A Nonmonotonic Disputation-Based Semantics and Proof Procedure for Logic Programs

Michael Thielscher[1]
International Computer Science Institute
1947 Center St., Berkeley, CA 94704-1198
michaelt@icsi.berkeley.edu

Abstract

Logic programs with nonmonotonic negation are embedded in a general, abstract disputation-based framework for nonmonotonic logics. This formalization induces a particular semantics, which is proved to extend well-founded semantics and whose expanded expressiveness is illustrated by different examples involving reasoning by cases. Moreover, we develop a formal proof procedure for skeptical reasoning in the general disputation framework. Its adaption to the logic programming context provides a goal-oriented and local proof procedure for the induced semantics.

1 Introduction

Initiated by the comparative studies of the closed world-assumption [21] and the negation as failure-rule [3] (see, e.g., [24]), the correspondence between logic programming with negation and nonmonotonic reasoning has been subject of many research activities and led to a number of fruitful results. The relationship between the two areas manifests in two directions. On the one hand, logic programs with negation have been shown to provide means for encoding problem domains involving default conclusions or implicit assumptions of minimality (see, e.g., [17]). On the other hand, the adaption of nonmonotonic reasoning principles has inspired various new semantics for logic programs, which show a wider range of applicability than the original so-called completion semantics [3]; for an overview see, e.g., [16, 1, 6]. Moreover, formal properties characterizing nonmonotonic logics have been transferred to the logic programming paradigm in order to assess and compare different semantics [4, 5].

In this paper, we pursue the latter line of research by investigating the applicability of an abstract, argumentation-based framework designed for formalizing skeptical reasoning in various concrete nonmonotonic logics. Unlike other general argumentation-based approaches, for instance [8], this framework has been especially designed for skeptical reasoning in form of a *disputation*. The general idea is to model two disputants that bring up arguments

[1] On leave from FG Intellektik, TH Darmstadt (Germany)

supporting the claim under consideration and counter-arguments against it, respectively, until one of the two arguers is unable to stick to the own standpoint any longer. A decisive advantage of this procedure is that proving is performed 'locally' insofar as it does not necessarily require to investigate all nor even entire "extensions"—in terms of extension-based nonmonotonic logics—to establish skeptical provability. This concept has been successfully applied to different nonmonotonic logics such as Poole's Theorist approach, Circumscription, and Default Logic (see [18, 12, 25, 28]) and led to actual implementations [19, 12, 25, 22].

Recently, the underlying ideas have been put on formal grounds by defining an abstract disputation-theoretic framework [26]. In this paper, we show how the general approach can be applied to logic programming with negation. This is accomplished by defining appropriate notions for both arguments and the way they may give rise to conflicts. This induces a particular semantics for arbitrary logic programs with negation. It turns out that this semantics extends well-founded semantics (WFS) [29] and supports, for instance, so-called "reasoning-by-cases" in various forms.

Moreover, we develop a formal proof procedure for skeptical entailment in the general disputation theory framework. This procedure models a dialogue as described above. The formalization of logic programs as disputation theories then allows us to employ our proof procedure in this special context. This essentially requires two suitable sub-routines for finding sound arguments and for generating appropriate counter-arguments, respectively. By achieving this for the case of propositional logic programs, we obtain a proof procedure suitable for the induced nonmonotonic semantics. The resulting algorithm has been prototypically implemented in PROLOG.

The rest of this paper is organized as follows. In the next section, 2, we recapitulate the basic notions underlying formal disputation theories, as introduced in [26]. In Section 3, we formalize logic programs as disputation theories, prove that the resulting induced semantics extends well-founded semantics, and illustrate the additional expressiveness. In Section 4, we then develop an abstract proof procedure for skeptical reasoning in arbitrary disputation theories. On this basis, in Section 5 we create an algorithm, including a simple loop checking mechanism, to compute entailment wrt. our semantics of logic programs. Finally, our results are discussed and potential extensions are outlined in Section 6. Due to lack of space, the proofs for the results in Section 3 and 4 had to be omitted; they can be found in [27].

2 Disputation Theories

The *arguments* constitute the basic entities in any particular disputation theory. In the general framework, an argument is considered a most abstract object without bearing a specific internal structure. Each argument is employed to support a certain *claim*. Claims, too, are considered abstract

objects in the general framework. While each argument is uniquely associated with a claim, claims themselves may be supported by several arguments or even by none of them.

Some arguments may conflict with other arguments. This is formalized by explicitly stating which subsets of arguments cannot reasonably be employed together. This is generally not restricted to sets consisting of two elements; that is, even a single argument may be considered conflicting, namely, if it is not reasonable in the current state of affairs; also, combining, say, three arguments might cause a conflict even though any two of them are mutually acceptable:

Definition 1 A *disputation theory* is a triple $\langle AR, CL, \mathit{conflict}\rangle$ where AR is a set of *arguments*, CL is a set of *claims*, and $\mathit{conflict} \subseteq 2^{AR}$ such that $\emptyset \notin \mathit{conflict}$ and each $\mathcal{A} \in \mathit{conflict}$ is finite. Each argument $a \in AR$ is associated with a particular claim, written $\mathit{claims}(a) \in CL$. ∎

Here and in Section 4, for illustration we use an example disputation theory with arguments $AR = \{\mathtt{A}_1, \mathtt{A}_2, \mathtt{B}_1, \mathtt{B}_2\}$ and claims $CL = \{\mathtt{C}_1, \mathtt{C}_2, \mathtt{C}_3\}$ such that $\mathit{claims}(\mathtt{A}_1) = \mathtt{C}_1$, $\mathit{claims}(\mathtt{B}_1) = \mathtt{C}_2$, $\mathit{claims}(\mathtt{A}_2) = \mathit{claims}(\mathtt{B}_2) = \mathtt{C}_3$, and

$$\mathit{conflict} = \{\{\mathtt{A}_1, \mathtt{B}_1\}, \{\mathtt{A}_1, \mathtt{B}_2\}, \{\mathtt{A}_2, \mathtt{B}_1\}, \{\mathtt{A}_2, \mathtt{B}_2\}\} \tag{1}$$

