Uniform Proofs and Disjunctive Logic Programming (Extended Abstract)[†]

Gopalan Nadathur[‡]
Department of Computer Science
University of Chicago
1100 E 58th Street, Chicago, IL 60637
gopalan@cs.uchicago.edu

Donald W. Loveland
Department of Computer Science
Duke University
Box 90129, Durham, NC 27708-0129
dwl@cs.duke.edu

Abstract

One formulation of the concept of logic programming is the notion of Abstract Logic Programming Language, introduced in [8]. Central to that definition is uniform proof, which enforces the requirements of inference direction, including goal-directedness, and the duality of readings, declarative and procedural. We use this technology to investigate Disjunctive Logic Programming (DLP), an extension of traditional logic programming that permits disjunctive program clauses. This extension has been considered by some to be inappropriately identified with logic programming because the indefinite reasoning introduced by disjunction violates the goal-oriented search directionality central to logic programming. We overcome this criticism by showing that the requirement of uniform provability can be realized in a logic more general than that of DLP under a modest, sound, modification of programs. We use this observation to derive inference rules that capture the essential proof structure of InH-Prolog, a known proof procedure for DLP.

1 Introduction

A class of sequent calculus proofs called uniform proofs is identified in [8] and used as a means for determining if a given logical language can serve as a basis for logic programming. The main characteristic of a uniform proof is that it reflects a goal-directedness in proof-search: the objective at any stage in the search is to construct a proof of a *single* (goal) formula from a collection of assumptions and if this formula is non-atomic, then the search proceeds by first processing its top-level logical symbol. Viewed differently, the logical symbols in the formulas being proved become primitives for directing search and the inference rules pertaining to these symbols become specifications of their search semantics. Classes of formulas and proof relations provide a basis for logic programming just in case provability in their context entails the existence of a uniform proof. The virtue of this "uniform proof property" is that it permits a duality between a declarative and a search-related reading for logical symbols that seems central to a programming use of logic. An auxiliary benefit of the property is that it can be used for describing efficient proof procedures for the underlying logical language [7, 10].

A central characteristic of disjunctive logic programming is that it allows for disjunctive information in the assumption set or program. The ability to express and process indefinite information and case constructions occur in real life and mechanisms to handle such specifications are important. Attention has focused on the semantics of disjunctive logic programming [1, 2, 11], proof procedures [4, 12] and applications to disjunctive databases [9, 13, 14]. One point of view of each of these topics appears in the book [3].

The basic requirement of disjunctive logic programming, that disjunctive information be representable in the assumption set, appears to be at variance with the desire for the uniform proof property to hold. For example, suppose that our assumption set contains the formula $p(a) \vee p(b)$ and that our goal is to prove $\exists x\, p(x)$. To obtain a uniform proof of this formula, it is necessary to be able to prove $p(t)$ for some specific term t; this is a natural outcome of goal-directedness and the limitation of a "goal" to a single formula. It is obviously not possible to construct such a proof. Notice, however, that $\exists x\, p(x)$ is provable from $p(a) \vee p(b)$ in most logical systems.

[†] Work on this paper has been partially supported by NSF grants CCR-92-08465 (Nadathur) and CCR-91-16203 (Loveland) and a grant from the Konrad-Zuse Programm (Nadathur).

[‡] On leave till July 1995 at Institut für Informatik, Ludwig-Maximilians Universität, München.

We attempt to reconcile this difference between the presence of disjunctive information and the desire for uniform proofs in this paper. Our approach in doing this is the following. Suppose that we wish to show that a formula G follows from a set of assumption formulas Γ in either classical or intuitionistic logic. We may, in general, not be able to do this by looking only for a uniform proof. However, we show that, if we restrict the syntax of G and the formulas in Γ in such a way that all of disjunctive logic programming is still included, a simple augmentation of Γ makes the search for a uniform proof a complete strategy. In particular, if the necessary syntactic restrictions are satisfied, then a uniform proof exists for G from $\Gamma, G \supset \bot$ if and only if there is a classical or an intuitionistic proof for G from Γ. This observation has a practical benefit in that it forms the basis for a proof procedure for our formulas (under a reduction to clausal form) that is a natural generalization of the one usually employed in logic programming. The main new aspect of this procedure is that it has a rule for "backchaining" that is sensitive to the presence of disjunctive information. The restart mechanism of nH-Prolog [4, 5] is used for dealing with this possibility and our procedure is, in fact, an abstract presentation of the InH-Prolog procedure [6].

