Goal-Directed Methods for Fuzzy Logics

GEORGE METCALFE AND NICOLA OLIVETTI

In this contribution we present uniform goal-directed rules for the implicational fragments of the three main formalizations of fuzzy logic; namely, Łukasiewicz logic **Ł**, Gödel logic $\mathbf{G}$, and Product logic $\mathbf{\Pi}$. We begin with a historical overview of the goal-directed methodology, focussing in particular on the pioneering work of Dov Gabbay, then proceed by recalling the fundamental systems of fuzzy logic, and developing corresponding goal-directed algorithms.

1 Historical Overview

1.1 N-Prolog

The main ideas underlying the goal-directed methodology were put forward in the early 1980s by Gabbay and developed further by a number of researchers. Recognizing that the deductive mechanism of Prolog could be generalized to support more sophisticated forms of reasoning, Gabbay (together with Reyle) [Gabbay and Reyle, 1984; Gabbay, 1985] proposed an extension of the Horn clause language, called N-Prolog, capable of performing *hypothetical reasoning*. In N-Prolog, implicational goals of the form $D \to G$ are permitted that succeed from a program P iff the goal G succeeds from the program $P \cup \{D\}$, e.g. the goal

$$pass(john, course123) \to can\_graduate(john)$$

with intuitive meaning "if John passes course123, can he graduate?" succeeds from P iff we can conclude that John can graduate from P expanded with the fact that he has passed course123. Note that here the deduction theorem is used to define the *meaning* of an implicational goal, and indeed that the evaluation of such goals may involve a change of the program; in this case, a simple addition of data to the program but in general, a possibly more sophisticated *update* of the program. Hypothetical goals may occur also in the body of a clause, as in for example, Gabbay's formalization of the law for British citizenship:

$$\begin{array}{rcl} born\_in\_UK(X) \wedge father(X,Y) & & \\ \wedge\ (alive(Y,T) \to british\_citizen(Y,T)) & \to & british\_citizen(X,T) \\ british\_citizen(X,T_1) \wedge (T_1 < T_2) \wedge alive(Z,T_1) & \to & british\_citizen(Z,T_2) \\ dead(X,T) \wedge alive(X,T) & \to & \bot \end{array}$$

The first clause establishes the counterfactual claim that X is a British citizen at time T if X was born in the UK and has a father Y who if alive at time T would be a British citizen. The second clause claims that being a British citizen persists over time, while the third expresses the incompatibility of $dead$ and $alive$.

The addition of hypothetical goals as described above is not simply a procedural trick, but rather has a well-understood logical basis; the goals that succeed from a program P in N-Prolog being exactly the logical consequences of P in *Intuitionistic logic*. Related work includes Hudelmaier's optimized version of N-Prolog [Hudelmaier, 1990], a comparison of various Prolog extensions by Reed and Loveland [Reed and Loveland, 1992], and a similar hypothetical extension proposed by McCarty [McCarty, 1988] in the context of deductive databases (see also [Bonner, 1990]). Also, further hypothetical extensions of logic programming were proposed by Gabbay and others [Gabbay *et al.*, 2000] in order to define a conditional extension of logic programming incorporating a revision mechanism.[1]

1.2 Algorithmic proofs

Some years later Gabbay [Gabbay, 1992] proposed a procedural *interpretation* of logics. Whereas the traditional view of logics is extensional, i.e. a logic is a set of theorems (valid formulas) in some language, Gabbay suggested identifying a logical system with a pair (SetOfTheorems, Procedure), where Procedure is a correct deduction method for SetOfTheorems, that is to say, it generates all and only the members of SetOfTheorems. According to this perspective, (ClassicalLogic, TruthTables) is for instance a different logical system from (ClassicalLogic, Tableaux) or (ClassicalLogic, Resolution).

This conceptual shift is motivated by a number of considerations:

1. A new geography of logical systems is created, providing an alternative perspective on the differences and similarities between logics (in the traditional sense). For instance Classical logic with truth tables may be viewed as being very close to finite many-valued logic with truth tables, and Classical logic with sequent calculus as being very close to Intuitionistic logic with sequent calculus, whereas Intuitionistic logic and (finite) many-valued logic do not seem to be closely related.

2. The identification of a logical system with a pair (SetOfTheorems, Procedure) allows the possibility of discovering (or re-discovering) logics (in the traditional sense) by changing the deduction procedure, e.g. by making it stronger or weaker.

3. Other reasoning mechanisms may be defined on top of the deduction procedure, prominent examples being negation by (finite) failure and abduction.

[1]A revision mechanism is needed to preserve the consistency of a database containing integrity constraints since they may be violated by the (hypothetical) insertion of new data.

Note, moreover, that while two deduction procedures for some logic may coincide on derivable formulas, they may differ in finite-failure and/or lead to the computation of different solutions for abduction.

1.3 The goal-directed methodology

Goal-directed methods fall under the procedural perspective mentioned above; being an extension of the standard computation mechanism of logic programming. Denoting by $\Gamma \vdash^? A$, the query "does A follow from Γ?" where Γ is a database (collection) of formulas and A is a goal formula, a deduction method for queries is *goal-directed* in the sense that each step in a proof is determined by the form of the current goal. More precisely, a complex goal is decomposed until its atomic constituents are reached, while an atomic goal q is matched (if possible) with the "head" of a formula $G \to q$ in the database, and its "body" G asked in turn. This latter step may be viewed as a sort of resolution or backchaining step. We illustrate this process here with a simple example of propositional N-Prolog for Intuitionistic logic with only $\to$ and $\wedge$; goal-formulas (G-formulas) and database formulas (D-formulas) being defined as follows:

$$D = q \mid G \to q \mid D \wedge D$$
$$G = q \mid D \to G \mid G \wedge G$$

A database Γ is a set of D-formulas $\{D_1, \ldots, D_n\}$, every formula in the fragment being equivalent both to a database (interpreted as a conjunction) and to a G-formula. The following goal-directed deduction procedure is sound and complete for this fragment:

$(success)$ $\Gamma \vdash^? q$ if $q \in \Gamma$

$(implication)$ From $\Gamma \vdash^? D \to G$ step to $\Gamma, D \vdash^? G$

(and) From $\Gamma \vdash^? G_1 \wedge G_2$ step to $\Gamma \vdash^? G_1$ and $\Gamma \vdash^? G_2$

$(reduction)$ From $\Gamma \vdash^? q$ step to $\Gamma \vdash^? G$ if $G \to q \in \Gamma$

The goal-directed model of deduction may be refined in several ways in order to capture (efficiently) a large variety of logical systems:

1. The database can be structured, e.g. as a multiset, a list, or some more complicated structure, or constrained, e.g. by allowing only D-formulas in a certain position to be matched with the current atomic goal in the $(reduction)$ step. More generally, databases may be structured according to Gabbay's theory of *Labelled Deductive Systems* (LDS) [Gabbay, 1996; Gabbay and Olivetti, 2002a], where a labelled goal is asked from a set of labelled data and the goal-directed rules impose constraints on label propagation and combination.

2. Goal-directed algorithms can be modified to ensure termination, either by loop-checking or by "diminishing resources" i.e. removing formulas "used" to match an atomic goal. Note that in the latter case, the deletion of used data should usually be compensated for in some way to maintain completeness.

3. Goals (possibly also states of the database) previously occurring in a deduction may be stored in a history, and re-asked (or re-used in some way) using *restart rules*.

For example, a (terminating) proof system for Classical logic is obtained by modifying the calculus above to allow the replacement of the current goal by any other goal previously occurring in the same derivation branch. To this end, queries are reformulated as structures of the form $\Gamma \vdash^? G; H$ where H is a set of (atomic) goals called the *history*. We revise the reduction rule as follows:

$$(reduction) \quad \text{From } \Gamma, G \to q \vdash^? q; H \text{ step to } \Gamma \vdash^? G; H \cup \{q\}$$

and add a rule allowing restarts from any goal in the history:

$$(restart) \quad \text{From } \Gamma \vdash^? q; H \text{ step to } \Gamma \vdash^? p; H \cup \{q\} \text{ if } p \in H$$

For instance Peirce's axiom (which is not valid in Intuitionistic logic) is derivable as follows:

$$\begin{array}{rlll} & \vdash^? & ((p \to q) \to p) \to p; \emptyset & (implication) \\ (p \to q) \to p & \vdash^? & p; \emptyset & (reduction) \\ & \vdash^? & p \to q; \{p\} & (implication) \\ p & \vdash^? & q; \{p\} & (restart) \\ p & \vdash^? & p; \{p, q\} & (success) \end{array}$$

Goal-directed presentations of a wide variety of logics have been developed in Gabbay and Olivetti's book [Gabbay and Olivetti, 2000], and also by several other authors, e.g. [Giordano *et al.*, 1992; Bollen, 1991; Harland, 1997]. Moreover it has been shown that goal-directed proof procedures are useful to obtain theoretical results on logics such as interpolation [Gabbay and Olivetti, 2002b].

