Complexity of Power Default Reasoning

Guo-Qiang Zhang
Department of Computer Science
University of Georgia
Athens, Georgia 30602
U.S.A.
gqz@cs.uga.edu

William C. Rounds
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, Michigan 48109
U.S.A.
rounds@umich.edu

Abstract

This paper derives a new and surprisingly low complexity result for inference in a new form of Reiter's propositional default logic [18]. The problem studied here is the ***default inference problem*** *whose fundamental importance was pointed out by Kraus, Lehmann, and Magidor [12]. We prove that "normal" default inference, in propositional logic, is a problem complete for co-NP(3), the third level of the* ***Boolean*** *hierarchy [4]. Our result (by changing the underlying semantics) contrasts favorably with a similar result of Gottlob [7], who proves that standard default inference is* Π_2^P*-complete. Our inference relation also obeys all of the laws for* ***preferential consequence relations*** *set forth by Kraus, Lehmann, and Magidor. In particular, we get the property of being able to reason by cases and the law of cautious monotony. Both of these laws fail for standard propositional default logic.*

The key technique for our results is the use of Scott's domain theory to integrate defaults into partial model theory of the logic, instead of keeping defaults as quasi-proof rules in the syntax. In particular, reasoning disjunctively entails using the ***Smyth powerdomain****.*

1 Introduction

1.1 General remarks

Non-monotonic reasoning is a significant research area spanning the artificial intelligence community and the community in logic programming and deductive databases. The general idea is to allow pseudo-logical inference rules, and/or "defeasible" conditionals, in order to allow database systems to derive conclusions or to answer queries in the absence of explicitly stored hard information. There are many ways of approaching the semantics of such systems. However, it seems that a key problem is to identify what we mean by "absence of information". In this connection, the database research group at Penn has pioneered the use of Scott domain theory as a tool for database query language design. A good overview can be found in Libkin [13]. In fact, one key idea of our semantics (updating) is taken from this work.

We follow the idea of using domain theory, but our focus is specifically on non-monotonic phenomena. This entails moving from continuous functions to discontinuous ones, and it leads us to Reiter's *extension* operator as a non-monotonic version of the least fixpoint operator, which appropriately generalizes to arbitrary Scott domains. (The fact that this operator could be a very general one was first noticed in a paper by Marek, Nerode, and Remmel [14].)

In Reiter's system, *extensions* of a first-order or propositional theory are obtained with respect to a set Σ of defaults of the form

$$\frac{P : Q}{Q},$$

which are interpreted informally as proof rules: If P has been proved, and Q is consistent, then infer Q. (These rules are of a special type called *normal*; in general the consequent of a rule could be a third formula R.)

Let W be a theory (typically, generated by a single formula). An extension of W is a theory E obtained as follows. Let $\eta(W, X)$ be the least possible logically closed theory containing W and closed under application of the default rules, as long as application of any rule gives a theory consistent with X. If $\eta(W, E) = E$ we say that E is an extension of W. In contrast to the usual kind of deductive closure of a theory, which is unique, there can be multiple extensions. However, one can recapture the idea of a "default theorem" by requiring a formula R to occur in all extensions of W. In this case we might write $W \mid\!\sim R$, where we are thinking of W as a single formula.

A seminal paper of Kraus, Lehmann, and Magidor [12] gives a uniform way of comparing differing versions of non-monotonic logic. They consider possible properties of a reasonable notion of non-monotonic entailment $\mid\!\sim$ between formulas. Among these are

- The Or law (reasoning by cases): If $W \mid\!\sim R$, and $V \mid\!\sim R$, then $(W \vee V) \mid\!\sim R$.
- The Cautious Monotony Law: if $W \mid\!\sim R$, and $W \mid\!\sim V$, then $(W \wedge V) \mid\!\sim R$.

The reason that default logic is called "non-monotonic" is that the principle of "strengthening the hypothesis" fails (a property in common with linear logic.) The Cautious Monotony law is a fairly reasonable replacement for this principle. A major problem with Reiter's logic, though, is that both these laws fail. Our version of Reiter's logic recaptures both laws; the best previous result in this direction is Brewka's *cumulative default logic*, which recaptures the Cautious Monotony law. In the complexity direction, Gottlob and (independently) Stillman [20] have shown that in propositional default logic, deciding the non-monotonic inference relation is a problem complete for Π_2^P, a result which is somewhat paradoxical, since defaults are supposed to speed up the ordinary process of inference.