2 Logical preliminaries

We shall work within the framework of a first-order logic in this paper. The logical symbols that we shall use are $\top$, $\bot$, $\wedge$, $\vee$, $\supset$, $\exists$, and $\forall$; the first two symbols are intended to denote the tautologous and the contradictory propositions, respectively. The symbol $\neg$ is excluded from our vocabulary, but it can be defined in terms of the other symbols present: $\neg A$ is a shorthand for $(A \supset \bot)$.

Notions of derivation that are of interest to us are formalized by sequent calculi. A sequent in our context will be a pair of multisets of formulas. The pair will, as usual, be written as $\Gamma \longrightarrow \Delta$, assuming that Γ and Δ are its elements. Such a sequent is an axiom if either $\top \in \Delta$ or for some A that is either $\bot$ or an atomic formula, it is the case that $A \in \Gamma$ and $A \in \Delta$. The rules that may be used in constructing sequent proofs are those that can be obtained from the schemata shown in Figure 1. In these schemata, Γ, Δ and Θ stand for multisets of formulas, B and D stand for formulas, c stands for a constant, x stands for a variable and t stands for a term. The notation B, Γ (Δ, B) is used here for a multiset containing the formula B whose remaining elements form the multiset Γ (respectively, Δ). Further, expressions of the form $[t/x]B$ are used to denote the result of replacing all free occurrences of x in B by t, with bound variables being renamed as needed to ensure the logical correctness of these replacements. There is the usual proviso with respect to the rules produced from the schemata $\exists$-L and $\forall$-R: the constant that replaces c should not appear in the formulas that form the lower sequent. The purpose of the schemata contr-L and contr-R is to blur the distinction between sets and multisets, and so we will be ambivalent about this difference at times. However, this distinction can be important in a situation where these contraction rules are not present.

We will be interested in three notions of derivability for sequents of the form $\Gamma \longrightarrow B$. A **C**-proof for such a sequent is a derivation obtained by making arbitrary uses of the inference rules. We denote the existence of such a proof, which is a classical proof, for the sequent by writing $\Gamma \vdash_C B$. **I**-proofs, that formalize the notion of intuitionistic derivability, are **C**-proofs in which every sequent has exactly one formula in its succedent. We write $\Gamma \vdash_I B$ to indicate the existence of an **I**-proof for $\Gamma \longrightarrow B$. Finally, a *uniform proof* is an **I**-proof in which any sequent whose succedent contains a non-atomic formula occurs only as the lower sequent of an inference rule that introduces the top-level logical symbol of that formula. Notice that if $\Gamma \longrightarrow B$ has a uniform proof, then the following must be true with respect to this proof:

(1) If B is $C \wedge D$, then the sequent must be inferred by $\wedge$-R from $\Gamma \longrightarrow C$ and $\Gamma \longrightarrow D$.

(2) If B is $C \vee D$ then the sequent must be inferred by $\vee$-R from either $\Gamma \longrightarrow C$ or $\Gamma \longrightarrow D$.

(3) If B is $\exists x\, P$ then the sequent must be inferred by $\exists$-R from $\Gamma \longrightarrow [t/x]P$ for some term t.

(4) If B is $C \supset D$ then the sequent must be inferred by $\supset$-R from $C, \Gamma \longrightarrow D$.

(5) If G is $\forall x\, P$ then, for some constant c that does not occur in the given sequent, it must be the case that the sequent is inferred by $\forall$-R from $\Gamma \longrightarrow [c/x]P$.

These properties permit the search for a uniform proof to proceed in a goal-directed fashion with the top-level structure of the goal, *i.e.*, the formula being proved, controlling the next step in the search at each stage.