Arguments can be grouped together in so-called *extensions*, with the aim of establishing a reasonable set of beliefs on the basis of the available arguments. Reasonable means not to simultaneously believe in arguments which are conflicting but to accept any argument which does not conflict with the set of beliefs:

Definition 2 Let $DT = \langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory. A set $\mathcal{A} \subseteq AR$ is *conflict-free* iff there is no $\mathcal{B} \subseteq \mathcal{A}$ such that $\mathcal{B} \in \mathit{conflict}$. An *extension* of DT is a maximal (wrt. set inclusion) conflict-free set. ∎

E.g., our example theory gives rise to two extensions, namely, $E_1 = \{\mathtt{A}_1, \mathtt{A}_2\}$ and $E_2 = \{\mathtt{B}_1, \mathtt{B}_2\}$. No mixture between these extensions is possible since the elements of E_1 are in pairwise conflict with the elements of E_2 (c.f. (1)).

As usual, the concept of (possibly multiple) extensions gives rise to two different notions of entailment, namely, a credulous and a skeptical one; the focus here is on the latter:

Definition 3 Let $DT = \langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory then a claim $\varphi \in CL$ is *skeptically entailed* iff it is claimed in every extension of DT by some argument. ∎

To ensure skeptical entailment, one generally needs to consider more than a single argument supporting the claim under consideration, each of which holds in some but not all extensions [26]:

Definition 4 Let $\langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory and $\varphi \in CL$ a claim. A set $\mathcal{P} \subseteq AR$ of arguments, each of which claims φ, is a *skeptical proof* for φ iff for each conflict-free set of arguments $\mathcal{A} \subseteq AR$ there is some $p \in \mathcal{P}$ such that $\mathcal{A} \cup \{p\}$ is conflict-free. ∎

In other words, a skeptical proof contains, for each reasonable set of arguments, an argument that can be added without getting tangled up. For instance, we have the skeptical proof $\{\mathtt{A}_2, \mathtt{B}_2\}$ for $\mathtt{C}_3$ in our example—while it is impossible to find a skeptical proof for, say, $\mathtt{C}_1$. It is easy to verify that a claim is skeptically entailed iff it admits a skeptical proof [26].

3 Logic Programs as Disputation Theories

We now discuss how any logic program with nonmonotonic negation can be formalized as disputation theory. The reader is assumed familiar with basic concepts of logic programming, as can be found in, e.g., the textbook [15]. We assume that the programs and goal clauses under consideration be built up from a countable set of predicates and functions given once and forever. We call $p_0 \leftarrow p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n$ a *general clause* if each p_i is an atomic formula ($0 \leq i \leq n$) and $0 \leq m \leq n$. A *general logic program* P is a set of general program clauses. By $\mathit{grd}(P)$ we denote the set of all ground instances of the clauses in P.

When aiming at formalizing a logic program as disputation theory, the main task is to establish a suitable concept of arguments, their claims, and how they give rise to conflicts. Suitable means to capture the intended meaning of a clause $p_0 \leftarrow p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n$, namely, that p_0 is derivable if so are $p_1, \ldots, p_m$ but none of $p_{m+1}, \ldots, p_n$. From this perspective, an argument claiming some atom p will consist of a set d of negative literals such that p is derivable assuming d. To this end, we employ the following basic notions, which have previously been used in other argumentation-based approaches to logic programming (e.g., [8, 9]):

Definition 5 Let P be a general logic program, d a set of negative ground literals and φ a ground atom. A *d-based chain for φ* is a sequence $e_0, e_1, \ldots, e_k$ of ground atoms such that $e_k = \varphi$ and, for each $0 \leq i \leq k$, $e_i \leftarrow p_1, \ldots, p_m, \neg p_{m+1}, \ldots, \neg p_n \in \mathit{grd}(P)$ and $p_1, \ldots, p_m \in \{e_0, \ldots, e_{i-1}\}$ and $\neg p_{m+1}, \ldots, \neg p_n \in d$.

A *support* for a ground atom φ is a set of ground negative literals d such that there exists a d-based chain for φ. If φ is a negative ground literal then d is support iff $\varphi \in d$. If d, d' are two sets of negative ground literals then d' is said to *contradict* d iff there exists a d'-based chain for some ground atom p such that $\neg p \in d$. ∎

On this basis, an argument will be a pair consisting of the literal it claims plus some suitable support.

To define an adequate notion of conflicts among arguments, we have to take into account a well-known peculiarity of the intended meaning of logic programs with nonmonotonic negation—compared to other nonmonotonic logics: Consider the program $P_1 = \{\mathtt{q} \leftarrow \neg\mathtt{p}\}$. Its intended meaning is that $\mathtt{p}$ is not derivable and, hence, is taken false, which is why $\mathtt{q}$ is derivable, hence true.[2] This is reflected in the following main definition, which shows how any general logic program can be interpreted as disputation theory:

Definition 6 Let P be a general logic program. The *corresponding disputation theory* $DT_P = \langle AR_P, CL_P, \mathit{conflict}_P \rangle$ is defined as follows:

1. $AR_P = \{(\varphi, d) : \varphi \text{ ground literal}, d \text{ support for } \varphi\}$;
2. $CL_P = \{\varphi : \varphi \text{ ground literal}\}$, and $\mathit{claims}((\varphi, d)) = \varphi$; and
3. $\mathit{conflict}_P$ consists of all finite sets $\mathcal{A} \subseteq AR_P$ such that
   (a) $\bigcup_{(\varphi,d)\in\mathcal{A}} d$ contradicts itself, or
   (b) there exists some d' contradicting $\bigcup_{(\varphi,d)\in\mathcal{A}} d$ but not vice versa.

■

Via Definition 3, this correspondence induces a particular semantics for general logic programs. An important property of this semantics is that it extends well-founded semantics, that is, given a program P any literal entailed by WFS(P) is also skeptically entailed in DT_P. Prior to proving this, for the sake of illustration we discuss some example programs, two of which show why our semantics is a proper extension of WFS.