We now give an overview of the techniques used in this paper. First, We generalize the idea of extensions by letting W, X and so on range over an arbitrary Scott domain. (In Reiter's case, the domain is the collection of deductively closed theories, ordered by set-theoretic inclusion.) We notice that domains provide a *model theory* for programs, so we think of defaults as programs. Since default conlusions could be disjunctive, we consider defaults as nondeterministic programs, and model them using the Smyth powerdomain.

In somewhat more detail, we consider a (syntactic) system of normal defaults to be just a finite collection Σ of pairs of the form (τ, θ), where τ, θ are ordinary propositional formulas using "and", "or", and "not" connectives. We read these in the usual way: " normally, if τ then θ." We also have an "initial" formula ϕ representing given information, and a formula ψ which may or may not be skeptically entailed by ϕ and Σ. The initial formula (in fact every ordinary formula) is interpreted using Kleene's three-valued truth tables. This results in the "initial disjunctive state" being the set of all partial truth assignments satisfying ϕ, which is a compact open set in the Scott topology over the domain D of partial truth assignments – i.e., a Smyth powerdomain element.

Instead of using syntactic defaults as proof rules, we think of Σ as defining a "nondeterministic state transformer" semantics. (The Smyth powerdomain was in fact originally designed for just this purpose.) We show how to induce *semantic defaults* from the syntactic presentation of default constraints in Σ. Extensions of a normal default set can be defined over *any* Scott domain, so this works perfectly well for the Smyth powerdomain. Extensions provide a fixpoint construction for the state transformers. The notion of non-monotonic consequence is now recaptured model-theoretically by saying that $\phi \mid\!\sim \psi$ (with respect to a given set Σ of defaults) if every extension of the set of models of ψ is contained in the set of models of ϕ (note the characteristic reversal associated with the Smyth powerdomain.)

We concentrate here on propositional logic, and prove that default inference, with normal defaults, is a problem complete for co-NP(3), the third level of the *Boolean* hierarchy [4]. This means that the class of valid instances of our problem can be written as a set $(L_1 \cap L_2) \cup L_3)$, where L_1 and L_3 are in co-NP, and L_2 is in NP. Operationally speaking, this means that to solve an instance of skeptical entailment, we will have to chain together at most two calls to an NP oracle; and in fact SAT is an obvious candidate for such an oracle. The best previous result using variants of the standard semantics for defaults is by Gottlob [8], who shows that for the *stationary* semantics of default logic, due to Przymusinska and Przymusinski [17] one obtains a $P^{NP[\log n]}$ complete inference problem.

Here is a summary of the properties of our method. Instances of our problem are triples (ϕ, Σ, ψ), where ϕ and ψ are propositional formulas, and Σ is a set of default constraints; the question is whether or not every model-theoretic extension of the set of models of ϕ is contained in the set of models of ψ.

- We isolate an important subclass of problem instances for which there is a simple co-NP test for deciding skeptical non-monotonic entailment. These are the so-called *safe* instances. Roughly, a safe instance is a model (partial truth assignment) for ϕ which satisfies all the defaults $\tau \rightsquigarrow \theta$ as if they were standard material implications. The entailment algorithm only works correctly if we know ahead of time that we have safe problem instances. It is NP-complete to tell if we do have such an instance, but there do exist polynomial criteria which we can impose on instances to guarantee safety.

- All other instances of the problem are the so-called *non-safe* instances. If we know ahead of time that we are looking at non-safe instances, then again we have a co-NP algorithm for deciding non-monotonic entailment. In fact, non-monotonic inference is identical to *monotonic* entailment in the non-safe setting.

- To justify our algorithms, we prove a classification theorem called the *Dichotomy Theorem*. This result shows us the structure of our model-theoretic extensions in both the safe and the unsafe setting. The theorem says that either there is a safe instance, in which case there is a unique default extension of a set M of assignments, or else no instances are safe and the multiple extensions are each singleton sets $\{g\}$ of maximal (total) truth assignments $g \in M$.

- We use the safe-instance testing algorithm to classify problem instances into just these two cases. Thus we need only three NP "oracles": one to classify the instances as safe or unsafe, and the other two to finish the reasoning as above. In fact, only two oracles need to be chained together.

- Standard propositional default reasoning runs into problems with repeated consistency checks, having to use an NP oracle for this. Our algorithm avoids these checks, except for the safety test at the beginning. This was an unexpected property of our semantics.