We shall write $\Gamma \vdash_O B$ to denote the existence of a uniform proof for $\Gamma \longrightarrow B$. Letting $\mathcal{D}$ and $\mathcal{G}$ denote collections of formulas and $\vdash$ denote a chosen proof relation, an abstract logic programming language is defined in [8] as a triple $\langle \mathcal{D}, \mathcal{G}, \vdash \rangle$ such that, for all

$$\frac{B, B, \Gamma \longrightarrow \Delta}{B, \Gamma \longrightarrow \Delta}\ \text{contr-L} \qquad \frac{\Gamma \longrightarrow \Delta, B, B}{\Gamma \longrightarrow \Delta, B}\ \text{contr-R}$$

$$\frac{\Gamma \longrightarrow \Delta, \bot}{\Gamma \longrightarrow \Delta, D}\ \bot\text{-R}$$

$$\frac{B, D, \Gamma \longrightarrow \Delta}{B \wedge D, \Gamma \longrightarrow \Delta}\ \wedge\text{-L} \qquad \frac{\Gamma \longrightarrow \Delta, B \qquad \Gamma \longrightarrow \Delta, D}{\Gamma \longrightarrow \Delta, B \wedge D}\ \wedge\text{-R}$$

$$\frac{B, \Gamma \longrightarrow \Delta \qquad D, \Gamma \longrightarrow \Delta}{B \vee D, \Gamma \longrightarrow \Delta}\ \vee\text{-L}$$

$$\frac{\Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, B \vee D}\ \vee\text{-R} \qquad \frac{\Gamma \longrightarrow \Delta, D}{\Gamma \longrightarrow \Delta, B \vee D}\ \vee\text{-R}$$

$$\frac{B \supset D, \Gamma \longrightarrow B, \Delta \qquad D, \Gamma \longrightarrow \Theta}{B \supset D, \Gamma \longrightarrow \Delta, \Theta}\ \supset\text{-L} \qquad \frac{B, \Gamma \longrightarrow \Delta, D}{\Gamma \longrightarrow \Delta, B \supset D}\ \supset\text{-R}$$

$$\frac{[t/x]B, \Gamma \longrightarrow \Delta}{\forall x\, B, \Gamma \longrightarrow \Delta}\ \forall\text{-L} \qquad \frac{\Gamma \longrightarrow \Delta, [t/x]B}{\Gamma \longrightarrow \Delta, \exists x\, B}\ \exists\text{-R}$$

$$\frac{[c/x]B, \Gamma \longrightarrow \Delta}{\exists x\, B, \Gamma \longrightarrow \Delta}\ \exists\text{-L} \qquad \frac{\Gamma \longrightarrow \Delta, [c/x]B}{\Gamma \longrightarrow \Delta, \forall x\, B}\ \forall\text{-R}$$

Figure 1: Rules for Deriving Sequents

finite subsets $\mathcal{P}$ of $\mathcal{D}$ and all $G \in \mathcal{G}$, $\mathcal{P} \vdash G$ if and only if $\mathcal{P} \vdash_O G$.

Our final observation concerns the so-called *Cut* rule that has the following form:

$$\frac{\Gamma_1 \longrightarrow B, \Delta_1 \qquad B, \Gamma_2 \longrightarrow \Delta_2}{\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2}$$

It is well-known that this rule is admissible with respect to classical and intuitionistic provability, *i.e.*, the same set of sequents have derivations with and without this rule. We use this fact in the next section.

3 A disjunctive logic programming language

A logic programming language can be described in a proof-theoretic setting by identifying classes of formulas that can serve as program clauses and queries and a proof relation that can be used to derive a query from a given collection of program clauses. Our present interest is that we be able to represent disjunctive information in programs. This leads naturally to the definition of program clauses and queries as the (closed) G- and D-formulas given by the syntax rules

$$\begin{array}{lcl} G & ::= & A \mid G \wedge G \mid G \vee G \mid \exists x\, G \\ D & ::= & A \mid G \supset D \mid D \wedge D \mid D \vee D \mid \exists x\, D \mid \forall x\, D \end{array}$$

in which A represents an atomic formula distinct from $\bot$. The main difference between the formulas presented here and those used to define logic programming based on Horn clauses in [8] is that program clauses (*i.e.*, the D-formulas) are permitted to contain disjunctions and existential quantifications. The traditional description of disjunctive logic programming uses formulas in clausal form. The defining characteristic in this situation is that clauses with "multiple" heads are permitted in programs. It is easily seen that our D-formulas encompass such formulas. In the converse direction, our formulas can, as will become obvious presently, be translated into an equivalent clausal form.

To complete the description of our language, it is necessary to specify the proof relation that is to provide its declarative semantics. The relation that is generally employed for this purpose in disjunctive logic programming is classical provability. In the present context, we have another choice, namely intuitionistic provability. It turns out that it is actually immaterial which of these relations is used since a sequent of the kind we are interested in has a derivation under one if it has a derivation under the other. This observation is of importance in our later analysis relating to uniform provability and so is established below.