First, recall the singleton program $P_1 = \{\mathtt{q} \leftarrow \neg\mathtt{p}\}$. Assuming the underlying alphabet consists of the two predicates $\mathtt{p}, \mathtt{q}$ only, the corresponding disputation theory, $DT_{P_1} = \langle AR_{P_1}, CL_{P_1}, \mathit{conflict}_{P_1} \rangle$, is as follows:

$$\begin{aligned}
AR_{P_1} &= \{(\neg\mathtt{p}, \{\neg\mathtt{p}\}), (\neg\mathtt{p}, \{\neg\mathtt{p}, \neg\mathtt{q}\}), (\mathtt{q}, \{\neg\mathtt{p}\}), (\mathtt{q}, \{\neg\mathtt{p}, \neg\mathtt{q}\}),\\
&\qquad (\neg\mathtt{q}, \{\neg\mathtt{q}\}), (\neg\mathtt{q}, \{\neg\mathtt{p}, \neg\mathtt{q}\})\}\\
CL_{P_1} &= \{\mathtt{p}, \neg\mathtt{p}, \mathtt{q}, \neg\mathtt{q}\}\\
\mathit{conflict}_{P_1} &= \{\mathcal{A} \subseteq AR_{P_1} : \textstyle\bigcup_{(\varphi,d)\in\mathcal{A}} d = \{\neg\mathtt{p}, \neg\mathtt{q}\}\} \quad \text{(according to 3.(a))}\\
&\quad \cup \{\mathcal{A} \subseteq AR_{P_1} : \textstyle\bigcup_{(\varphi,d)\in\mathcal{A}} d \supseteq \{\neg\mathtt{q}\}\} \quad \text{(according to 3.(b))}
\end{aligned}$$

Note that any argument containing $\neg\mathtt{q}$ in its support forms a singleton conflicting set due to the fact that $d' = \{\neg\mathtt{p}\}$ contradicts $d = \{\neg\mathtt{q}\}$ but not vice versa (c.f. Definition 6, clause 3.(b)). Hence, DT_{P_1} admits a unique extension, namely, $E = \{(\neg\mathtt{p}, \{\neg\mathtt{p}\}), (\mathtt{q}, \{\neg\mathtt{p}\})\}$. In other words, both $\neg\mathtt{p}$ and $\mathtt{q}$ are skeptically entailed, as intended.

The following program, P_2, shows that, in contrast to WFS, our semantics supports what is usually referred to as "reasoning-by-cases" (see, for

[2] This contrasts the logic programming intuition with, for instance, simple (i.e., without preferences among the predicates) minimization since the corresponding implication $\neg\mathtt{p} \supset \mathtt{q}$ admits two minimal models, viz. $M_1 = \{\mathtt{q}\}$ and also $M_2 = \{\mathtt{p}\}$.

instance, [23]):

$$\begin{array}{lcl} \mathtt{q} & \leftarrow & \neg\mathtt{p} \\ \mathtt{p} & \leftarrow & \neg\mathtt{q} \\ \mathtt{r} & \leftarrow & \mathtt{p} \\ \mathtt{r} & \leftarrow & \mathtt{q} \end{array} \tag{2}$$

Assume again that the underlying alphabet consists of exactly the predicates occurring in this program. Among others, the corresponding set of arguments, AR_{P_2}, contains $(\mathtt{q}, \{\neg\mathtt{p}\})$, $(\mathtt{p}, \{\neg\mathtt{q}\})$ and, therefore, $(\mathtt{r}, \{\neg\mathtt{p}\})$, $(\mathtt{r}, \{\neg\mathtt{q}\})$. Now, observe that $d = \{\neg\mathtt{p}\}$ and $d' = \{\neg\mathtt{q}\}$ contradict each other without being self-contradictory; hence, neither argument having just $\{\neg\mathtt{p}\}$ or else just $\{\neg\mathtt{q}\}$ as support forms a singleton conflicting set. In contrast, $\{(\neg\mathtt{r}, \{\neg\mathtt{r}\})\} \in \mathit{conflict}_{P_2}$ since $d' = \{\neg\mathtt{p}\}$ (as well as $d' = \{\neg\mathtt{q}\}$) contradicts $d = \{\neg\mathtt{r}\}$ but not vice versa. Moreover, it does not help to extend the support for $\neg\mathtt{r}$ by, say, $\neg\mathtt{p}$ (in order to 'defend' against the 'attack' $d' = \{\neg\mathtt{q}\}$) since $\{\neg\mathtt{p}, \neg\mathtt{r}\}$ contradicts itself. Therefore, DT_{P_2} admits two extensions, viz.

$$\begin{aligned} E_1 &= \{(\neg\mathtt{p}, \{\neg\mathtt{p}\}), (\mathtt{q}, \{\neg\mathtt{p}\}), (\mathtt{r}, \{\neg\mathtt{p}\})\} \\ E_2 &= \{(\neg\mathtt{q}, \{\neg\mathtt{q}\}), (\mathtt{p}, \{\neg\mathtt{q}\}), (\mathtt{r}, \{\neg\mathtt{q}\})\} \end{aligned}$$

Hence, $\mathtt{r}$ is skeptically entailed since it is claimed in both extensions.