- In the case of conjunctive systems, where ϕ and ψ are in DNF, and the conclusions θ of the default rules contain no disjunctions, the skeptical inference problem is polynomial. This contrasts sharply with results of Kautz and Selman [11], who show NP-hardness results for similar (normal) instances in standard default logic.

- In our system, we can carry out default reasoning by cases, and the system also obeys the desirable law of Cautious Monotony [12]. Both of these laws fail in standard default logic.

1.2 Previous work

Buneman and his students [3] pioneered the use of domain theory in database applications. Libkin's thesis [13] contains a very good overview. The Buneman school considers many more powerdomain constructions besides the Smyth powerdomain.

In artificial intelligence and natural language Pollard and Moshier [16] considered a number of ways to use powerdomains to model grammatical inference, though the first use of partial-order techniques in a setting related to artificial intelligence, was by Pereira and Shieber [15] in providing a semantics for disjunctive feature structures in computational linguistics. Gunter it et al. [9] have used powerdomain orderings to model semantics of version spaces and ATMs's. Our semantics was anticipated as well by Ait-Kaci [2], but (for his ϵ-terms) he did not use domain theory explicitly. Another implicit use of our semantics is in a paper by Gelfond *et al.* [6], who introduce a notion of *disjunctive defaults* in order to overcome certain problematic examples involving disjunctive reasoning in default logic.

Plan of the paper. We present background definitions in the next section. In Section 3 we talk about defaults, informally preenting our semantics. We feel that this is an important section showing how our defaults really work. Section 4 is the technical heart of the paper, containing the the Dichotomy Theorem. The last sections contain algorithms and completeness results.

2 Basic definitions

2.1 Complexity definitions

The boolean hierarchy (BH) [4, 5] refers to language classes defined as "hardware" over NP. refers to language classes defined as follows:

NP(0)=P,

NP$(2i+1)$=$\{L_A \cup L_B \mid L_A \in$ NP$(2i),\ L_B \in$ NP$\}$,

NP$(2i+2)$=$\{L_A \cap L_B \mid L_A \in$ NP$(2i+1),\ L_B \in$ co-NP$\}$,

BH=$\bigcup_{i>0}$ NP(i).

Naturally, there are the complementary classes

$$\text{co-NP}(i) = \{L \mid \overline{L} \in \text{NP}(i)\}.$$

Of particular interest to us is the third level of the boolean hierarchy. We have

NP(3)=$\{(L_1 \cap L_2) \cup L_3 \mid L_1, L_3 \in$ NP, $L_2 \in$ co-NP$\}$, and so

co-NP(3)=$\{(L_1 \cap L_2) \cup L_3 \mid L_1, L_3 \in$ co-NP, $L_2 \in$ NP$\}$.

The Boolean hierarchy is defined in terms of certain special Boolean combinations of NP sets, but in [4] it is shown that the BH is exactly the Boolean closure of NP. One can therefore think of the class as defined by Boolean expressions, where the basic expressions are just NP predicates.

By contrast, the polynomial hierarchy is defined inductively allowing each level of the hierarchy to use as many oracle calls to the previous level as wanted, provided there are not more than polynomially many calls. The notions of hard and complete problems are defined as usual with respect to log space reducibility.

2.2 Domain theory

We include this section for the sake of making the paper self-contained, but we point out that the full generality of these definitions is not really needed, as we will be working in a finite domain.

- A complete partial order (cpo) is a poset $(X, \sqsubseteq)$ with a bottom element $\perp$ and least upper bounds of directed sets. (A set L is directed if it is nonempty, and for any two elements x, y in L there is z in L with x, y both $\sqsubseteq z$.) We read "$x \sqsubseteq y$" as "x subsumes y".

- A subset $Y \subseteq X$ is consistent (or compatible) if it has an upper bound in X.

- If $(X, \sqsubseteq)$ is a cpo, we define a new domain $X^\top$ by adjoining a new element $\top$ to X, and put $x \sqsubseteq \top$ for $x \in X$.

- A compact (finite) element x of X is one such that whenever $x \sqsubseteq \bigsqcup L$ with L directed, we also have $x \sqsubseteq y$ for some $y \in L$. The set of compact elements of a cpo X is written as $\mathsf{K}(X)$.

- A cpo is algebraic if each element is the least upper bound of a set of compact elements. We say that the set of compact elements is a **basis** of X.

- A Scott domain is an algebraic cpo in which every consistent subset has a least upper bound.