1.4 Uniform proofs

The goal-directed approach is closely related to the Uniform Proof paradigm proposed by Miller and others at the beginning of 1990s, see e.g. [Miller *et al.*, 1991; Harland and Pym, 1991; Hodas and Miller, 1994], and used as the basis for logic programming in various non-classical logics. The methodology and underlying perspective is different, however. The starting point for developing a uniform proof calculus is an analysis of a sequent calculus for the relevant logic; a uniform proof of a sequent $\Gamma \vdash A$ being a sequent calculus proof where (reading upwards) the goal A is decomposed first and the rules are applied to formulas in Γ only when A is atomic. In this respect the connectives occurring in the goal A may be viewed as "instructions" for directing proof search. A uniform proof calculus for a logic is then obtained from an analysis of the *permutability* of the rules of the calculus. This analysis allows the identification of fragments of the logic, such that if a sequent expressed in this fragment has a proof, then it has a uniform proof. The approach has been applied most successfully to logics (or fragments of logics) having a single-conclusion sequent calculus, although the treatment of multiple-conclusion calculi is also possible [Nadathur, 1998].

2 t-Norm Based Fuzzy Logics

Fuzzy logics are many-valued logics that form a suitable basis for reasoning under *vagueness*, providing the core of systems formalising approximate reasoning in (the field of) Fuzzy Logic. Such logics may be defined in a number of ways. Here, we focus on the influential "t-norm based" approach of Hájek [Hájek, 1998], which makes the following two basic assumptions or "design choices":

1. The set of truth values for the logic is the *real unit interval* $[0, 1]$.
2. The logic is *truth-functional*, i.e. the truth value of a compound formula is a function of the truth values of its subformulas.

Hájek also proposes restrictions on truth functions interpreting conjunction, thereby arriving at the well-known class of *continuous* t*-norms*: that is; continuous binary functions $* : [0, 1]^2 \to [0, 1]$ such that for all $x, y, z \in [0, 1]$:

1. $x * y = y * x$ (Commutativity)
2. $(x * y) * z = x * (y * z)$ (Associativity)
3. $x \leq y$ implies $x * z \leq y * z$ (Monotonicity)
4. $1 * x = x$ (Identity)

Suitable functions for interpreting *implication* (satisfying e.g. a generalized modus ponens property) are then the *residua* of continuous t-norms, defined as functions $\Rightarrow_*$: $[0, 1]^2 \to [0, 1]$ such that for all $x, y \in [0, 1]$:

$$x \Rightarrow_* y =_{def} max\{z \ : \ x * z \leq y\}$$

The most important examples of continuous t-norms and their residua are:

	t-Norm	Residuum
Łukasiewicz	$x{*_{\mathbf{Ł}}}y = max(0, x+y-1)$	$x{\Rightarrow_{\mathbf{Ł}}}y = min(1, 1-x+y)$
Gödel	$x *_{\mathbf{G}} y = min(x, y)$	$x \Rightarrow_{\mathbf{G}} y = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases}$
Product	$x *_{\mathbf{\Pi}} y = x \cdot y$	$x \Rightarrow_{\mathbf{\Pi}} y = \begin{cases} 1 & \text{if } x \leq y \\ \frac{y}{x} & \text{otherwise} \end{cases}$

Apart from the historical and practical importance of the Łukasiewicz and Gödel t-norms and their associated logics, there is a further reason to consider the above t-norms fundamental: namely, that any continuous t-norm is an ordinal sum construction of these three, see e.g. [Hájek, 1998] for details.

Following Hájek, it can now be seen that each continuous t-norm determines a *propositional fuzzy logic* $\mathbf{L}_*$ based on a language with binary connectives $\odot$ and $\rightarrow$, and a constant $\bot$; *valuations* for $\mathbf{L}_*$ being functions v assigning to each propositional variable a truth value from the real unit interval $[0, 1]$, uniquely extended to formulas by:

$$\begin{array}{rcl} v(A \odot B) & = & v(A) * v(B) \\ v(A \rightarrow B) & = & v(A) \Rightarrow_* v(B) \\ v(\bot) & = & 0 \end{array}$$

A is *valid* for $\mathbf{L}_*$, written $\models_{\mathbf{L}_*} A$, if $v(A) = 1$ for all valuations v for $\mathbf{L}_*$.

Note that for any continuous t-norm $*$, the min and max functions expressing the lattice properties of $[0, 1]$ can be defined using just $*$ and $\Rightarrow_*$, i.e. for all $x, y \in [0, 1]$:

1. $min(x, y) = x * (x \Rightarrow_* y)$.
2. $max(x, y) = min((x \Rightarrow_* y) \Rightarrow_* y, (y \Rightarrow_* x) \Rightarrow_* x)$.

Moreover, the function $x \Rightarrow_* 0$ gives suitable properties for interpreting *negation*, being e.g. anti-monotonic with $0 \Rightarrow_* 0 = 1$ and $1 \Rightarrow_* 0 = 0$. Accordingly, the following connectives are also defined:

- $A \wedge B =_{def} A \odot (A \rightarrow B)$.
- $A \vee B =_{def} ((A \rightarrow B) \rightarrow B) \wedge ((B \rightarrow A) \rightarrow A)$.
- $\neg A =_{def} A \rightarrow \bot$.

In [Hájek, 1998] Hájek gives the following axiomatization for a *Basic fuzzy logic* $\mathbf{BL}$ that he conjectures (proved in [Cignoli *et al.*, 2000]) axiomatizes the formulas valid in *all* logics based on continuous t-norms:

$$\begin{array}{rl}
\text{(A1)} & (A \to B) \to ((B \to C) \to (A \to C)) \\
\text{(A2)} & (A \odot B) \to A \\
\text{(A3)} & (A \odot B) \to (B \odot A) \\
\text{(A4)} & (A \odot (A \to B)) \to (B \odot (B \to A)) \\
\text{(A5a)} & (A \to (B \to C)) \to ((A \odot B) \to C) \\
\text{(A5b)} & ((A \odot B) \to C) \to (A \to (B \to C)) \\
\text{(A6)} & ((A \to B) \to C) \to (((B \to A) \to C) \to C) \\
\text{(A7)} & \bot \to A
\end{array}$$

$$\frac{A \quad A \to B}{B}\ (mp)$$

Axiomatizations for the three fundamental logics are obtained as follows:

- *Łukasiewicz logic* **Ł** is $\mathbf{BL}$ plus (INV) $\neg\neg A \to A$.
- *Gödel logic* $\mathbf{G}$ is $\mathbf{BL}$ plus (ID) $A \to (A \odot A)$.
- *Product logic* $\mathbf{\Pi}$ is $\mathbf{BL}$ plus (Π) $\neg\neg A \to ((A \to (A \odot B)) \to B)$ and (S) $\neg(A \wedge \neg A)$.

For **Ł**, we remark that an alternative axiomatization can be given based on a language with connectives $\to$ and $\bot$, i.e. (mp) with:

$$\begin{array}{rl}
(\text{Ł}1) & A \to (B \to A) \\
(\text{Ł}2) & (A \to B) \to ((B \to C) \to (A \to C)) \\
(\text{Ł}3) & ((A \to B) \to B) \to ((B \to A) \to A) \\
(\text{Ł}4) & ((A \to \bot) \to (B \to \bot)) \to (B \to A)
\end{array}$$

The t-norm approach described above has been generalized in a number of directions. In particular, since a sufficient and necessary condition for a t-norm to have a residuum, is that it be *left-continuous*, a logic called *Monoidal t-norm logic* has been introduced in [Esteva and Godo, 2001] that captures exactly the tautologies of all left-continuous t-norm logics [Jenei and Montagna, 2002].