Finally, consider the following program, P_3, which can be regarded as a "reasoning-by-cases" example dual to P_2 [13]:

$$\begin{array}{lcl} \mathtt{q} & \leftarrow & \neg\mathtt{p} \\ \mathtt{p} & \leftarrow & \neg\mathtt{q} \\ \mathtt{r} & \leftarrow & \neg\mathtt{p}, \neg\mathtt{q} \end{array} \tag{3}$$

Here, the intuitive conclusion is that since either $\mathtt{p}$ or else $\mathtt{q}$ is true, as above, $\mathtt{r}$ should be false. Our semantics supports this intuition: The reader is invited to verify that again the corresponding disputation theory admits two extensions, viz.

$$\begin{aligned} E_1 &= \{(\neg\mathtt{p}, \{\neg\mathtt{p}\}), (\neg\mathtt{p}, \{\neg\mathtt{p}, \neg\mathtt{r}\}), (\mathtt{q}, \{\neg\mathtt{p}\}), (\mathtt{q}, \{\neg\mathtt{p}, \neg\mathtt{r}\}), (\neg\mathtt{r}, \{\neg\mathtt{p}, \neg\mathtt{r}\})\} \\ E_2 &= \{(\neg\mathtt{q}, \{\neg\mathtt{q}\}), (\neg\mathtt{q}, \{\neg\mathtt{q}, \neg\mathtt{r}\}), (\mathtt{p}, \{\neg\mathtt{q}\}), (\mathtt{p}, \{\neg\mathtt{q}, \neg\mathtt{r}\}), (\neg\mathtt{r}, \{\neg\mathtt{q}, \neg\mathtt{r}\})\} \end{aligned}$$

and, therefore, $\neg\mathtt{r}$ is skeptically entailed.

Having illustrated some properties of the semantics induced by representing general logic programs as disputation theories, in the remainder of this section we formally prove that this semantics forms a proper extension of well-founded semantics. To begin with, the latter is defined as follows:

Definition 7 [29] A pair $I = (I+, I-)$ is called a *three-valued* interpretation if $I+$ (atoms that are true) and $I-$ (atoms that are false) are sets of ground atoms.

Let P be a set of general program clauses. By P^+ we denote the clause set obtained from P by deleting all clauses that contain a negative literal,

and by P^- we denote the clause set obtained from P by deleting all negative literals. Moreover, we define $T^3(P)$ as the three-valued interpretation $(M_{P^+}, \overline{M_{P^-}})$, where M_P (resp. $\overline{M_P}$) denotes the minimal Herbrand model of a program P (resp. all ground atoms not in M_P).

Let P be a general logic program and I a three-valued interpretation. By $P\,|_I$ we denote the program obtained from $grd(P)$ by deleting all clauses containing a literal that is false in I and by deleting all literals that are true in I. Let Φ_P be a unary function on three-valued interpretations such that $\Phi_P(I) = T^3(P\,|_I)$ then the *well-founded model* of P, written WF_P, is the least fixpoint of Φ_P. ∎

It has been shown in [29] that this definition is applicable to arbitrary general logic programs insofar as monotonicity of Φ guarantees uniqueness of $WF_P = (WF+, WF-)$ and we always have $WF+ \cap WF- = \emptyset$. If p is a ground atom then $WF_P \models p$ iff $p \in WF+$, and $WF_P \models \neg p$ iff $p \in WF-$.

Proving that our semantics extend WFS requires three observations, below stated as lemmas. Their proofs can be found in [27]. To ease presentation, we assume that the elements of the second component, $I-$, of a three-valued interpretation $(I+, I-)$ be *negated* (ground) atoms.

Lemma 8 *Let P be a general logic program with well-founded model $WF_P = (WF+, WF-)$ then $WF-$ does not contradict itself.*

Lemma 9 *Let P be a general logic program, and let $I = (I+, I-)$ be a three-valued interpretation such that $I- \subseteq \Phi-$, where $\Phi_P(I) = (\Phi+, \Phi-)$. If there exists an $I-$-based chain for each $p \in I+$, then there exists a $\Phi-$-based chain for each $p \in \Phi+$.*

Lemma 10 *Let P be a general logic program and $I = (I+, I-)$ be a three-valued interpretation such that each $p \in I+$ is $I-$-supported. If each set d of negative ground literals which contradicts $I-$ is precluded,[3] then this holds for $\Phi-$, too, where $\Phi_P(I) = (\Phi+, \Phi-)$.*

On this basis, we can prove the following relationship between well-founded semantics and the semantics induced via Definitions 6 and 3:

Theorem 11 *Let P be a logic program and DT_P the corresponding disputation theory. If φ is a ground literal such that $WF_P \models \varphi$ then φ is skeptically entailed in DT_P.*

4 Proving by Disputation

This section is devoted to a theory for constructing skeptical proofs by means of alternately generating arguments and counter-arguments. Informally, the

[3] We call a set d of negative literals *precluded* if d contradicts itself or there is some d' contradicting d but not vice versa.

basic idea is to first come up with an argument p supporting the claim φ under consideration. Thereafter, we investigate all sets of arguments $\mathcal{A}$ which are, in some sense, *incompatible* with p and try to verify that they instead admit other arguments supporting φ. Towards this end, we need to make precise the notions of both incompatibility and "$\mathcal{A}$ admits an argument claiming φ" (below formalized as "φ being $\mathcal{A}$-*claimable*"):

Definition 12 Let $\langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory. Two conflict-free sets $\mathcal{A}, \mathcal{B} \subseteq AR$ are *compatible* (resp. *incompatible*) iff $\mathcal{A} \cup \mathcal{B}$ is (resp. is not) conflict-free. An $\mathcal{A}$-*extension* is an extension compatible with $\mathcal{A}$. A claim $\varphi \in CL$ is $\mathcal{A}$-*claimable* iff there is some $a \in AR$ such that $\mathit{claims}(a) = \varphi$ and $\{a\}$ is compatible with $\mathcal{A}$. ∎

Skeptical entailment can now be proved by generating a suitable argument and, then, investigating all incompatible (wrt. the former) arguments:

Proposition 13 *Let $\langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory, and let $\varphi \in CL$ be a claim. Then φ is skeptically entailed iff the following holds: Let p be any conflict-free argument claiming φ, then for each conflict-free $\mathcal{A} \subseteq AR$ which is incompatible with $\{p\}$, φ is $\mathcal{A}$-claimable.*

The reader should notice that, according to this result, argument p can be chosen arbitrarily so that once we find some $\{p\}$-incompatible $\mathcal{A}$ such that φ is not $\mathcal{A}$-claimable, φ is known not to be skeptically entailed—there is no need to backtrack over the first choice of p.