- We write $x \uparrow y$ if the set $\{x, y\}$ is consistent.

- $\uparrow x = \{y \mid y \sqsupseteq x\}$.

Every domain (in fact algebraic cpo) has a "natural" positive logic associated with it. This logic is given by the *Scott topology* of the domain.

Definition 2.1 *Let $(X, \sqsubseteq)$ be a cpo. A set $U \subseteq X$ is said to be* ***Scott open*** *if (i) it is upward closed ($x \in U$ and $x \sqsubseteq y \to y \in U$), and (ii) for any directed $L \subseteq X$, we have $\bigsqcup L \in U$ iff $U \cap L \neq \emptyset$.*

2.3 Three-valued logic

First we define our propositional language L. It is nothing more than the usual language for propositional logic over a countable set V of propositional variables, with the addition of a special formula $\mathbf{true}$. We restrict the connectives to conjunction, disjunction, and negation. The departure from standard logic comes in the semantics, where we use partial truth assignments to propositional variables, and interpret formulae by using Kleene's three-valued truth tables.

Definition 2.2 *A* partial truth assignment *is a function d from V into the flat domain $2 = \{0, 1, \perp\}$. The set $D = D(V)$ of all such assignments is partially ordered pointwise in the standard way, and is a Scott domain under this ordering. We will let the variables $d, e, \ldots$ range over this domain.*

For our complexity results we will consider truth assignments from a *finite subset* of V, so that we get a finite Scott domain. The finite subset will be the set of variables appearing in the instances of our problem.

We recall the (strong) truth tables proposed by Kleene for the Boolean connectives:

$\wedge$	**1**	**0**	$\perp$
1	**1**	**0**	$\perp$
0	**0**	**0**	**0**
$\perp$	$\perp$	**0**	$\perp$

$\vee$	**1**	**0**	$\perp$
1	**1**	**1**	**1**
0	**1**	**0**	$\perp$
$\perp$	**1**	$\perp$	$\perp$

p	$\neg p$
1	**0**
0	**1**
$\perp$	$\perp$

Figure 1. The strong Kleene tables.

Definition 2.3 *Let φ be a formula of L. We define the valuation $[\![\varphi]\!] : D \to \{0, 1, \perp\}$ recursively on the structure of φ:*

1. $[\![\mathbf{true}]\!](d) = 1$;
2. $[\![p]\!](d) = d(p)$, *where p is a variable;*

3. $[\![\varphi \wedge \psi]\!](d) = [\![\varphi]\!](d) \wedge [\![\psi]\!](d)$;
4. $[\![\varphi \vee \psi]\!](d) = [\![\varphi]\!](d) \vee [\![\psi]\!](d)$;
5. $[\![\neg\varphi]\!](d) = \neg[\![\varphi]\!](d)$

where of course the functions $\wedge$, and so forth, on the right side are the Kleene three-valued functions.

Definition 2.4 *If $\phi \in L$, we define $Mod(\phi)$, the set of* positive models *of ϕ, to be the set*

$$\{d \in D \mid [\![\phi]\!](d) = 1\}.$$

There is a similar definition for the negative and undecided models.

Some well-known results for this system include the fact that every formula is equivalent (in the sense of having exactly the same positive and negative models) to a formula where negation signs appear only on atoms. Such a formula will be said to be in **negation normal form**. Such a NNF can be found using the de Morgan laws. Further, the set of positive models of ϕ is upward-closed in the ordering of partial truth assignments; in fact such a set is compact open in the Scott topology on D.

2.4 The Smyth powerdomain

Definition 2.5 *Let $(X, \sqsubseteq)$ be a Scott domain: a consistently complete partial order. An* antichain *is a finite nonempty set of pairwise incomparable compact elements of X.*

We take antichains as representatives of (finite) disjunctive structures. These have the usual preorder:

Definition 2.6 ***(Smyth preorder)*** *Let A and B be antichains on a domain $(X, \sqsubseteq)$. We say $A \sqsubseteq^\sharp B$ iff for all $e \in B$, there is some $d \in A$ with $d \sqsubseteq e$.*

The collection of finite antichains over a domain D is the set of compact elements of the Smyth powerdomain $P(D)$ of D. It is usually obtained by "ideal completion", but we prefer the following characterization. For this theorem, we define a subset X of D to be *saturated* if X is the intersection of all open sets U with $U \supseteq X$. For a proof of the result see [1].