Lemma 1 *Let Γ be a multiset of formulas and let $B_1, \ldots, B_n$ be formulas. If $\Gamma \longrightarrow B_1, \ldots, B_n$ has a **C**-proof in which the rules $\supset$-R and $\forall$-R are not used, then $\Gamma \longrightarrow B_1 \vee \ldots \vee B_n$ has an **I**-proof.*

Lemma 1 is proved by an induction on the height of a (cut-free) **C**-proof of $\Gamma \longrightarrow B_1, \ldots, B_n$. It is easily seen to be true in the base case. For the inductive step, we consider the cases for the last rule. For instance, suppose the last rule is an $\exists$-R. Then the derivation at the end has the following form:

$$\frac{\Gamma \longrightarrow B_1, \ldots, [t/x]B_i, \ldots, B_n}{\Gamma \longrightarrow B_1, \ldots, \exists x\, B_i, \ldots, B_n.}$$

By hypothesis, $\Gamma \longrightarrow B_1 \vee \ldots \vee [t/x]B_i \vee \ldots \vee B_n$ has an **I**-proof. Now, it is easily seen that

$$B_1 \vee \ldots \vee [t/x]B_i \vee \ldots \vee B_n \longrightarrow B_1 \vee \ldots \vee \exists x\, B_i \vee \ldots \vee B_n$$

has an **I**-proof. The desired conclusion follows from these observations by the admissibility of the *Cut* rule. The other cases have similar arguments.

Theorem 2 *Let Γ be a multiset of D-formulas and let G be a G-formula. Then $\Gamma \longrightarrow G$ has a **C**-proof if and only if it has an **I**-proof.*

Proof. The if direction is obvious. For the only if direction, we observe that there can be no occurrences of $\forall$-R and $\supset$-R in a cut-free **C**-proof of $\Gamma \longrightarrow G$. The desired conclusion now follows by virtue of Lemma 1.

□

The truth of Lemma 1 is independent of the appearance of the $\bot$-R rule in **C**-proofs and so Theorem 2 remains true even if our G- and D-formulas are extended by letting A in the syntax rules defining them be $\bot$ in addition to atomic formulas. Incorporating this extension permits D-formulas to also encode negative clauses: the clause $(\neg B_1 \vee \ldots \vee \neg B_m)$ can be written equivalently as $(B_1 \wedge \ldots \wedge B_m) \supset \bot$. Now, the (classical) unsatisfiability of a set of clauses Γ is equivalent to the existence of a **C**-proof for $\Gamma \longrightarrow \bot$. Thus, Theorem 2 shows that the reducibility of an arbitrary formula to clausal form provides another encoding of classical logic in intuitionistic logic.

4 Uniform provability

The language described in the previous section does not have the uniform proof property. For instance, consider the sequent

$$p(a) \vee p(b) \longrightarrow \exists x\, p(x).$$

This sequent has both a **C**-proof and an **I**-proof but it does not have a uniform proof. Notice that $p(a) \vee p(b)$ is a program clause of our language and $\exists x\, p(x)$ is a query.

Although the uniform proof property does not hold directly of our language, it holds of it in a derivative sense. In particular, suppose that Γ is a (multi)set of program clauses and that G is a query. Then $\Gamma \longrightarrow G$ has a **C**- or an **I**-proof if and only if $G \supset \bot, \Gamma \longrightarrow G$ has a uniform proof. This surprising observation has an interesting consequence: it can be used to describe a proof procedure for our language that is goal-directed and that differs from the procedure usually employed with Horn clauses only in the structure of the backchaining rule. We explore this aspect of our observation in the next section. The rest of this section is devoted to showing that the observation is, in fact, true.

We first note that the mentioned transformation of the assumption set is sound and complete with respect to intuitionistic logic.

Lemma 3 *Let Γ be a multiset of program clauses and let G be a query. Then $G \supset \bot, \Gamma \vdash_I G$ if and only if $\Gamma \vdash_I G$.*

Proof. The if direction is obvious. For the only if direction, we observe first that if $G \supset \bot, \Gamma \vdash_I G$ then $G \supset \bot, \Gamma \vdash_C G$ and, hence, $(G \supset \bot) \vee G, \Gamma \vdash_C G$. Noting that $\vdash_C (G \supset \bot) \vee G$ and using the *Cut* rule, we see that $\Gamma \vdash_C G$. The desired observation now follows from Theorem 2.