Recall our example from Section 2. Starting with $p = \mathtt{A}_2$ as argument claiming $\mathtt{C}_3$, there are three conflict-free sets of arguments incompatible with p according to Definition 12, namely, all non-empty subsets of $\{\mathtt{B}_1, \mathtt{B}_2\}$. Each of these is compatible with argument $\mathtt{B}_2$, which also claims $\mathtt{C}_3$. Hence, this claim is skeptically entailed.

While this procedure avoids investigation of all extensions compatible with the first argument being generated to support the claim, it still requires exhaustive investigation among incompatible sets of arguments. We therefore refine our method by searching for *minimal* counter-arguments $\mathcal{A}$ only. The notion of minimality is based on a given partial ordering on sets of arguments that should obey the following property:

Definition 14 Let $\langle AR, CL, \mathit{conflict}\rangle$ be a disputation theory. A partial ordering $\preceq \subseteq 2^{AR} \times 2^{AR}$ is called *conflict preserving* iff the following holds for each $\mathcal{A}, \mathcal{B} \subseteq AR$ such that $\mathcal{A} \preceq \mathcal{B}$: For each $\mathcal{C} \subseteq AR$, if $\mathcal{C}$ and $\mathcal{A}$ are incompatible then so are $\mathcal{C}$ and $\mathcal{B}$. ∎

For instance, in our example theory we might use $\{\} \preceq \mathcal{A}$ for each $\mathcal{A} \subseteq AR$; $\{\mathtt{A}_1\} \preceq \mathcal{A}$ for each non-empty $\mathcal{A} \subseteq \{\mathtt{A}_1, \mathtt{A}_2\}$; and $\{\mathtt{B}_1\} \preceq \mathcal{B}$ for each non-empty $\mathcal{B} \subseteq \{\mathtt{B}_1, \mathtt{B}_2\}$. This ordering is conflict preserving wrt. (1).

Now, given a particular, minimal counter-argument $\mathcal{A}$ (that is, a set of arguments incompatible with the argument considered first, p), we intend

to prove—by a recursive call of the entire procedure—that the claim, φ, is claimed in *every* $\mathcal{A}$-extension. This shall replace the task of verifying that φ be $\mathcal{A}'$-claimable for every $\mathcal{A}' \succeq \mathcal{A}$:

Proposition 15 *Let $\langle AR, CL, conflict\rangle$ be a disputation theory, $\mathcal{A} \subseteq AR$ a conflict-free set of arguments and $\varphi \in CL$ a claim. Then φ is $\mathcal{A}'$-claimable for each conflict-free $\mathcal{A}' \subseteq AR$ such that $\mathcal{A} \preceq \mathcal{A}'$ iff φ is claimed in every $\mathcal{A}$-extension.*

Of course, we have to ensure that every incompatible (wrt. p) $\mathcal{A}'$ is covered; that is, we have to find a whole collection of minimal counter-arguments which is complete with that respect. Let p be an argument. We call a set Ω of sets of arguments *p-complete conflicting* if for any conflict-free set of arguments $\mathcal{A}'$ either $\mathcal{A}'$ and $\{p\}$ are compatible or there exists some $\mathcal{A} \in \Omega$ such that $\mathcal{A} \preceq \mathcal{A}'$. We then obtain the following main result:

Theorem 16 *Let $\langle AR, CL, conflict\rangle$ be a disputation theory and $\varphi \in CL$ a claim. Then φ is skeptically entailed iff the following holds: Let p be an arbitrary conflict-free argument claiming φ and let Ω be a p-complete conflicting set, then for each $\mathcal{A} \in \Omega$, φ is claimed in every $\mathcal{A}$-extension.*

For instance, given the ordering above plus argument $p = \mathtt{A}_2$ claiming $\mathtt{C}_3$, the singleton set $\Omega = \{\{\mathtt{B}_1\}\}$ is p-complete conflicting. The only $\{\mathtt{B}_1\}$-extension is $\{\mathtt{B}_1, \mathtt{B}_2\}$, and claim $\mathtt{C}_3$ is also claimed in this extension (via $\mathtt{B}_2$). This proves skeptical entailment of $\mathtt{C}_3$ according to the theorem.

It is possible to extract from this theorem the following abstract proof procedure for skeptical entailment, which additionally offers a proof $\mathcal{P}$ (in the sense of Definition 4) in case of success. To obtain the latter, we take the argument p generated at the beginning and combine it with all arguments generated in each recursive call:

1. Let p be a conflict-free argument claiming φ. If none exists, return `no`; else let $\mathcal{P} := \{p\}$.
2. Let Ω be p-complete conflicting. For all $\mathcal{A} \in \Omega$ do:
   2.1. Let $\mathcal{P}'$ be a skeptical proof for φ (obtained via a recursive call of this procedure) in the $\mathcal{A}$-restricted disputation theory. If it does not exist, return `no`; else let $\mathcal{P} := \mathcal{P} \cup \mathcal{P}'$.
3. Return $\mathcal{P}$.

Here, "$\mathcal{A}$-restricted disputation theory" means to restrict attention to arguments compatible with $\mathcal{A}$. In our example, we might start with $p = \mathtt{A}_2$ as above (claiming $\mathtt{C}_3$), and the recursive call using $\mathcal{A} = \{\mathtt{B}_1\}$ adds $\mathcal{P}' = \{\mathtt{B}_2\}$ to $\mathcal{P} = \{\mathtt{A}_2\}$, which results in the skeptical proof $\{\mathtt{A}_2, \mathtt{B}_2\}$ for $\mathtt{C}_3$.