Theorem 2.1 *The Smyth powerdomain $P^\sharp(X)$ of a Scott domain $(X, \sqsubseteq)$ is isomorphic to the domain consisting of all nonempty Scott-compact saturated subsets of X, ordered by reverse subset inclusion. The basis elements are the nonempty compact open subsets of X. Moreover, the domain $P^\sharp(X)^\top$ is obtained by adjoining the empty set as an extra compact set.*

For a finite set of propositional variables, the domain D is just a finite domain, and so the Smyth powerdomain is the finite set of nonempty upwards closed subsets of D, ordered by reverse inclusion.

We still have not introduced any default notions. This is our next order of business.

3 Defaults by example

To introduce our version of default logic, we use an example based on Gunter and Scott's introduction [10] to powerdomain theory.

Disjunctive fruit descriptions. Suppose I hold a bag of fruit, and I wish to give you information about what is in the bag. One such description might be " A fruit in the bag is a yellow fruit or a red fruit." This disjunctive description is based on two conjunctive items "yellow fruit" and "red fruit" A way to think of "yellow fruit" is as the partial function mapping the variables y to 1 and f to 1, and all other variables to an "undefined" element $\perp$. We will just write yf for such an assignment. These assignments are partially ordered pointwise as maps from the set of variables into $\{0, 1\}_\perp$, and form a Scott domain thereby. To model the disjunctive combination, we can therefore use the antichain $\{yf, rf\}$. Now the foregoing description is less informative than " A fruit in the bag is a yellow fruit or a cherry or a strawberry", and correspondingly we have

$$\{rf, yf\} \sqsubseteq^\sharp \{rfc, rfs, yf\},$$

where we have further qualified red fruit with cherry and strawberry features.

We next consider three-valued formulas. If ϕ is a propositional formula, and g is a partial truth assignment such that $g \models \phi$, then in Kleene's system, for any $f \sqsupseteq g$, we have $f \models \phi$. So, the set of satisfying partial truth assignments for a formula is the upward closure of the set of minimal satisfiers. And in general, any such upward-closed set is the set $Mod(\phi)$ of satisfiers of such a formula ϕ. Consider the set $MinMod(\phi)$ of minimal satisfiers of a formula. It is easy to check that $MinMod(\phi) \sqsubseteq^\sharp MinMod(\psi)$ iff $Mod(\phi) \supseteq Mod(\psi)$.

Now we turn to the specification of defaults. Our general theory of defaults in so-called Scott domains allows us to define (normal) default systems over any domain. In this paper, we want to define defaults in the Smyth powerdomain $P(D)$. We think that such defaults should be derived entities, arising from syntactic default statements. Suppose that we wish to say

(1) By default, a piece of fruit in the bag is a red fruit or a yellow fruit.

This should be a defeasible statement if there is a grape in the bag. Further, if we have solid information that there is nothing red in the bag, we still should be able to infer by default that there is a piece of yellow fruit.

In our approach, default assertions such as 1 will be modelled by *syntactic constraints* of the form $\tau \leadsto \theta$, where τ and θ are propositional formulas. Thus, (1) would read

$$f \leadsto (r \wedge f) \vee (y \wedge f).$$

Positive assertions of evidence are flat statements like $f \wedge \neg r \wedge \neg y$, coding the grape example, or just $\neg r$, coding the "no red things" assertion. (In general, these assertions are allowed to be disjunctive.) We might also want to state that there are no red things in the bag, just by default. This would provide another default constraint

$$\textbf{true} \leadsto \neg r.$$

In standard default logic, we represent constraints like the above as pseudo-proof rules of the form

$$\frac{\tau : \theta}{\theta}.$$

(We consider only normal defaults in this paper.) The meaning is that if we have the formula τ in our theory, and θ is consistent with our desired "final" theory, then we can add θ. Standard default logic in fact works with a Scott domain: the domain of propositional theories, ordered by set-theoretic inclusion. Our default semantics works in exactly the same way, only using the Smyth powerdomain. So normal defaults will be objects of the form

$$\frac{K : L}{L}$$

where K and L are upward-closed sets of partial truth assignments. We will insist, however, that these defaults (which might as well be pairs (K, L)) be derived from our syntactic default constraints.