□

Theorem 4 *Let Γ be a multiset of program clauses and let G be a query. Then $G \supset \bot, \Gamma \vdash_I G$ if and only if $G \supset \bot, \Gamma \vdash_O G$.*

Proof. The if direction is immediate. We provide only a sketch of the argument for the only if direction.

Suppose that $G \supset \bot, \Gamma \longrightarrow G$ has a cut-free **I**-proof. Then it must have one in which (a) for every non-atomic formula appearing in the right of a sequent there is a rule introducing its top-level symbol and (b) no left rule immediately succeeds a right rule pertaining to a top-level logical symbol of a formula in a common sequent except in the following cases: the left rule is $\vee$-L and the right rule is either $\exists$-L or $\vee$-R or the left rule is $\exists$-L and the right rule is $\exists$-R. (To ensure (a) it may be necessary to introduce some inference steps right after $\bot$-R rules and (b) follows from known permutabilities for intuitionistic calculus applied to the kinds of sequents that arise in our situation.) Let us call a **I**-proof satisfying the requirements mentioned above an $\mathbf{I}'$-proof. Now, let $\mathcal{R}$ be either an $\vee$-L or an $\exists$-L rule. We associate a measure with $\mathcal{R}$ that is the count of the number of connectives and quantifiers that appear in the succedent of the lower sequent of $\mathcal{R}$. We use this measure to associate one with an $\mathbf{I}'$-proof Ξ as follows: the measure of Ξ is the sum of the measures of all the $\vee$-L and $\exists$-L rules that appear in Ξ. We claim that any $\mathbf{I}'$-proof of $G \supset \bot, \Gamma \longrightarrow G$ that has a nonzero such measure can be transformed into an $\mathbf{I}'$-proof of smaller measure. It follows easily from this that $G \supset \bot, \Gamma \longrightarrow G$ has a uniform proof.

To show the claim we consider the various cases by which an $\mathbf{I}'$-proof of $G \supset \bot, \Gamma \longrightarrow G$ might have nonzero measure. One possibility is that there is an $\vee$-L rule right after an $\vee$-R. For instance, the following steps might appear somewhere in the derivation:

$$\dfrac{\dfrac{B, \Gamma' \longrightarrow D}{B, \Gamma' \longrightarrow D \vee E}\ \vee\text{-R} \qquad C, \Gamma' \longrightarrow D \vee E}{B \vee C, \Gamma' \longrightarrow D \vee E}\ \vee\text{-L}$$

We shall assume that D and E are atomic — this assumption can be dispensed with in a detailed proof. Our objective in this case is to show that the use of $\vee$-L and $\vee$-R rules above can be reordered, thereby reducing the overall measure.

Using the fact that what is displayed above is a subpart of an $\mathbf{I}'$-proof of $G \supset \bot, \Gamma \longrightarrow G$, it can be shown that $C, \Gamma' \longrightarrow G$ has an $\mathbf{I}'$-proof of smaller measure than the one for $G \supset \bot, \Gamma \longrightarrow G$ — the essential idea is to mimic the $\mathbf{I}'$-proof for $G \supset \bot, \Gamma \longrightarrow G$ and to note that at least one use of the $\vee$-L rule — the one shown above — is not needed. By induction it follows then that $C, \Gamma' \longrightarrow G$ has an $\mathbf{I}'$-proof of zero measure. It can also be seen that C, Γ' can be written as either $G \supset \bot, \Gamma''$ or $\bot, \Gamma''$. We assume the former; the argument is simpler if the latter is true. Thus, we can construct the following sub-derivation:

$$\dfrac{\dfrac{B, \Gamma' \longrightarrow D \qquad \dfrac{C, \Gamma' \longrightarrow G \qquad \dfrac{\bot, \Gamma'' \longrightarrow \bot}{\bot, \Gamma'' \longrightarrow D}\ \bot\text{-R}}{C, \Gamma' \longrightarrow D}\ \supset\text{-L}}{B \vee C, \Gamma' \longrightarrow D}\ \vee\text{-L}}{B \vee C, \Gamma' \longrightarrow D \vee E}\ \vee\text{-R}$$

Using the known $\mathbf{I}'$-proofs of $B, \Gamma' \longrightarrow D$ and $C, \Gamma' \longrightarrow G$ together with this to replace the earlier subderivation,we obtain the desired $\mathbf{I}'$-proof of reduced measure.