3 Uniform Goal-Directed Methods for Fuzzy Logics

A variety of proof methods have been defined for fuzzy logics. In particular, calculi for many of the most important t-norm based logics have been presented in the framework of *hypersequents*, a generalization of sequents to multisets of sequents, introduced independently by Avron [Avron, 1987] and Pottinger [Pottinger, 1983]. The first calculus of this type was defined for $\mathbf{G}$ by Avron in [Avron, 1991] and consists of the same “standard” rules for connectives as for e.g. Intuitionistic logic, together with further structural rules characterizing the linearity of truth values for the logic. More recently, hypersequent calculi with "non-standard" rules, have been defined by the current authors and Gabbay for **Ł** [Metcalfe *et al.*, 2005b] and $\mathbf{\Pi}$ [Metcalfe *et al.*, 2004a]. Moreover, taking such calculi as a starting point, goal-directed methods for $\mathbf{G}$ [Metcalfe *et al.*, 2003] and **Ł** [Metcalfe *et al.*, 2004b] have been defined that have a number of appealing properties, including the subformula property, termination, and optimal complexity, and provide a suitable basis for fuzzy logic programming [Metcalfe *et al.*, 2005a].

One unsatisfactory feature of the mentioned calculi, however, is the lack of uniformity in the rules. Essentially a new calculus has to be defined (and indeed implemented) for each logic. This contrasts for example with the situation for substructural logics where differences between logics consist solely in the presence or absence of structural rules in the calculus. This issue has been tackled recently for **Ł**, $\mathbf{G}$, and $\mathbf{\Pi}$ by Ciabattoni et al. in [Ciabattoni *et al.*, 2005] where calculi with uniform rules are defined in a framework of *relational hypersequents*: roughly speaking, a further generalization of hypersequents that allows two types of sequent to occur. In this section, we make use both of previous work on goal-directed calculi for $\mathbf{G}$ and **Ł**, and the new insights provided by the relational hypersequent approach, to give uniform goal-directed rules for all three fundamental fuzzy logics. Focussing for simplicity of presentation on the implicational fragments, we define uniform rules that are sound and invertible for **Ł**, $\mathbf{G}$, and $\mathbf{\Pi}$, then show how these can be used either as the basis for Co-NP decision procedures, or extended to give fully goal-directed algorithms.

3.1 Uniform implication rules

We begin with the notion of a goal-directed query for fuzzy logics, generalizing both the structures for Intuitionistic and Classical logic given above, and the definitions of previous work. Such queries consists of a database together with a *multiset* of goals (rather than just one), and a history of previous *states* of the database with goals (rather than just goals). As above, we also allow the possibility of *restarts*, limited however to at most one for each multiset of goals. More precisely, and noting that henceforth we assume all set notation to refer to multisets:

DEFINITION 1 (Goal-Directed Query). A *goal-directed query* (query for short) is a structure of the form:

$$\Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\} \text{ where}$$

- $\Gamma_1, \ldots, \Gamma_n$ are multisets of formulas called *databases*.
- $\Delta_1, \ldots, \Delta_n$ are multisets of formulas called *goals*.
- $R_1, \ldots, R_n$ are multisets of at most one atomic formula called *restarts*.

Intuitively, the meaning of a query is that for each valuation for the logic, there should be a state of the database where the associated goals "follow from" that database (possibly using restart), where "follow from" is particular to the logic.

DEFINITION 2 (Validity of Queries). Let Q be a query:

$$\Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

Q is *satisfied* for $\mathbf{S}$ by a valuation v if for some i, $1 \leq i \leq n$:

$$\text{either } \#^v_{\mathbf{S}}\Gamma_i \leq \#^v_{\mathbf{S}}\Delta_i \text{ or } \#^v_{\mathbf{S}}(\Gamma_i \cup R_i) < \#^v_{\mathbf{S}}\Delta_i$$

where $\#^v_{\mathbf{S}}\emptyset = 1$ for $\mathbf{S} \in \{\mathbf{Ł}, \mathbf{G}, \mathbf{\Pi}\}$ and:

- $\#^v_{\mathbf{Ł}}(\Gamma) = 1 + \sum_{A \in \Gamma}\{v(A) - 1\}$
- $\#^v_{\mathbf{G}}(\Gamma) = min_{A \in \Gamma}\{v(A)\}$
- $\#^v_{\mathbf{\Pi}}(\Gamma) = \prod_{A \in \Gamma}\{v(A)\}$

Q is *valid* for $\mathbf{S}$, written $\models_{\mathbf{S}} Q$, iff Q is satisfied by all valuations v for $\mathbf{S}$.

Observe that for $\mathbf{G}$ and $\mathbf{\Pi}$, the valuation $\#^v_{\mathbf{S}}$ of a multiset of formulas is defined using the relevant t-norm, min and $\cdot$, respectively. For $\mathbf{Ł}$, on the other hand, the valuation $\#^v_{\mathbf{Ł}}$ is defined using the "unbounded" part of the t-norm, i.e. instead of $min(1, x + y - 1)$, we use simply $x + y - 1$. Note moreover, that the restart formula gives a choice for each state: either the restart is absent from the database and the relation is "less than or equal to", or the restart is present, and the relation is "strictly less than". However we emphasize that despite the complicated interpretation, the crucial point is that a single formula A is valid for any of the three logics $\mathbf{S} \in \{\mathbf{Ł}, \mathbf{G}, \mathbf{\Pi}\}$ iff the query $\vdash^? A; \emptyset; \emptyset$ is valid for $\mathbf{S}$. Hence checking validity for queries includes checking validity for formulas as a special case.

We now define uniform rules for handling implication in these logics, writing $\{A_1, \ldots, A_n\} \to B$ as shorthand for $A_1 \to (A_2 \to \ldots (A_n \to B) \ldots)$:

DEFINITION 3 (Implication Rules).

$(implication)$ From $\Gamma \vdash^? \Pi \to q, \Delta; R; H$ step to
$\Gamma \vdash^? \Delta; R; H$ and $\Gamma, \Pi \vdash^? q, \Delta; R; H$

$(l\text{-}reduction)$ From $\Gamma, \Pi \to q \vdash^? q, \Delta; R; H$ step to
$\Gamma, q \vdash^? \Pi, q, \Delta; R; H \cup \{(\Gamma \vdash^? q, \Delta; R)\}$ and
$\vdash^? \Pi; \{q\}; H \cup \{(\Gamma \vdash^? q, \Delta; R)\}$

$(r\text{-}reduction)$ From $\Gamma \vdash^? q, \Delta; R_1; H \cup \{(\Gamma', \Pi \to q \vdash^? \Delta'; R_2)\}$ step to
$\Gamma', q \vdash^? \Pi, \Delta'; R_2; H \cup \{(\Gamma' \vdash^? \Delta'; R_2), (\Gamma \vdash^? q, \Delta; R_1)\}$ and
$\vdash^? \Pi; \{q\}; H \cup \{(\Gamma' \vdash^? \Delta'; R_2), (\Gamma \vdash^? q, \Delta; R_1)\}$

The implication rule $(implication)$ treats a query with an implicational goal $\Pi \to q$, and steps to *two* further queries: one where this goal is removed, and one where Π is added to the database and q replaces $\Pi \to q$ as a goal. However, note that if $\Pi \to q$ is the *only* goal, then a rule with one premise is sufficient, i.e.:

$(1\text{-}implication)$ From $\Gamma \vdash^? \Pi \to q; R; H$ step to $\Gamma, \Pi \vdash^? q; R; H$

The presence of an implicational formula in one of the states of the database is treated by two rules. *Local reduction* $(l\text{-}reduction)$ and *remote reduction* $(r\text{-}reduction)$ treat the cases where a goal matches the head of a formula in the current database, and in a database of a state in the history, respectively.

EXAMPLE 4. We illustrate these rules with a simple example:

$$\begin{array}{rll}
 & \vdash^? \{p, p \to q\} \to q; \emptyset; \emptyset & (1\text{-}implication) \\
p, p \to q & \vdash^? q; \emptyset; \emptyset & (l\text{-}reduction) \\
/ & \qquad \backslash & \\
p, q \;\vdash^? p, q; \emptyset; \{(p \vdash^? q; \emptyset)\} & \qquad \vdash^? p; \{q\}; \{(p \vdash^? q; \emptyset)\} &
\end{array}$$

We now check that the implication rules are sound (i.e. if the premises are valid, then the conclusion is valid) and invertible (i.e. if the conclusion is valid, then the premises are valid) for each logic.

