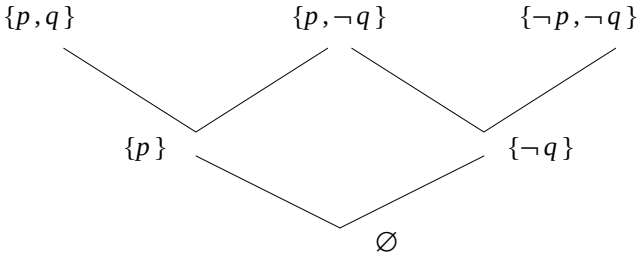
- **credulous** if M is maximal in $<\mu, \subseteq>$,
- **sceptical** if M is minimal in $<\mu, \subseteq>$. □

Example 2.

Consider the ordered logic program P_5 which has but one component C.

$$C = \{ \rightarrow p \quad \neg q \rightarrow \neg p \quad \rightarrow q \quad \rightarrow \neg q \}$$

The partial order of c-assumption-free c-partial models for P_5 in C is depicted in the next figure. There are 3 credulous c-partial models and there is a single sceptical one.



□

While the previous example illustrates that an ordered logic program may have several alternative credulous c-partial models in a component, we can show that the

uniqueness of the sceptical c-partial model is guaranteed.

Theorem 2.

Let P be an ordered logic program and C a component of it. Then $S_P^{\infty}{}_{|C}(\emptyset)$ is the unique sceptical c-partial model. ■

5. SCEPTICAL PROOF THEORY

Our next goal is to develop a definition of "proof for properties of objects": for an ordered logic program P and a component C of it, we want a proof procedure that decides whether a given ground literal is true in C of P. In a monotonic environment, producing evidence for the truth of all literals in the body of a rule suffices to conclude the truth of the literal in the head of this rule. Obviously, our proof mechanism will be more complicated as it is essential to take into account the behavior of the competitors of this rule. Therefore, the proof procedure will have to show that any competitor of this rule is in fact "blocked", i.e. at least one literal of its body cannot become true. This is done by inferring not only positive conclusions like "p holds" but also negative ones like "demonstrably, p does not hold".

Proofs of properties of objects are conveniently represented as trees. The intuition is as follows. Let P be an ordered logic program and C a component of it. The nodes of a sceptical prooftree in C of P are labeled by adorned literals, the adornment being either "+" or "-". If a node n in a sceptical prooftree T has a positive adornment, e.g. p^+, then the subtree of T with root n is again a sceptical prooftree which represents a proof for "p is true in C of P". The proof is based on the idea that p must be true in C of P only if there is an applicable, not c-defeasible rule whose head is p, which means that (1) its body can be proven to be true and that (2) the body or the head of each competitor can be proven not to be true. If a node n in a sceptical prooftree has a negative adornment, e.g. p^- then the subtree with root n is again a sceptical prooftree representing a proof for "p is not true in C of P". The proof is based on the idea that p cannot be true in C of P only if each rule whose head is p is either non-applicable or c-defeated which means that (1) its body can be proven not to be true or that (2) there is a competitor of which both the body and the head can be proven to be true. In particular, if the root of the overall sceptical prooftree has a positively (resp. negatively) adorned label p^+ then it represents a proof for "p is true in C of P" (resp. "p is not true in C of P") in terms of a number of sceptical (sub)prooftrees.

Definition 11.

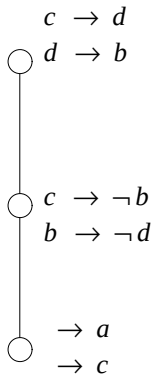
Let P be an ordered logic program and C a component of it. A **sceptical prooftree** in C of P is a finite tree where nodes are labeled by members of $\{p^+ \mid p \in B_P \cup \neg B_P\}$ and $\{p^- \mid p \in B_P \cup \neg B_P\}$. Moreover, each node n must satisfy one of the following conditions depending on its label:

- If n has label p^+ then there must be some rule r in $ground(C^*)$ with $H(r)=p$ such that n has a child labeled c^+ for each $c \in B(r)$. Moreover, for each competitor $\hat{r}$ in $comp_{P|C}(r)$, n must have a child labeled d^- for some $d \in B(\hat{r}) \cup H(\hat{r})$.
- If n has label p^- then either
  - (a) for every rule r in $ground(C^*)$ with $H(r)=p$, n must have a child labeled c^- for some $c \in B(r)$, or r must have a competitor $\hat{r}$ in $comp_{P|C}(r)$ such that n has a child labeled d^+ for each $d \in B(\hat{r}) \cup H(\hat{r})$; or
  - (b) n must have an ancestor n' with the same label (p^-) such that no node between n and n' has a + label.

A sceptical prooftree is said to be a sceptical prooftree of p^s in C of P if its root is labeled p^s. □

Example 3.

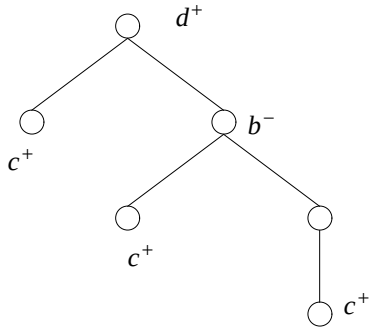
The figure below illustrates an ordered logic program and a prooftree of d^+ in the bottom component C of this program.



□

Definition 12.