The other cases are amenable to a similar argument.

□

There is a constructive content to the discussions in this section that we attempt to bring out by means of an example. Consider the following $\mathbf{I}$-proof of $p(a) \vee p(b) \longrightarrow \exists x\, p(x)$:

$$\dfrac{\dfrac{p(a) \longrightarrow p(a)}{p(a) \longrightarrow \exists x\, p(x)}\ \exists\text{-R} \qquad \dfrac{p(b) \longrightarrow p(b)}{p(b) \longrightarrow \exists x\, p(x)}\ \exists\text{-R}}{p(a) \vee p(b) \longrightarrow \exists x\, p(x)}\ \vee\text{-L}$$

By following the argument in the proof of Theorem 4, we obtain from this $\mathbf{I}$-proof the uniform proof for $\exists x\, p(x) \supset \bot, p(a) \vee p(b) \longrightarrow \exists x\, p(x)$ that is shown in Figure 2.

5 A deductive calculus with directionality

A virtue of the uniform proof property is that it provides some control over the proof search process: So long as the formula in the succedent of the sequent is non-atomic, the top-level connective can be used to direct the next step in the search. However, the way to proceed is not as clear in the situation when the succedent is an atomic formula since a variety of left rules may be possible. An analogous problem arises in the case of Horn clauses. The solution in that situation is to reduce the left rules to a single backchaining rule; of course, some nondeterminism persists in that a choice has to be made of the program clause that is to be backchained over. The surprising observation is that there is a natural generalization of this backchaining rule that can be used to similar effect even in the context of our language. We show this to be the case in this section.

To permit the statement of this backchaining rule, it is necessary, first of all, to consider a simplification to the syntax of program clauses. In particular, we assume henceforth that these are closed D-formulas given by the syntax rule

$$D \ ::= \ A \vee \ldots \vee A \mid G \supset (A \vee \ldots \vee A) \mid \forall x\, D;$$

as before, A is intended to be a syntactic variable for an atomic formula distinct from $\bot$ in this rule. A program clause in the syntax of Section 3 can be transformed into a classically equivalent set of program clauses in the current syntax. By invoking Theorem 2, we see that this transformation is also intuitionistically acceptable. Notice that Theorem 2 is used in an essential way in justifying this step since operations like (static) Skolemization and the raising of scope of universal quantifiers that are used in the transformation process are, in general, *not* intuitionistically valid. As a specific example, $\forall x\, (D_1(x) \vee D_2)$ is not necessarily intuitionistically equivalent to $(\forall x\, D_1(x)) \vee D_2$ even if x has no free occurrences in D_2.

$$
\dfrac{p(a), \exists x\, p(x) \supset \bot \longrightarrow p(a) \qquad \dfrac{\dfrac{p(b), \exists x\, p(x) \supset \bot \longrightarrow p(b)}{p(b), \exists x\, p(x) \supset \bot \longrightarrow \exists x\, p(x)}\,\exists\text{-R} \qquad \dfrac{\bot, p(b) \longrightarrow \bot}{\bot, p(b) \longrightarrow p(a)}\,\bot\text{-R}}{p(b), \exists x\, p(x) \supset \bot \longrightarrow p(a)}\,\supset\text{-L}}{\dfrac{p(a) \vee p(b), \exists x\, p(x) \supset \bot \longrightarrow p(a)}{\exists x\, p(x) \supset \bot, p(a) \vee p(b) \longrightarrow \exists x\, p(x)}\,\exists\text{-R}}\,\vee\text{-L}
$$

Figure 2: A Uniform Proof for $\exists x\, p(x) \supset \bot, p(a) \vee p(b) \longrightarrow \exists x\, p(x)$

If the syntax of our program clauses is simplified in this manner, then the rules $\exists$-L and $\wedge$-L can be excluded from our calculus since they cannot occur in cut-free proofs of sequents of the kind we are interested in. The contr-L rule can also be eliminated if we restate the $\forall$-L rule as follows:

$$\frac{[t/x]B, \forall x\, B, \Gamma \longrightarrow \Delta}{\forall x\, B, \Gamma \longrightarrow \Delta}$$

(We are effectively building contraction into this rule.) We hereafter assume this modification to the $\forall$-L rule.