LEMMA 5. *The implication rules are sound and invertible for* **Ł**, $\mathbf{G}$, *and* $\mathbf{\Pi}$.

Proof. Note first that we may assume the common part H of the histories of the premises and rules to be empty, since if H is satisfied for a valuation then clearly both the premises and conclusion are satisfied. Let v be a valuation for $\mathbf{S} \in \{\textbf{Ł}, \mathbf{G}, \mathbf{\Pi}\}$. We treat each rule in turn:

- $(implication)$. The cases of **Ł** and $\mathbf{\Pi}$ follow almost immediately by elementary arithmetic, so we just check the case of $\mathbf{G}$. As a first subcase, if $\#^v_{\mathbf{G}}\Pi \leq v(q)$, then $v(\Pi \to q) = 1$, and clearly the first premise is satisfied iff the conclusion is satisfied. Moreover, if the conclusion is satisfied, then either $\#^v_{\mathbf{G}}\Gamma \leq \#^v_{\mathbf{G}}\Delta$ or $min(\#^v_{\mathbf{G}}\Gamma, v(p)) < \#^v_{\mathbf{G}}\Delta$ where $R = \{p\}$. If the former, then clearly $min(\#^v_{\mathbf{G}}\Gamma, \#^v_{\mathbf{G}}\Pi) \leq min(\#^v_{\mathbf{G}}\Delta, v(q))$. If the latter and $\#^v_{\mathbf{G}}\Pi < v(q)$, then $min(\#^v_{\mathbf{G}}\Gamma, v(p), \#^v_{\mathbf{G}}\Pi) < min(v(q), \#^v_{\mathbf{G}}\Delta)$; otherwise $\#^v_{\mathbf{G}}\Pi = v(q)$ and $min(\#^v_{\mathbf{G}}\Gamma, \#^v_{\mathbf{G}}Pi) \leq min(v(q), \#^v_{\mathbf{G}}\Delta)$. Now for the second subcase, suppose that $\#^v_{\mathbf{G}}\Pi > v(q)$ and hence $v(\Pi \to q) = v(q)$. Clearly if the conclusion is satisfied, then both premises are satisfied. Also, if the first premise is satisfied, then it follows from the fact that $min(\#^v_{\mathbf{G}}\Pi, v(q)) = v(q)$ that the conclusion is also satisfied.

- $(l\text{-}reduction)$. First, assume that $\#^v_{\mathbf{S}}\Pi \leq v(q)$, and hence that $v(\Pi \to q) = 1$. Clearly, both premises are satisfied if the conclusion is satisfied. Moreover, the conclusion is satisfied, for $\#^v_{\mathbf{S}}\Pi = 1$ if the first premise is satisfied, and for $\#^v_{\mathbf{S}}\Pi < 1$, if the second premise is satisfied. Now assume that $\#^v_{\mathbf{S}}\Pi > v(q)$. Clearly, the second premise is satisfied. For **Ł** and **Π**, it follows by simple arithmetic that the conclusion is satisfied iff the first premise is satisfied. For **G**, $v(\Pi \to q) = v(q)$, and it follows that the first premise is satisfied iff the conclusion is satisfied.
- $(r\text{-}reduction)$. Very similar to the previous case. ■

However, to use these rules as a calculus for establishing the validity of formulas, we require a further rule allowing "switching" between the current database and states of the history.

DEFINITION 6 (Switch Rule).

$$(switch) \quad \text{From } \Gamma_1 \vdash^? \Delta_1; R_1; H \cup \{(\Gamma_2 \vdash^? \Delta_2; R_2)\} \text{ step to}$$
$$\Gamma_2 \vdash^? \Delta_2; R_2; H \cup \{(\Gamma_1 \vdash^? \Delta_1; R_1)\}$$

Observe that each implication rule reduces a formula occurring in the query into subformulas. Hence, assuming that a loop checking mechanism is used for $(switch)$ to stop it repeating ad infinitum, these rules terminate with queries where all goals are atomic and fail to match the head of any non-atomic database formula. We call such queries *irreducible*.

DEFINITION 7 (Irreducible Queries). Let Q be a query:

$$\Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

Q is *irreducible* iff:

1. Δ_i is atomic for $i = 1, \ldots, n$.
2. $\Gamma_i = \Pi_i \cup \Sigma_i$ for $i = 1, \ldots, n$ where Σ_i is atomic.
3. $Head(\Pi_1 \cup \ldots \cup \Pi_n) \cap (\Delta_1 \cup \ldots \cup \Delta_n) = \emptyset$.

where $Head(\Pi \to q) = q$ and $Head(\Gamma) = \{Head(A) \ : \ A \in \Gamma\}$.

PROPOSITION 8. *Applying the implication rules with $(switch)$ to queries (using loop-checking for applications of $(switch)$) terminates with irreducible queries.*

Proof. We define the following measures and well-orderings:

- $c(q) = 1$ for q atomic.

- $c(A \to B) = c(A) + c(B) + 1$ for formulas A, B.
- $mc(\Gamma) = \{c(A) \ : \ A \in \Gamma\}$ for a multiset of formulas Γ.

For a query $Q = \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$:

- $mmc(Q) = \{mc(\Gamma_i) \cup mc(\Delta_i) \ : \ 1 \leq i \leq n\}$.

For multisets α, β of integers:

1. $\alpha <_m \beta$ if $\alpha \subset \beta$.
2. $\alpha <_m \beta$ if $\alpha <_m \gamma$ where $\gamma = (\beta - \{j\}) \cup \{i, \ldots, i\}$ and $i < j$.

For multisets ϕ, ψ of multisets of integers:

1. $\phi <_{mm} \psi$ if $\phi \subset \psi$.
2. $\phi <_{mm} \psi$ if $\phi <_{mm} \chi$ where $\chi = (\psi - \{\alpha\}) \cup \{\beta, \ldots, \beta\}$ and $\beta <_m \alpha$.

For each implication rule with premises $Q_1, \ldots, Q_n$ and conclusion Q, we have $mmc(Q_i) <_{mm} mmc(Q)$ for $i = 1, \ldots, n$. Hence we arrive (using $(switch)$ to move to states in the history) at queries which, since $(implication)$ cannot be applied, must have atomic goals, and, since $(l\text{-}reduction)$ and $(r\text{-}reduction)$ cannot be applied, do not have a database formula whose head matches a goal. ■

Moreover, if an irreducible query is valid, then by removing non-atomic database formulas we obtain an atomic query that is also valid.

LEMMA 9. *For* $\mathbf{S} \in \{\mathbf{Ł}, \mathbf{G}, \mathbf{\Pi}\}$, *if the following conditions hold:*

1. $\models_{\mathbf{S}} \Gamma_1, \Pi_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2, \Pi_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n, \Pi_n \vdash^? \Delta_n; R_n)\}$.
2. Γ_i *and* Δ_i *are atomic for* $i = 1, \ldots, n$.
3. $Head(\Pi_1 \cup \ldots \cup \Pi_n) \cap (\Delta_1 \cup \ldots \cup \Delta_n) = \emptyset$.