In order to employ this abstract procedure in the context of a concrete nonmonotonic logic, steps 1 and 2 need to be made precise by developing sub-routines for both finding conflict-free arguments and generating complete conflicting sets. (See [19, 25, 28] for how these procedures have been designed for example nonmonotonic logics). For the case of propositional logic programs, this will be accomplished in the following section.

```
TYPE support = SET OF negative_literal;
FUNCTION find_base(φ : atom) : support;
VAR goal, loop : SET OF atom;
    d          : support;
BEGIN
  goal := {φ}; loop := ∅; d := ∅;
  LOOP
    IF goal = ∅ THEN RETURN d;
    CHOOSE p ∈ goal;
    goal := goal \ {p}; loop := loop ∪ {p};
    CHOOSE p ← p_1, ..., p_m, ¬p_{m+1}, ..., ¬p_n ∈ P;
    IF {p_1, ..., p_m} ∩ loop ≠ ∅ THEN FAIL;        /* 1 */
    goal := goal ∪ {p_1, ..., p_m};
    d := d ∪ {¬p_{m+1}, ..., ¬p_n}
  END-LOOP
END-FUNCTION
```

Figure 1: A procedure generating support for a given atom φ on the basis of program P. The command CHOOSE describes nondeterministic choice, and FAIL initiates backtracking over all choice operations, which may finally lead to failure of the entire procedure.

5 A Proof Procedure for Logic Programs

In order to adapt our general proof procedure to logic programming, let us first consider the task to generate conflict-free arguments for some claim, i.e., literal. Following Definition 6, this amounts to finding some supporting negative literals d such that d is not precluded (c.f. Footnote 3). According to Definition 5, a support for a negative literal is any set d containing it. A support for an atom is essentially obtained by applying SLD-resolution plus collecting, as the support, all negative literals occurring in applied program clauses. In order to prevent infinite SLD-derivations, we additionally employ a simple loop detecting mechanism. Figure 1 depicts the resulting procedure, which defines a function *find_base*(φ) and computes a set of negative literals d which support the atom φ wrt. the given program clauses. We use the variable *loop* to collect all resolved atoms and, thereby, to avoid applying clauses with previously resolved atoms in the body (see /* 1 */). Hence, a member p of the goal should not be selected until all clauses have been applied which are needed for a successful derivation and have p in their body. In case of finite propositional logic programs this procedure is guaranteed to terminate.

Having generated a support d for the claim at hand, it remains to verify that d does not contradict itself but contradicts any d' which in turn contradicts d. The latter may require to extend the computed support: Recall program (3). The support $\{\neg\mathtt{r}\}$ for $\neg\mathtt{r}$ does not provide a conflict-free ar-

```
FUNCTION find_argument(D : support; φ : literal) : support;
VAR verified, attacked : SET OF negative_literal;
    d, d', d̄           : support;
BEGIN
  IF is_atom(φ) THEN CHOOSE d := find_base(φ) ELSE d := {φ};
  d := d ∪ D; verified := ∅; attacked := ∅;
  LOOP
    IF verified = d THEN RETURN d;
    LET ¬p ∈ d \ verified;                          /* 1 */
    FOR ALL d̄ := find_base(p) DO                    /* 2 */
      CHOOSE ¬q ∈ d̄;
      CHOOSE d' := find_base(q);                    /* 3 */
      attacked := attacked ∪ {¬q};
      d := d ∪ d';                                  /* 4 */
      IF attacked ∩ d ≠ ∅ THEN FAIL;                /* 5 */
      verified := verified ∪ {¬p}
    END-FOR;
  END-LOOP;
END-FUNCTION
```

Figure 2: A procedure generating conflict-free arguments for a given claim (i.e., literal) φ while considering 'pre-constraints' given by $\mathcal{D}$ (c.f. Footnote 4).

gument, for $\{\neg\mathtt{p}, \neg\mathtt{q}\}$ contradicts $\{\neg\mathtt{r}\}$ but not vice versa. Yet adding, say, $\neg\mathtt{p}$ to the support yields the conflict-free argument $(\neg\mathtt{r}, \{\neg\mathtt{p}, \neg\mathtt{r}\})$ claiming $\neg\mathtt{r}$. This is reflected in our procedure depicted in Figure 2, which defines a function *find_argument*$(\mathcal{D}, \varphi)$ and computes a conflict-free, $\mathcal{D}$-compatible[4] argument claiming literal φ: Given a support d, we have to analyze any possible contradiction. To this end, we 'verify' each $\neg p \in d$ (see $/* \ 1 \ */$) by investigating each $\overline{d}$ which contradicts d by supporting p (see $/* \ 2 \ */$). More precisely, we have to ensure that d in turn contradicts $\overline{d}$, which is the case if we can find some $\neg q \in \overline{d}$ such that q is supported by d (see $/* \ 3 \ */$). As indicated above, this may require to extend d (see $/* \ 4 \ */$). Finally, $/* \ 5 \ */$ ensures that the overall resulting support d does not contradict itself. As an example, the reader may verify that, given program (3), calling *find_argument*$(\emptyset, \neg\mathtt{r})$ results in $(\neg\mathtt{r}, \{\neg\mathtt{p}, \neg\mathtt{r}\})$ as a conflict-free (and $\emptyset$-compatible) argument for $\neg\mathtt{r}$.

Finally, we need a procedure that generates a complete conflicting set of arguments given some conflict-free argument (φ, d). This can easily be obtained by employing our function *find_argument* to compute all conflict-

[4] Recall that the entire algorithm is a recursive procedure where each recursive call deals with an "$\mathcal{A}$-restricted" disputation theory for some set $\mathcal{A}$ of arguments (see the end of Section 4). In the logic programming context, this can be obtained by requiring that every support include the negative literals $\mathcal{D} := \cup_{(\varphi,d)\in\mathcal{A}} d$ —a property which we call $\mathcal{D}$*-compatibility*.

```
FUNCTION complete_conflicting(D : support; d : support; ) : SET OF support;
VAR d : support;
    Ω : SET OF support;
BEGIN
  FOR ALL ¬p ∈ d DO
    FOR ALL d̄ := find_argument(D, p) DO Ω := Ω ∪ {d̄};
  RETURN Ω
END-FUNCTION
```

Figure 3: A procedure generating complete conflicting sets for some argument with support d while considering 'pre-constraints' given by $\mathcal{D}$.