Let Γ be a set of program clauses and let G be a query. By virtue of Theorem 4, a sound and complete strategy for finding a **C**- or an **I**-proof for $\Gamma \longrightarrow G$ is to look for a uniform proof for $G \supset \bot, \Gamma \longrightarrow G$. An analysis of the proof of the theorem indicates that the enhancement to the assumption set is unnecessary if we change the $\vee$-L rule to be the following:

$$\frac{B, \Gamma \longrightarrow A \qquad D, \Gamma \longrightarrow G}{B \vee D, \Gamma \longrightarrow A}$$

In this rule, A stands for an atomic formula and G for the original query. Notice that this is a rule that is dependent on the original query and we indicate this dependence by using the name $\vee$-L_G for it. The discussions that follow must also be understood relative to this choice of G.

By incorporating the various modifications described above, we obtain a calculus in which the only left rules are $\vee$-L_G, $\supset$-L and $\forall$-L. Further, assuming that Γ is a (multi)set of program clauses, the classical and intuitionistic provability of $\Gamma \longrightarrow G$ is equivalent to the uniform provability of the same sequent in this modified calculus. The backchaining rule that is of interest to us combines the three left rules of our modified calculus into a single rule.

Definition 1. Let D be a program clause. Then $[D]$ denotes a collection of pairs of sets of formulas given as follows:

(1) If D is $A_1 \vee \ldots \vee A_n$, then

$$[D] = \{\langle \emptyset, \{A_1, \ldots A_n\}\rangle\}.$$

(2) If D is $G \supset (A_1 \vee \ldots \vee A_n)$, then

$$[D] = \{\langle \{G\}, \{A_1, \ldots, A_n\}\rangle\}.$$

(3) If D is $\forall x\, D_1$, then

$$[D] = \bigcup\{[[t/x]D_1] \mid t \text{ is a term}\}.$$

This notation is extended to a (multi)set Γ of program clauses as follows: $[\Gamma] = \bigcup\{[D] \mid D \in \Gamma\}$.

Definition 2. Let Γ be a (multi)set of program clauses. As mentioned already, G is assumed to be a fixed query.

(1) The ATOMIC rule is the following

$$\frac{A_1, \Gamma \longrightarrow G \qquad \ldots \qquad A_n, \Gamma \longrightarrow G}{\Gamma \longrightarrow C}$$

provided that $\langle \emptyset, \{C, A_1, \ldots, A_n\}\rangle \in [\Gamma]$. In the degenerate case, *i.e.*, when the second component of the pair shown is simply $\{C\}$, this rule has no upper sequents.

(2) The BACKCHAIN rule is the following

$$\frac{\Gamma \longrightarrow G' \quad A_1, \Gamma \longrightarrow G \quad \ldots \quad A_n, \Gamma \longrightarrow G}{\Gamma \longrightarrow C}$$

provided that $\langle \{G'\}, \{C, A_1, \ldots, A_n\}\rangle \in [\Gamma]$. In the degenerate case, *i.e.*, when the second component of the pair shown is simply $\{C\}$, this rule has $\Gamma \longrightarrow G'$ as its only upper sequent.

The rules defined above are obvious generalizations of the ones that make use of program clauses in the usual Horn clause setting. The latter rules correspond, in fact, to the degenerate versions of our rules. From a proof search perspective, both the ATOMIC and the BACKCHAIN rules exhibit directionality: the atomic formula to be proved determines the clauses whose use is to be considered.

Lemma 5 *Let Γ be a multiset of program clauses and let C be an atomic formula. Then $\Gamma \longrightarrow C$ has a uniform proof with l sequents in the calculus with the original right rules and the left rules $\vee$-L_G, $\supset$-L and $\forall$-L, if and only if one of the following conditions hold:*

(1) For some $A_1, \ldots, A_n$, $\langle \emptyset, \{C, A_1, \ldots, A_n\}\rangle \in [\Gamma]$ and, for $1 \leq i \leq n$, $A_i, \Gamma \longrightarrow G$ has a uniform proof with fewer than l sequents in the same calculus.