Then: $\models_{\mathbf{S}} \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$

Proof. Assume $\not\models_{\mathbf{S}} \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$. This means that there is a valuation v for $\mathbf{S}$ such that $\#^v_{\mathbf{S}}\Gamma_i > \#^v_{\mathbf{S}}\Delta_i$ and $\#^v_{\mathbf{S}}(\Gamma_i \cup R_i) \geq \#^v_{\mathbf{S}}\Delta_i$ for $i = 1, \ldots, n$. We define a new valuation w for $\mathbf{S}$ as follows:

$$w(q) \quad = \quad \begin{cases} 1 & \text{if } q \in Head(\Pi_1 \cup \ldots \cup \Pi_n) \\ v(q) & \text{otherwise} \end{cases}$$

Since $Head(\Pi_1 \cup \ldots \cup \Pi_n) \cap (\Delta_1 \cup \ldots \cup \Delta_n) = \emptyset$ we have that $\#^w_{\mathbf{S}}\Delta_i = \#^v_{\mathbf{S}}\Delta_i$ for $i = 1, \ldots, n$. We also get that $\#^w_{\mathbf{S}}\Pi_i = 1$, $\#^w_{\mathbf{S}}\Gamma_i \geq \#^v_{\mathbf{S}}\Gamma_i$, and $\#^w_{\mathbf{S}}R_i \geq \#^v_{\mathbf{S}}R_i$ for $i = 1, \ldots, n$. Hence $\#^w_{\mathbf{S}}(\Pi_i \cup \Gamma_i) > \#^w_{\mathbf{S}}\Delta_i$ and $\#^w_{\mathbf{S}}(\Pi_i \cup \Gamma_i \cup R_i) \geq \#^w_{\mathbf{S}}\Delta_i$ for $i = 1, \ldots, n$, i.e. $\not\models_{\mathbf{S}} \Gamma_1, \Pi_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2, \Pi_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n, \Pi_n \vdash^? \Delta_n; R_n)\}$, a contradiction as required. ■

3.2 Co-NP Decision Procedures

Since the implication rules are both sound and invertible for **Ł**, $\mathbf{G}$, and $\mathbf{\Pi}$, and terminating (modulo loop-checking for $(switch)$), we are able to reduce checking the validity of a query in these logics to checking the validity of irreducible queries, which reduces in turn (by Lemma 9) to checking the validity of *atomic* queries. This is really only a useful step if we then have that checking the validity of atomic queries is less complex than the original validity problem for each logic. In fact, while the validity problem for all these logics is Co-NP complete (see e.g., [Hájek, 1998] for proofs and references), it can be shown (following [Ciabattoni *et al.*, 2005]) that checking validity for atomic queries, and hence for irreducible queries, is in each case *polynomial*.

LEMMA 10. *For an atomic query Q:*

$$\Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

$\models_{\mathbf{S}} Q$ *iff* $C^Q_{\mathbf{S}}$ *is inconsistent over* $[0, 1]$ *for:*

$$C^Q_{\mathbf{S}} = \{\circ_{\mathbf{S}}\Gamma_i > \circ_{\mathbf{S}}\Delta_i, \circ_{\mathbf{S}}(\Gamma_i \cup R_i) \geq \circ_{\mathbf{S}}\Delta_i \ : \ 1 \leq i \leq n\}$$

where $\circ_{\mathbf{S}}\emptyset = 1$ *for* $\mathbf{S} \in \{\text{Ł}, \mathbf{G}, \mathbf{\Pi}\}$*, and:*

- $\circ_{\text{Ł}}(\Gamma) = 1 + \sum_{q \in \Gamma}\{x_q - 1\}$
- $\circ_{\mathbf{G}}(\Gamma) = min_{q \in \Gamma}\{x_q\}$
- $\circ_{\mathbf{\Pi}}(\Gamma) = \prod_{q \in \Gamma}\{x_q\}$

where x_q is a real-valued variable for any propositional variable q.

Proof. Immediate from Definition 2. ■

THEOREM 11. *Checking the validity of atomic queries for* **Ł**, $\mathbf{G}$, *and* $\mathbf{\Pi}$ *is polynomial.*

Proof. By Lemma 10, the result follows if we can show for each atomic query Q that checking the inconsistency of $C^Q_{\mathbf{S}}$ over $[0, 1]$ is polynomial for $\mathbf{S} \in \{\text{Ł}, \mathbf{G}, \mathbf{\Pi}\}$. For **Ł** this follows from the fact that linear programming is polynomial; for $\mathbf{G}$ this follows from a theorem on relations over a finite domain; while for $\mathbf{\Pi}$ we require a rather subtle argument involving linear programming. Details of these proofs may be found in [Ciabattoni *et al.*, 2005]. ■

Co-NP decision procedures are obtained by modifying the reduction rules in order to prevent an exponential growth in the size of queries. To this end we introduce *new propositional variables* during the reduction process, that may be thought of as marking different options for the database.

DEFINITION 12 (New Reduction Rules).

$(l\text{-}reduction)$ From $\Gamma, \Pi \rightarrow q \vdash^? q, \Delta; R; H$ step to

$\Gamma, q \vdash^? q, \Delta; R; H$ and $\vdash^? p, \Pi; \{q\}; H \cup \{(\Gamma, p \vdash^? q, \Delta; R)\}$

where p is a new propositional variable

$(r\text{-}reduction)$ From $\Gamma \vdash^? q, \Delta; R_1; H \cup \{(\Gamma', \Pi \rightarrow q \vdash^? \Delta'; R_2)\}$ step to

$\Gamma \vdash^? q, \Delta; R_1; H \cup \{(\Gamma', q \vdash^? \Delta'; R_2)\}$ and

$\vdash^? \Pi, p; \{q\}; H \cup \{(\Gamma \vdash^? q, \Delta; R_1), (\Gamma', p \vdash^? \Delta'; R_2)\}$

where p is a new propositional variable

LEMMA 13. *The new reduction rules are sound and invertible for* **Ł**, $\mathbf{G}$, *and* $\mathbf{\Pi}$.

Proof. We just consider $(l\text{-}reduction)$ (as before disregarding common parts of the history), the case of $(r\text{-}reduction)$ being very similar. For soundness, we treat several cases. Assume $\#^v_{\mathbf{S}}\Pi \leq v(q)$. If $\#^v_{\mathbf{S}}\Pi = 1$, then we are done by the first premise. If $\#^v_{\mathbf{S}}\Pi < 1$, then extend v with $v(p) = 1$ and the result follows from the second premise. If $\#^v_{\mathbf{S}}\Pi > v(q)$, then we take $v(p) = v(\Pi \rightarrow q)$, and we are done by the second premise. For invertibility, note that if the conclusion is satisfied by a valuation v, then clearly the first premise is satisfied. For the second premise, if $v(q) \geq \#^v_{\mathbf{S}}(\Pi \cup \{p\})$, then it follows for each logic that $v(p) \leq v(\Pi \rightarrow q)$ and hence from the conclusion that $\#^v_{\mathbf{S}}(\Gamma \cup \{p\}) \leq \#^v_{\mathbf{S}}(\Delta \cup \{q\})$. ■

THEOREM 14. *The new reduction rules,* $(implication)$, *and* $(switch)$ *provide Co-NP decision procedures for the validity problems for* **Ł**, $\mathbf{G}$, *and* $\mathbf{\Pi}$.

Proof. As in Proposition 8, for the premises $Q_1, \ldots, Q_n$ and conclusion Q of the new reduction rules, $mmc(Q_i) < mmc(Q)$ for $i = 1, \ldots, n$. Moreover each application of these rules and $(implication)$ gives only a constant increase in the size (number of symbols) in the query. Hence, to show that a formula is not valid in **Ł**, $\mathbf{G}$, or $\mathbf{\Pi}$, we can apply the new reduction rules (sound and invertible by Lemma 13) and $(implication)$ exhaustively (using $(switch)$ to move between different databases) until we reach irreducible queries, making a non-deterministic choice of two branches where necessary. The result then follows from Theorem 11. ■

3.3 Rules for deciding irreducible queries

As an alternative to polynomial decision procedures, we provide goal-directed rules below that check validity *directly* for each logic, thereby obtaining an algorithmic interpretation of **Ł**, $\mathbf{G}$, and $\mathbf{\Pi}$. We begin by defining a “standard” stock of rules for calculi, including a new “mingle” rule that allows the combination of databases and goals.

DEFINITION 15 (Standard Rules). The *standard rules* consist of the implication rules, $(switch)$, and:

$(mingle)$ From $\Gamma_1 \vdash^? q, \Delta_1; R_1; H \cup \{(\Gamma_2, q \vdash^? \Delta_2; R_2)\}$ step to
$\Gamma_1, \Gamma_2 \vdash^? \Delta_1, \Delta_2; \emptyset; H \cup \{(\Gamma_1 \vdash^? q, \Delta_1; R_1), (\Gamma_2, q \vdash^? \Delta_2; R_2)\}$

We now define calculi for **Ł**, **G**, and **Π** by extending the standard rules with different restart and success rules, noting that we write $\subseteq_m$ and $\subseteq_s$ for the *multiset* and *set* subset relations respectively.