free arguments supporting some p with $\neg p \in d$. This is depicted in Figure 3, where a function *complete_conflicting*$(\mathcal{D}, d)$ is defined whose execution yields a (φ, d)-complete conflicting set wrt. an $\mathcal{A}$-restricted disputation theory, where $\mathcal{D} = \bigcup_{(\varphi,d)\in\mathcal{A}} d$. E.g., recall program (2). Assume we have already computed $(\mathtt{r}, \{\neg\mathtt{p}\})$ as a conflict-free argument for $\mathtt{r}$, then the function call *complete_conflicting*$(\emptyset, \{\neg\mathtt{p}\})$ yields $\{\{\neg\mathtt{q}\}\}$ since $\{\neg\mathtt{q}\}$ is the (only) outcome of *find_argument*$(\emptyset, \mathtt{p})$. Or, assume we are interested in a $(\mathtt{r}, \{\neg\mathtt{q}\})$-complete conflicting set in the $(\mathtt{p}, \{\neg\mathtt{q}\})$-restricted theory then *complete_conflicting*$(\{\neg\mathtt{q}\}, \{\neg\mathtt{q}\})$ yields $\emptyset$, for *find_argument*$(\{\neg\mathtt{q}\}, \mathtt{q})$ fails—there is obviously no conflict-free argument for $\mathtt{q}$ which contains $\neg\mathtt{q}$ in its support.

On the basis of these sub-routines, we can now apply our general algorithm from Section 4 to decide entailment wrt. the semantics obtained in Section 3. As an example, consider program (2) again. We intend to prove $\mathtt{r}$ be entailed:

1. Calling *find_argument*$(\emptyset, \mathtt{r})$ yields $\{\neg\mathtt{p}\}$;[5] hence, $(\mathtt{r}, \{\neg\mathtt{p}\})$ constitutes a conflict-free argument claiming $\mathtt{r}$. Let $\mathcal{P} = \{(\mathtt{r}, \{\neg\mathtt{p}\})\}$.
2. As argued above, calling *complete_conflicting*$(\emptyset, \{\neg\mathtt{p}\})$ yields $\{\{\neg\mathtt{q}\}\}$; hence, $\{\{(\mathtt{p}, \{\neg\mathtt{q}\})\}\}$ is $(\mathtt{r}, \{\neg\mathtt{p}\})$-complete conflicting.
   2.1. The recursive call first requires to find a conflict-free, $\{(\mathtt{p}, \{\neg\mathtt{q}\})\}$-compatible argument for $\mathtt{r}$. Calling *find_argument*$(\{\neg\mathtt{q}\}, \mathtt{r})$ results in $\{\neg\mathtt{q}\}$.[6] Now, *complete_conflicting*$(\{\neg\mathtt{q}\}, \{\neg\mathtt{q}\})$ yields $\emptyset$, as argued above, which is why we are done; that is, the recursive call yields $\mathcal{P}' = \{(\mathtt{r}, \{\neg\mathtt{q}\})\}$ as skeptical proof for $\mathtt{r}$ in the $\{(\mathtt{p}, \{\neg\mathtt{q}\})\}$-restricted disputation theory.
3. The computed proof for $\mathtt{r}$ thus is $\mathcal{P} \cup \mathcal{P}' = \{(\mathtt{r}, \{\neg\mathtt{p}\}), (\mathtt{r}, \{\neg\mathtt{q}\})\}$.

[5] Note that the result may equally well be $\{\neg\mathtt{q}\}$, depending on the nondeterministic choice.

[6] Note that now this is the only possible outcome since $(\neg\mathtt{p}, \{\neg\mathtt{p}\})$ is not $\{(\mathtt{p}, \{\neg\mathtt{q}\})\}$-compatible.

Internet Availability

We have implemented this proof procedure in PROLOG. The program `lp.pl` is available via ftp at `aida.intellektik.informatik.th − darmstadt.de` and can be found, together with [27], in `pub/AIDA/Disputation/` .

6 Discussion

We have shown how any logic program with nonmonotonic negation can be embedded in a general, disputation-based framework for nonmonotonic reasoning. We have proved that the induced semantics extends well-founded semantics, and we have illustrated its expanded expressiveness regarding different example programs that involve "reasoning-by-cases." Moreover, we have developed a general skeptical proof procedure for disputation theories, and we have adapted this procedure to the logic programming paradigm.

The fact that the induced semantics extends WFS distinguishes our formalization from other argumentation-based approaches to logic programming with negation, such as [7, 14, 9]. While the notion of "support" in Definition 5 has been adopted from these methods, they employ different intuitions when formalizing conflicts among arguments. This results in a close correspondence to stable models [11] rather than WFS.[7]

A number of different semantics all of which extend WFS have been proposed in the past years, most of which, like WFS^+ and WFS' (see [4]) or WFS_{bc} [23], produce a different result than ours when applied to program (3). An extension which does treat this program similarly is "extended WFS" as defined in [13], which, on the other hand, allows to derive $\mathtt{p}$ from the program $P = \{\mathtt{p} \leftarrow \neg\mathtt{p}\}$, while neither $\mathtt{p}$ nor $\neg\mathtt{p}$ is skeptically entailed in the corresponding disputation theory DT_P. As regards yet another extension, namely, GWFS [2], Example 4.3 of [5] may serve as illustration for the differences to the semantics obtained in this paper. On the other hand, our Definitions 5 and 6 resemble the basic notions of *regular model semantics* [30]. We even conjecture that there is a one-to-one correspondence between the extensions of DT_P and the regular models of a program P, in which case our skeptical proof procedure would also be suitable for this and other equivalent semantics (see [30]).