(2) For some $A_1, \ldots, A_n$, $\langle \{G'\}, \{C, A_1, \ldots, A_n\}\rangle \in [\Gamma]$ and $\Gamma \longrightarrow G'$ and, for $1 \leq i \leq n$, $A_i, \Gamma \longrightarrow G$ have uniform proofs with fewer than l sequents in the same calculus.

Proof. By an induction on the size of the uniform proof for $\Gamma \longrightarrow C$.

□

Theorem 6 *Let Γ be a set of program clauses and let G be a query. Then $\Gamma \longrightarrow G$ has a **C**-proof or an **I**-proof if and only if it has a uniform proof in a calculus containing all the right rules and the ATOMIC and BACKCHAIN rule relativized to G.*

Proof. Recall that a sequent of the kind we are interested in has a **C**-proof or an **I**-proof just in case it has a uniform proof in a calculus containing all the right rules, $\vee$-L$_G$, $\supset$-Land $\forall$-L. The theorem follows by an easy induction from this and Lemma 5.

□

The final calculus described in this section can translate into a proof procedure very close to one already in the literature in the Disjunctive Logic Programming field, namely, the Inheritance Near-Horn Prolog procedure (InH-Prolog)[6, 12]. The ATOMIC and BACKCHAIN rules implement precisely the restart rule of InH-Prolog, and with the superimposed search ordering of InH-Prolog providing an ordering of pursuit of the upper sequents of these rules, the mapping from proof searches in this calculus to InH-Prolog proof search is nearly one-to-one. This relationship will be made explicit in the full paper. Finally, a very similar calculus can also be described in the situation where the atomic formula in the syntax rules for program clauses and queries is permitted to be $\perp$. Indeed, this is the usual presentation format for InH-Prolog, although it is known that use of the original goal is interchangeable with use of $\perp$ for all procedures in the Near-Horn Prolog family. Again, this will be addressed in the full version of this paper.

6 Conclusion

In this paper, we have investigated a generalization of the logical language underlying disjunctive logic programming from a proof-theoretic perspective. While the uniform proof property described in [8] does not hold of this language, we have shown that a modest addition to programs makes the search for a uniform proof a complete strategy. This observation has been utilized to describe a deductive calculus that captures the essence of the proof structure of Inheritance near-Horn Prolog, one of the proof procedures earlier proposed for disjunctive logic programming. This methodology also provides a basis for analyzing other proposed proof procedures. A fuller discussion of these issues will appear in a complete version of the paper.

References

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

[2] J. Lobo, J. Minker, and A. Rajasekar. Extending the semantics of logic programs to disjunctive logic programs. In G. Levi and M. Martelli, editors, *Logic Programming: Proc. of the Sixth Int'l Conf.*, Lisbon, Portugal, 1989.

[3] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, Cambridge, MA, 1992.

[4] D.W. Loveland. Near-Horn Prolog. In J. Lassez, editor, *Logic Programming: Proc. of the Fourth Int'l Conf.*, pages 456–469. MIT Press, 1987.

[5] D.W. Loveland. Near-Horn Prolog and beyond. *J. Automated Reasoning*, 7:1–26, 1991.

[6] D.W. Loveland and D.W. Reed. A near-Horn Prolog for compilation. In J. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson.* MIT Press, 1991.

[7] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.

[8] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[9] J. Minker. On indefinite databases and the closed world assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, Berlin, 1982.

[10] Gopalan Nadathur. A proof procedure for the logic of hereditary Harrop formulas. *Journal of Automated Reasoning*, 11(1):115–145, August 1993.

[11] T.C. Pryzmusinski. Stationary semantics for disjunctive logic programs and deductive databases. In S. Debray and M. Hermenegildo, editors, *Logic Programming: Proc. of the 1990 North American Conf.*, pages 40–62. MIT Press, 1990.

[12] D.W. Reed, D.W. Loveland, and B.T. Smith. An alternative characterization of disjunctive logic programs. In V. Saraswat and K. Ueda, editors, *Logic Programming: Proc. of the 1991 Int'l Symp.* MIT Press, 1991.

[13] K.A. Ross and R.W. Topor. Inferring negative information from disjunctive databases. *J. Automated Reasoning*, 4(2):397–424, 1988.

[14] A. Yahya and L.J. Henschen. Deduction in non-Horn databases. *J. Automated Reasoning*, 1(2):141–160, 1985.