DEFINITION 16 (**GDŁ**). **GDŁ** consists of the standard rules plus:

$(success_{\text{Ł}})$ $\Gamma \vdash^? \Delta; R; H$ if $\Delta \subseteq_m \Gamma$

$(restart_{\text{Ł}})$ From $\Gamma_1 \vdash^? q, \Delta_1; R_1; H \cup \{(\Gamma_2 \vdash^? \Delta_2; R_2)\}$
where $R_1 \cup R_2 = R \cup \{q\}$, step to
$\Gamma_1, \Gamma_2 \vdash^? \Delta_1, \Delta_2; R; H \cup \{(\Gamma_1 \vdash^? q, \Delta_1; R_1), (\Gamma_2 \vdash^? \Delta_2; R_2)\}$

THEOREM 17. *Q succeeds in* **GDŁ** *iff* $\models_{\text{Ł}} Q$.

Proof. The left-to-right direction requires proving the soundness of $(mingle)$, $(success_{\text{Ł}})$, and $(restart_{\text{Ł}})$ for **Ł**, which we leave as exercises in elementary arithmetic. For the right-to-left direction we proceed as follows. Let $Q = \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$ be a query. If $\models_{\text{Ł}} Q$, then the set $C_{\text{Ł}}^{Q}$ is inconsistent over $[0,1]$. Hence by linear programming methods there exist $\lambda_1, \mu_1, \ldots, \lambda_n, \mu_n \in \mathbb{N}$ such that $\lambda_i > 0$ for some $1 \leq i \leq n$ and:

$$\bigcup_{i=1}^{n} (\lambda_i + \mu_i)\Delta_i \subseteq_m \bigcup_{i=1}^{n} ((\lambda_i + \mu_i)\Gamma_i \cup \mu_i R_i)$$

We show that Q succeeds in **GDŁ** by induction on $\gamma = \sum_{i=1}^{n} (\lambda_i + \mu_i)$. If $\gamma = 1$ then we have $\Delta_i \subseteq_m \Gamma_i$ for some $1 \leq i \leq n$, and hence that Q succeeds by an application of $(switch)$ if necessary and $(success_{\text{Ł}})$. For $\gamma > 1$ we consider i such that $\lambda_i > 0$. If $\Delta_i \subseteq_m \Gamma_i$, then we are done by $(switch)$ if necessary and $(success_{\text{Ł}})$. Otherwise, we have $q \in \Delta_i$ where one of the following cases occurs:

1. $q \in \Gamma_j$ for some $j \neq i$ with $\lambda_j > 0$. Since we can always apply $(switch)$, we can assume without loss of generality that $i = 1$ and $j = 2$. Now by applying $(mingle)$ to Q we obtain a query Q':

   $\Gamma_1, \Gamma_2 - \{q\} \vdash^? \Delta_1 - \{q\}, \Delta_2; \emptyset; \{(\Gamma_1 \vdash^? \Delta_1; R_1), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$

If $\lambda_1 \geq \lambda_2$, then:

$$(\lambda_1 - \lambda_2)\Delta_1 \cup \lambda_2(\Delta_1 - \{q\} \cup \Delta_2) \cup \Delta' \subseteq_m (\lambda_1 - \lambda_2)\Gamma_1 \cup \lambda_2(\Gamma_1 \cup \Gamma_2 - \{q\}) \cup \Gamma'$$

for $\Delta' = \bigcup_{i=3}^n \lambda_i \Delta_i \cup \bigcup_{i=1}^n \mu_i \Delta_i$ and $\Gamma' = \bigcup_{i=3}^n \lambda_i \Gamma_i \cup \bigcup_{i=1}^n \mu_i(\Gamma_i \cup R_i)$. Since $(\lambda_1 - \lambda_2) + \lambda_2 + \sum_{i=3}^n \lambda_i + \sum_{i=1}^n \mu_i < \lambda$, by the induction hypothesis, Q' succeeds in $\mathbf{GDŁ}$ and we are done. The case where $\lambda_2 \geq \lambda_1$ is very similar.

2. $q \in R_j$ for some j with $\mu_j > 0$. If $i \neq j$, then without loss of generality we assume that $i = 1$ and $j = 2$. Otherwise $i = j$, and observe that either $\Delta_i \subseteq_m \Gamma_i$ and we can apply $(success_{\mathbf{Ł}})$, or there must exist $k \neq i$ such that either $\lambda_k > 0$ or $\mu_k > 0$. If the latter, then assume without loss of generality that $i = 1$ and $k = 2$. Now, in both cases, by applying $(restart_{\mathbf{Ł}})$ we obtain:

$$Q' = \Gamma_1, \Gamma_2 \vdash^? \Delta_1 - \{q\}, \Delta_2; R_1; \{(\Gamma_1 \vdash^? \Delta_1; R_1), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

We then proceed similarly to above, separating the possibilities that $\lambda_1 \leq \mu_2$ and $\mu_2 \leq \lambda_1$. ■

DEFINITION 18 ($\mathbf{GDG}$). $\mathbf{GDG}$ consists of the standard rules plus:

$(success_{\mathbf{G}})$ $\Gamma \vdash^? \Delta; R; H$ if $\Delta \subseteq_s \Gamma$

$(restart_{\mathbf{G}})$ From $\Gamma_1 \vdash^? q, \Delta_1; R; H \cup \{(\Gamma_2 \vdash^? \Delta_2; \{q\})\}$ step to $\Gamma_1, \Gamma_2 \vdash^? \Delta_1, \Delta_2; \emptyset; H \cup \{(\Gamma_1 \vdash^? q, \Delta_1; R), (\Gamma_2 \vdash^? \Delta_2; \{q\})\}$

To show the completeness of $\mathbf{GDG}$ it will be helpful to have the following lemma, which establishes (roughly speaking) that we can restrict our attention to queries with only one goal per database.

LEMMA 19. *For queries* $Q = \Gamma \vdash^? \Delta_1, \Delta_2, R; H$, $Q_1 = \Gamma \vdash^? \Delta_1, R; H$ *and* $Q_2 = \Gamma \vdash^? \Delta_2, R; H$*:*

1. *If* $\models_{\mathbf{G}} Q$, *then* $\models_{\mathbf{G}} Q_1$ *and* $\models_{\mathbf{G}} Q_2$.

2. *If* Q_1 *and* Q_2 *succeed in* $\mathbf{GDG}$, *then* Q *suceeds in* $\mathbf{GDG}$.

Proof. The first part follows from the definition of validity for $\mathbf{G}$, the second is proved by induction on the joint lengths of derivations of Q_1 and Q_2 in $\mathbf{GDG}$. ■

THEOREM 20. *Q succeeds in* $\mathbf{GDG}$ *iff* $\models_{\mathbf{G}} Q$.

Proof. The left-to-right direction requires checking the soundness of $(mingle)$, $(success_{\mathbf{G}})$, and $(restart_{\mathbf{G}})$ for $\mathbf{GDG}$ (an easy exercise). For the right-to-left direction, let $Q = \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$ be a query, assuming by Lemma 19 that $|\Delta_i| = 1$ for $i = 1, \ldots, n$. If $\models_{\mathbf{G}} Q$ then the set $C^Q_{\mathbf{G}}$ is inconsistent over $[0, 1]$. By a result on linear orders (see e.g. [Baaz *et al.*, 2001] for details), this holds iff there are $p_1, \ldots, p_m$, and distinct j_i for $i = 1, \ldots, m$ such that:

(a) $p_i \in \Delta_{j_i}$ and $p_{i+1} \in \Gamma_{j_i} \cup R_{j_i}$ for $i = 1, \ldots, m - 1$.

(b) $p_m \in \Delta_{j_m}$ and $p_1 \in \Gamma_{j_m}$.