The general disputation framework has been developed especially in view of skeptical reasoning in nonmonotonic logics. The underlying concept is to model a disputation between two arguers who bring up arguments supporting the claim under consideration and counter-arguments, respectively. The decisive advantage of this method is that it does neither strictly require the inspection of all extensions nor the generation of entire extensions to decide whether a formula is skeptically entailed. Thus, much like the standard

[7] However, in [9] itself an alternative formalization of conflicting arguments is proposed, which results in an equivalent of WFS. The application of our disputation framework via Definition 6 may therefore be regarded as an extension of this proposal.

resolution variants applied in the logic programming context, the proof procedure described in Section 5 is goal-oriented and local. This distinguishes it from both schemes based on stable models [11]—where unrelated clauses, like $\mathtt{p} \leftarrow \neg\mathtt{p}$, may influence the derivability of the goal under consideration—and algorithms that compute entire well-founded models. Since we have restricted our proof procedure to propositional logic programs, an important direction of future research is to lift it to the first-order case and, then, to assess the resulting calculus in comparison with existing extensions of SLDNF-resolution, like [10] (whose adequacy wrt. the argumentation-based approach [9] has been shown) or SLS-resolution [20].

References

[1] K. R. Apt and R. Bol. Logic programming and negation: A survey. *J. of Logic Programming*, 19/20:9–71, 1994.

[2] C. Baral, J. Lobo, and J. Minker. Generalized well-founded semantics for logic programs. In M. E. Stickel, ed., *Proc. of CADE*, Springer LNAI 449, p. 102–116, Kaiserslautern, Germany, July 1990.

[3] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, ed.'s, *Logic and Data Bases*, p. 293–322. Plenum Press, 1978.

[4] J. Dix. A classification-theory of semantics of normal logic programs: I. Strong principles. *Fundamenta Informatica*, 22(3):227–255, 1995.

[5] J. Dix. A classification-theory of semantics of normal logic programs: II. Weak Principles. *Fundamenta Informatica*, 22(3):257–288, 1995.

[6] J. Dix. Semantics of logic programs: Their intuitions and formal properties. An overview. In A. Fuhrmann and H. Rott, ed.'s, *Logic, Action and Information*, p. 227–311. de Gruyter, 1995.

[7] P. M. Dung. Negation as hypotheses: An abductive foundation for logic programming. In K. Furukawa, ed., *Proc. of ICLP*, p. 3–17, Paris, France, June 1991. MIT Press.

[8] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming. In R. Bajcsy, ed., *Proc. of IJCAI*, p. 852–857, Chambéry, France, Aug. 1993. Morgan Kaufmann.

[9] P. M. Dung. An argumentation-theoretic foundation for logic programming. *J. of Logic Programming*, 22(2):151–177, 1995.

[10] K. Eshghi and R. A. Kowalski. Abduction compared with negation by failure. In G. Levi and M. Martelli, ed.'s, *Proc. of ICLP*, p. 234–255, Lisbon, Portugal, 1989. MIT Press.

[11] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, ed.'s, *Proc. of IJCSLP*, p. 1070–1080, Seattle, OR, 1988. MIT Press.

[12] M. L. Ginsberg. A circumscriptive theorem prover. *Artif. Intell.*, 39(2):209–230, 1989.

[13] Y. Hu and L. Y. Yuan. Extended well-founded model semantics for general logic programs. In K. Furukawa, ed., *Proc. of ICLP*, p. 412–425, Parics, France, June 1991. MIT Press.

[14] A. C. Kakas, P. Mancarella, and P. M. Dung. The acceptability semantics for logic programs. In P. V. Hentenryck, ed., *Proc. of ICLP*, p. 504–519, Santa Margherita Ligure, Italy, June 1994. MIT Press.

[15] J. W. Lloyd. *Foundations of Logic Programming*. Series Symbolic Computation. Springer, second, extended edition, 1987.

[16] J. Minker. An overview of nonmonotonic reasoning and logic programming. *J. of Logic Programming*, 17:95–126, 1993.

[17] L. M. Pereira, J. N. Aparício, and J. J. Alferes. Non-monotonic reasoning with logic programming. *J. of Logic Programming*, 17:227–263, 1993.

[18] D. Poole. A logical framework for default reasoning. *Artif. Intell.*, 36(1):27–47, 1988.

[19] D. Poole. Compiling a default reasoning system into Prolog. *New Generation Comput.*, 9(1):3–38, 1991.

[20] T. C. Przymusinski. On the declarative and procedural semantics of logic programs. *J. of Automated Reasoning*, 5:167–205, 1989.

[21] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, ed.'s, *Logic and Data Bases*, p. 55–76. Plenum Press 1978.

[22] T. Schaub and M. Thielscher. Skeptical query-answering in constrained default logic. In D. M. Gabbay, ed., *Proc. of the Int.'l Conf. on Formal and Applied Practical Reasoning*, Springer LNAI 1085, p. 567–581, Bonn, Germany, June 1996.

[23] J. S. Schlipf. Formalizing a logic for logic programming. *Annals of Mathem. and Artif. Intell.*, 5:279–302, 1992.

[24] J. C. Shepherdson. Negation as failure: A comparison of Clark's completed data base and Reiter's closed world assumption. *J. of Logic Programming*, 1:51–79, 1984.

[25] M. Thielscher. On prediction in Theorist. *Artif. Intell.*, 60(2):283–292, 1993.

[26] M. Thielscher. What is a skeptical proof? In I. Wachsmuth, C.-R. Rollinger, and W. Brauer, ed.'s, *Proc. of the German Annual Conf. on Artif. Intell. (KI)*, Springer LNAI 981, p. 161–172, Bielefeld, Germany, Sept. 1995.

[27] M. Thielscher. On a Nonmonotonic Disputation-Based Semantics and Proof Procedure for Logic Programs. Technical Report AIDA–96–09, FG Intellektik, TH Darmstadt, May 1996.

[28] M. Thielscher and T. Schaub. Default reasoning by deductive planning. *J. of Automated Reasoning*, 15(1):1–40, 1995.

[29] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. of the ACM*, 38(3):620–650, 1991.

[30] J.-H. You and L. Y. Yuan. On the equivalence of semantics for normal logic programs. *J. of Logic Programming*, 22(3):211–221, 1995.