Let P be an ordered logic program, C a component of it and p^r (where r is either + or –) an adorned ground literal. The existence of a sceptical prooftree of p^r in C of P is denoted by $P_{C\ c}\ p^r$. Moreover, P_C^{c+} denotes $\{p \mid P_{C\ c}\ p^+\}$, P_C^{c-} denotes $\{p \mid P_{C\ c}\ p^-\}$ and $P_C^{c?}=P_C^{c+} \cup P_C^{c-}$. P_C^{c+} is called the **credulous extension** of P in C. P_C^{c-} is called the **credulous exclusion** of P in C. □



So, the credulous extension P_C^{c+} is the set of ground literals that, following the credulous approach, can be proven to be true in C of P and the credulous exclusion P_C^{c-} is the set of ground literals that can be proven not to be true in C of P while $P_C^{c?}$ contains those ground literals for which no proof exists.

As shown in , laenens phd P_C^{c+} and P_C^{c-} are well-defined: $P_C^{c+} \cap P_C^{c-} = \emptyset$ and P_C^{c+} is a consistent interpretation. ■

It turns out that the credulous extension P_C^{c+} is the sceptical c-partial model and that the credulous exclusion P_C^{c-} is the unfounded set w.r.t. the sceptical c-partial model.

Theorem 3.

Let P be an ordered logic program and C a component of it. Then $P_C^{c+} = S_P^{\infty}{}_{|C}(\emptyset)$, and $P_C^{c-} = U_P^c{}_{|C}(S_P^{\infty}{}_{|C}(\emptyset))$. ■

Note that the above theorem gives us a procedural top-down characterization of the sceptical c-partial model.

Another interesting result is that the sceptical proof theory is sound and complete w.r.t. the c-assumption-free c-partial model semantics.

Theorem 4. (Soundness)

Let P be an ordered logic program, C a component of it and M a c-assumption-free c-partial model for P in C. Then $P_C^{c+} \subseteq M$ and $M \cap P_C^{c-} = \emptyset$. ■

Theorem 5. (Soundness and Completeness)

Let P be an ordered logic program, C a component of it. The intersection of all c-assumption-free c-partial models is exactly the credulous extension. ■

A natural question to ask then is what is the relationship between the conservative analogues of credulous and sceptical c-partial models. This is the topic of the next section.

6. CONSERVATIVE VS. CREDULOUS APPROACH

6.1. PARTIAL VS. C-PARTIAL MODELS

In this section, we present the relationship between partial models and c-partial models.

Lemma 6.

Let P be an ordered logic program, C a component of it and I an interpretation for P in C. Then

(a) *the greatest c-unfounded set of P in C w.r.t. I is a subset of the greatest unfounded set of P in C w.r.t. I, i.e. $U_P^c{}_{|C}(I) \subseteq U_P{}_{|C}(I)$.*

(b) *the greatest c-assumption set of P in C w.r.t. I is a subset of the greatest assumption set of P in C w.r.t. I, i.e. $A_P^c{}_{|C}(I) \subseteq A_P{}_{|C}(I)$.* ■

Theorem 6.

Let P be an ordered logic program and C a component of it. Every partial model for P in C is a c-partial model for P in C. Moreover, every assumption-free partial model for P in C is a c-assumption-free c-partial model for P in C. ■

The next counterexample illustrates that the reverse is not necessarily true.

Example 4.

Reconsider P_5. The empty set is a c-partial model for P_5 in C which is not a partial model for P_5 in C. □

6.2. STABLE VS. CREDULOUS MODELS

As for the relationship between stable partial models and credulous c-partial models, we have the result that each stable partial model can be extended to a credulous c-partial model.

Theorem 7.

Let P be an ordered logic program and C a component of it. Every stable partial model for P in C is a subset of some credulous c-partial model for P in C. ■

The reverse may be false as is illustrated next.

Example 5.

Reconsider P_5. In Example 2, $\{p\}$ is the (only) stable partial model for P_5 in C, which can be extended to two credulous c-partial models for P_5 in C: $\{p, \neg q\}$ and $\{p, q\}$. The reverse does no hold, i.e. there can be credulous c-partial models that are not extensions

of stable partial models. This is illustrated in P_5 by the credulous c-partial model $\{\neg p , \neg q\}$ model for P_5 in C, which is not an extension of any stable partial model for P_5 in C. □

6.3. WELL-FOUNDED VS. SCEPTICAL MODELS

As far as the relationship between well-founded and sceptical models is concerned, we find the following result.

Theorem 8.

Let P be an ordered logic program and C a component of it. The sceptical c-partial model for P in C is a subset of the well-founded partial model for P in C. ■

Example 6.

For P_5, the sceptical c-partial model in C is the empty set, which is a subset of the well-founded partial model $\{p\}$. □

By definition, the sceptical c-partial model is a subset of the intersection of all credulous c-partial models. As is shown in the next example, this feature is not always true for the well-founded partial model.

Example 7.

For P_5, the intersection of all credulous c-partial models in C is the empty set whereas the well-founded partial model in C is $\{p\}$. □

For a discussion on the computation of c-partial models, we refer to , laenens phd where it is shown how to effectively compute c-partial models, credulous c-partial models and the sceptical c-partial model.

7. CONCLUSION

The combination of the conservative and credulous approach to the semantics of ordered logic programs yields a hierarchy of partial models: at the bottom end we have the unique (extremely) sceptical model, which is included in the unique well-founded (partial) model. The well-founded partial model can be extended to yield (possibly) multiple stable partial models, which result from different (conservative) assumption-free *choices* (see FIXOL). The credulous approach allows us to extend these stable partial models further by making (conservative) assumptions.

It should be noted that the new concepts for ordered logic programs are proper generalizations of the corresponding concepts for classical logic programs by considering a special class of ordered logic programs, called ordered versions EXLOP of logic programs: the well-founded gelder ross principles resp. the stable partial models gelfond stable sacca zaniolo deterministic of a classical logic program coincide with the well-founded resp. the stable partial models of its ordered version. In addition, in the case of ordered version of logic programs, the well-founded and the stable partial models coincide with the sceptical and the credulous partial models respectively.