We show that such queries are derivable in $\mathbf{GDG}$ by induction on m. If $m = 1$, then using $(switch)$ if necessary, we succeed since $\{p_1\} = \Delta_{j_m} \subseteq_s \Gamma_{j_m}$. For $m > 1$, assuming without loss of generality that $j_m = 1$ and $j_{m-1} = 2$, we have $p_m \in \Delta_1$ and $p_1 \in \Gamma_1$, and $p_{m-1} \in \Delta_2$ and $p_m \in \Gamma_2 \cup R_2$. If $p_m \in \Gamma_2$, then we apply $(mingle)$ to obtain the query Q':

$$\Gamma_1, \Gamma_2 - \{p_m\} \vdash^? \Delta_2; \emptyset; \{(\Gamma_1 \vdash^? \Delta_1; R_1), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

But now, since $\Delta_2 = \{p_{m-1}\}$, we have $p_1, \ldots, p_{m-1}$ meeting requirements (a) and (b) above, and hence by the induction hypothesis Q' succeeds in $\mathbf{GDG}$. For the case where $p_m \in R_2$, we apply $(restart_{\mathbf{G}})$ to obtain Q'':

$$\Gamma_1, \Gamma_2 \vdash^? \Delta_2; \emptyset; \{(\Gamma_1 \vdash^? \Delta_1; R_1), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$$

Again, since $\Delta_2 = \{p_{m-1}\}$, we have $p_1, \ldots, p_{m-1}$ meeting requirements (a) and (b) above, and hence by the induction hypothesis Q'' succeeds in $\mathbf{GDG}$. ■

To obtain a calculus for $\mathbf{\Pi}$, we adapt the $(restart_{\mathbf{Ł}})$ rule of $\mathbf{GDŁ}$ to check that the restart formula used is "zero-ok", i.e. that the query is satisfied whenever this formula takes value 0.

DEFINITION 21 ($\mathbf{GD\Pi}$). $\mathbf{GD\Pi}$ is exactly the same as for $\mathbf{GDŁ}$ except that for $(restart_{\mathbf{\Pi}})$ we add the condition:

- If $q \in R_1$, then q is zero-ok for the conclusion of the rule.

where q is *zero-ok* for a query $Q = \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$ if one of the following two conditions holds:

1. $q \in \Gamma_i$ for some i, $1 \leq i \leq n$.
2. $R_i = \{q\}$ for some i, $1 \leq i \leq n$, and p is zero-ok for all $p \in \Delta_i$.

LEMMA 22. *Let $Q = \Gamma \vdash^? \Delta; \{q\}; H$ be a query, and let $Q' = \Gamma \vdash^? \Delta; \emptyset; H$:*

1. *If $\models_{\mathbf{\Pi}} Q$ and q is not zero-ok, then $\models_{\mathbf{\Pi}} Q'$.*
2. *If Q' succeeds in $\mathbf{GD\Pi}$, then Q succeeds in $\mathbf{GD\Pi}$.*

Proof. The first part follows from the definition of validity for $\mathbf{\Pi}$, the second via an induction on the height of a proof of Q' in $\mathbf{GD\Pi}$. ■

THEOREM 23. *Q succeeds in $\mathbf{GD\Pi}$ iff $\models_{\mathbf{\Pi}} Q$.*

Proof. The left-to-right direction follows by checking the soundness of $(mingle)$, $(success_{\mathbf{\Pi}})$, and $(restart_{\mathbf{\Pi}})$ for $\mathbf{GD\Pi}$ (a straightforward exercise). For the right-to-left direction, let $Q = \Gamma_1 \vdash^? \Delta_1; R_1; \{(\Gamma_2 \vdash^? \Delta_2; R_2), \ldots, (\Gamma_n \vdash^? \Delta_n; R_n)\}$ be a query. We can assume by Lemma 22 that all restarts occurring in Q are zero-ok. If $\models_{\mathbf{\Pi}} Q$, then $C^Q_{\mathbf{\Pi}}$ is inconsistent over $[0,1]$ and therefore also $(0,1]$. Hence, by linear programming methods there exist $\lambda_1, \mu_1, \ldots, \lambda_n, \mu_n \in \mathbb{N}$ such that $\lambda_i > 0$ for some i, $1 \leq i \leq n$, and:

$$\bigcup_{i=1}^{n} (\lambda_i + \mu_i)\Delta_i \subseteq_m \bigcup_{i=1}^{n} ((\lambda_i + \mu_i)\Gamma_i \cup \mu_i R_i)$$

We now proceed as for $\mathbf{GDŁ}$ with a proof by induction on $\gamma = \sum_{i=1}^n (\lambda_i + \mu_i)$ that Q succeeds in $\mathbf{GD\Pi}$, the only difference being that every application of $(restart_{\mathbf{\Pi}})$ requires (which holds by assumption) that the restart formula is zero-ok. ■

EXAMPLE 24. Consider the following atomic query:

$$p \vdash^? q; \{q\}; \{(q \vdash^? p, p; \emptyset)\}$$

In the case of $\mathbf{G}$ we apply $(mingle)$ and obtain:

$$p, q \vdash^? q, p, p; \emptyset; \{(p \vdash^? q; \{q\}), (q \vdash^? p, p; \emptyset)\}$$

which succeeds by $(success_{\mathbf{G}})$ for $\mathbf{GDG}$ since $\{q, p, p\} \subseteq_s \{p, q\}$. For $\mathbf{Ł}$ on the other hand, we apply the $(restart_{\mathbf{Ł}})$ rule of $\mathbf{GDŁ}$, giving:

$$p, q \vdash^? p, p; \emptyset; \{(p \vdash^? q; \{q\}), (q \vdash^? p, p; \emptyset)\}$$

We now apply $(mingle)$ to get:

$$p, q, p \vdash^? p, p, q; \emptyset; \{(p \vdash^? q; \{q\}), (q \vdash^? p, p; \emptyset), (p, q \vdash^? p, p; \emptyset)\}$$

which succeeds by $(success_{\mathbf{Ł}})$ in $\mathbf{GDŁ}$ since $\{p, p, q\} \subseteq_m \{p, q, p\}$. Moreover, since in the application of $(restart)$ we have that q occurs in one of the databases and is hence zero-ok, we also obtain a proof in $\mathbf{GD\Pi}$.

4 Concluding Remarks

Uniform goal-directed rules have been defined for the implicational fragments of the three fundamental fuzzy logics $\mathbf{Ł}$, $\mathbf{G}$, and $\mathbf{\Pi}$, that can be extended to both co-NP decision procedures and fully goal-directed calculi. There remain, however, a number of issues requiring further attention. In particular, we began this chapter by recalling logic-programming as a source and underlying motivation for the goal-directed paradigm. In order to develop logic programming languages for fuzzy logics, however, various extensions of the proof methods presented here are required. First of all, a richer propositional language should be defined using suitable notions of database and goal formulas. Note, however, that the importance of extending the language may differ depending on the particular logic; for example, $\mathbf{Ł}$ may be based on a language with just $\to$ and $\bot$, while for $\mathbf{G}$ and $\mathbf{\Pi}$ more connectives are necessary. These issues have been considered for $\mathbf{Ł}$ and $\mathbf{G}$ in [Metcalfe *et al.*, 2003] and [Metcalfe *et al.*, 2004b] respectively; for example, the following rules are suitable for treating conjunctive and disjunctive goals in all three logics:

$$\begin{array}{ll} (and) & \text{From } \Gamma \vdash^? A \wedge B, \Delta; R; H \text{ step to} \\ & \Gamma \vdash^? A, \Delta; R; H \text{ and } \Gamma \vdash^? B, \Delta; R; H \\ (or) & \text{From } \Gamma \vdash^? A \vee B, \Delta; R; H \text{ step to} \\ & \Gamma \vdash^? A, \Delta; R; H \cup \{(\Gamma \vdash^? B, \Delta; R)\} \end{array}$$

For logic programming applications, the language should also be extended to the first-order setting, a natural compromise being the fragment containing all universally-quantified formulas. In this case, the reduction rules should incorporate a suitable *unification* mechanism. Also, *logical consequence* is crucial, being the basis for fuzzy logic programming methods found in the literature [Klawonn and Kruse, 1994; Vojtás, 2001; Smutná-Hlinená and Vojtás, 2004]. That is, queries should be permitted that express that a goal G is a logical consequence of a database Γ, rather than the internal consequence that $\Gamma^* \to G$ is valid, where Γ^* is a t-norm combination of the formulas of Γ. In the case of $\mathbf{Ł}$ and $\mathbf{\Pi}$ these two notions of consequence are different, although they may be related via the deduction theorem for the logics, see [Metcalfe *et al.*, 2004b] for details.

From a more theoretical perspective, observe that although we have defined uniform goal-directed methods for $\mathbf{Ł}$, $\mathbf{G}$, and $\mathbf{\Pi}$, these results also extend to a number of other logics. In particular, the uniform rules are (obviously) sound and invertible, and provide the basis for decision procedures for *intersections* of the three main fuzzy logics discussed and axiomatized in [Cignoli *et al.*, 2000]. The rules are also sound and invertible for *finite-valued* Łukasiewicz and Gödel logics, and a number of related fuzzy logics such as Cancellative hoop logic [Esteva *et* *al.*, 2003]. Hence the question remains as to exactly which logics can be captured using these rules. In particular, it would be interesting to show that the rules are uniform for all t-norm logics, including Hájek's logic $\mathbf{BL}$ which still lacks a reasonable calculus. Finally, observe that the goal-directed calculi for **Ł**, $\mathbf{G}$, and $\mathbf{\Pi}$, are really very closely related, the only differences lying in the underlying data structure, sets or multisets, and slight alterations in the restart rules. Moreover, note that a goal-directed calculus for *Classical logic* is obtained simply by changing the $(mingle)$ rule in $\mathbf{GDG}$ to allow the combination of databases without the corresponding combination of goals. These observations give further evidence of the uniformity of the goal-directed methodology, and more generally of the algorithmic proof perspective pioneered by Gabbay.

BIBLIOGRAPHY

[Avron, 1987] A. Avron. A constructive analysis of RM. *Journal of Symbolic Logic*, 52(4):939–951, 1987.

[Avron, 1991] A. Avron. Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of Mathematics and Artificial Intelligence*, 4(3–4):225–248, 1991.

[Baaz *et al.*, 2001] M. Baaz, A. Ciabattoni, and C. Fermüller. Cut-elimination in a sequents-of-relations calculus for Gödel logic. In *International Symposium on Multiple Valued Logic (ISMVL'2001)*, pages 181–186. IEEE, 2001.

[Bollen, 1991] A. W. Bollen. Relevant logic programming. *Journal of Automated Reasoning*, pages 563–585, 1991.

[Bonner, 1990] A. J. Bonner. Hypothetical datalog: Complexity and expressibility. *Theoretical Computer Science*, pages 3–51, 1990.

[Ciabattoni *et al.*, 2005] A. Ciabattoni, C. G. Fermüller, and G. Metcalfe. Uniform Rules and Dialogue Games for Fuzzy Logics. In *Proceedings of LPAR 2004*, volume 3452 of *LNAI*, pages 496–510. Springer, 2005.

[Cignoli *et al.*, 2000] R. Cignoli, F. Esteva, L. Godo, and A. Torrens. Basic fuzzy logic is the logic of continuous t-norms and their residua. *Soft Computing*, 4(2):106–112, 2000.

[Esteva and Godo, 2001] F. Esteva and L. Godo. Monoidal t-norm based logic: towards a logic for left-continuous t-norms. *Fuzzy Sets and Systems*, 124:271–288, 2001.

[Esteva *et al.*, 2003] F. Esteva, L. Godo, P. Hájek, and F. Montagna. Hoops and fuzzy logic. *Journal of Logic and Computation*, 13(4):532–555, 2003.

[Gabbay and Olivetti, 2000] D. Gabbay and N. Olivetti. *Goal-directed Proof Theory*. Kluwer Academic Publishers, 2000.

[Gabbay and Olivetti, 2002a] D. Gabbay and N. Olivetti. Goal oriented deductions. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 9, pages 199–285. Kluwer Academic Publishers, second edition, 2002.

[Gabbay and Olivetti, 2002b] D. Gabbay and Nicola Olivetti. Interpolation in goal-directed proof systems 1. In *Logic Colloquium 2001*. A K Peters Ltd, 2002.

[Gabbay and Reyle, 1984] D. Gabbay and Uwe Reyle. N-Prolog: An extension of Prolog with hypothetical implication. *Journal of Logic Programming*, 4:319–355, 1984.

[Gabbay *et al.*, 2000] D. Gabbay, L. Giordano, A. Martelli, N. Olivetti, and M. L. Sapino. Conditional reasoning in logic programming. *Journal of Logic Programming*, 44(1–3):37–74, 2000.

[Gabbay, 1985] D. Gabbay. N-Prolog part 2. *Journal of Logic Programming*, pages 251–283, 1985.

[Gabbay, 1992] D. Gabbay. Elements of algorithmic proof. In S. Abramsky et al., editors, *Handbook of Logic in Theoretical Computer Science*, volume 2, pages 311–413. Oxford University Press, 1992.

[Gabbay, 1996] D. Gabbay. *Labelled Deductive Systems*, volume 1—Foundations. Oxford University Press, 1996.

[Giordano *et al.*, 1992] L. Giordano, A. Martelli, and G. F. Rossi. Extending Horn clause logic with implication goals. *Theoretical Computer Science*, 95:43–74, 1992.

[Hájek, 1998] P. Hájek. *Metamathematics of Fuzzy Logic*. Kluwer, Dordrecht, 1998.

[Harland and Pym, 1991] J. Harland and D. Pym. The uniform proof-theoretic foundation of linear logic programming. In *Proc. of the 1991 International Logic Programming Symposium*, pages 304–318, 1991.

[Harland, 1997] J. Harland. On goal-directed provability in classical logic. *Computer Languages*, 23(2–4):161–178, 1997.

[Hodas and Miller, 1994] J. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110:327–365, 1994.

[Hudelmaier, 1990] J. Hudelmaier. Decision procedure for propositional n-prolog. In P. Schroeder-Heister, editor, *Extensions of Logic Programming*, pages 245–251. Springer-Verlag, 1990.

[Jenei and Montagna, 2002] S. Jenei and F. Montagna. A proof of standard completeness for Esteva and Godo's MTL logic. *Studia Logica*, 70(2):183–192, 2002.

[Klawonn and Kruse, 1994] F. Klawonn and R. Kruse. A Łukasiewicz logic based Prolog. *Mathware & Soft Computing*, 1(1):5–29, 1994.

[McCarty, 1988] L. T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, 5(2):93–132, 1988.

[Metcalfe *et al.*, 2003] G. Metcalfe, N. Olivetti, and D. Gabbay. Goal-directed calculi for Gödel-Dummett logics. In M. Baaz and J. A. Makowsky, editors, *Proceedings of CSL 2003*, volume 2803 of *LNCS*, pages 413–426. Springer, 2003.

[Metcalfe *et al.*, 2004a] G. Metcalfe, N. Olivetti, and D. Gabbay. Analytic proof calculi for product logics. *Archive for Mathematical Logic*, 43(7):859–889, 2004.

[Metcalfe *et al.*, 2004b] G. Metcalfe, N. Olivetti, and D. Gabbay. Goal-directed methods for Łukasiewicz logics. In J. Marcinkowski and A. Tarlecki, editors, *Proceedings of CSL 2004*, volume 3210 of *LNCS*, pages 85–99. Springer, 2004.

[Metcalfe *et al.*, 2005a] G. Metcalfe, N. Olivetti, and D. Gabbay. Łukasiewicz logic: From proof systems to logic programming. To appear in Logic Journal of the IGPL, 2005.

[Metcalfe *et al.*, 2005b] G. Metcalfe, N. Olivetti, and D. Gabbay. Sequent and hypersequent calculi for abelian and Łukasiewicz logics. *ACM Transactions on Computational Logic*, 6(3):578–613, 2005.

[Miller *et al.*, 1991] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[Nadathur, 1998] G. Nadathur. Uniform provability in classical logic. *Journal of Logic and Computation*, 8:209–229, 1998.

[Pottinger, 1983] G. Pottinger. Uniform, cut-free formulations of T, S4 and S5 (abstract). *Journal of Symbolic Logic*, 48(3):900, 1983.

[Reed and Loveland, 1992] D. W. Reed and D. W. Loveland. A comparison of three Prolog extensions. *Journal of Logic Programming*, 12(1):25–50, 1992.

[Smutná-Hlinená and Vojtás, 2004] D. Smutná-Hlinená and P. Vojtás. Graded many-valued resolution with aggregation. *Fuzzy Sets and Systems*, 143(1):157–168, 2004.

[Vojtás, 2001] P. Vojtás. Fuzzy logic programming. *Fuzzy Sets and Systems*, 124:361–370, 2001.