We now show how to obtain defaults in the Smyth powerdomain from syntactic constraints. The difficulty to keep in mind is that these defaults should permit reasoning by cases. We have a two-stage process. The intermediate stage is a construct called a *semantic constraint.* In the example (4), we would create a pair

$$(f, \uparrow \{rf, yf\})$$

where now f is a truth assignment defined to be 1 only on the variable f, and $\uparrow \{rf, yf\}$ stands for the upward closure of the (similarly notated) truth assignments rf and yf. In such a constraint, the set on the right is a Smyth powerdomain element, and the element on the left is just a truth assignment. If we were working with a more general constraint $\tau \leadsto \theta$, we would create several semantic constraints of the form (d, θ) where d is a minimal satisfier of τ, and θ is used ambiguously to denote the upward closure of the set of minimal models of the syntactic formula θ.

The final stage in the creation of normal defaults is to lift semantic constraints so that the first element of a pair is a Smyth powerdomain element. We do this as follows. If (d, θ) is a semantic constraint, then we add a large number of defaults to our system. Take any K such that d subsumes a generator b of K. Consider the upward closure of all of the generators except b, and add in the set $(\uparrow \{b\}) \cap \theta$. This is called *updating* K by replacing b by $b \cap \theta$, and is notated $K[b \leftarrow b \cap \theta]$. Then our defaults are all pairs $(K, K[b \leftarrow b \cap \theta])$, where the second element of the pair is not actually empty.

This "update semantics" says that if we have a disjunctive description K at some stage, and some generator b of K entails a definite description d, then we can have all the disjunctive possibilities we had before, except that the description b can be improved by conjoining it with the description θ, where (d, θ) is a semantic constraint derived from our syntactic presentation. Notice that the updated K is a more informative description than K was.

In our example, suppose that we have the description $K = \uparrow \{rf, yf\}$. Consider the syntactic default **true** $\leadsto \neg r$. This gives rise to a semantic constraint $(\perp, \uparrow \{\neg r\})$, where $\perp$ is the totally undefined truth assignment. Since $\perp \sqsubseteq rf$, we can update rf in K. Thus $K[rf \leftarrow rf \cap \neg r] = \uparrow \{yf\}$. We could also update yf, deriving $\uparrow \{rf, yf\neg r\}$. Perhaps this would rule out orange fruit.

Now, we can turn the crank of Reiter's extension mechanism. Using our Smyth power defaults, we define extensions in exactly the same way that they were defined for default logic. We start with a powerdomain element representing the satisfiers of a positive evidential assertion ϕ. Then we "iterate" the defaults induced from Σ. What happens, in effect, is that instead of getting larger and larger theories, as in standard default logic, our ordering is reversed, and we get smaller and smaller sets of partial models. Thus the whole enterprise is really directed the same way as in standard default logic; the difference is that we have used partiality in the underlying domain, and have preserved it in the Smyth powerdomain. In the next subsection we show in general how to obtain extensions in general domains, and we recapitulate the formal definition of "update defaults" in the Smyth powerdomain.

3.1 Default domain theory

First, recall that if $(X, \sqsubseteq)$ is a Scott domain, we define a new domain $X^\top$ by adjoining a new element $\top$ to X, and put $x \sqsubseteq \top$ for $x \in X$.

Definition 3.1 *Let $(X, \sqsubseteq)$ be a Scott domain. A default set is a subset Δ of $\mathsf{K}(X) \times \mathsf{K}(X)$. We call a pair $(a, b) \in \Delta$ a (normal) default and think of it as a rule* $\dfrac{a : b}{b}$.

Example 3.1

Consider the domain of deductively closed sets of sentences with the inclusion order. Then a default consists of a pair $(\overline{\varphi}, \overline{\psi})$, where φ and ψ are sentences, and $\overline{\theta}$ is the deductive closure of the singleton set $\{\theta\}$. This corresponds exactly to the normal default rule

$$\frac{\varphi : \psi}{\psi}$$

in the sense of Reiter.

Definition 3.2 *The function $\eta(x, e)$ is the least element y of $X^\top$ such that (i) $x \sqsubseteq y$; and (ii) for any default $(a, b) \in \Delta$, if $a \sqsubseteq y$ and $b \uparrow e$, then $b \sqsubseteq y$. We call $e \in X$ an extension of $x \in X$ if $\eta(x, e) = e$.*

Clearly extensions over the domain of first-order theories are exactly Reiter's extensions, and in this case $\eta(W, E) = \Gamma(W, E)$ where Γ is Reiter's extension operator.

Reiter provides, for the first-order case, the following "iterative" characterization of the function η:

Proposition 3.1 $\eta(x, e) = \bigsqcup_i \eta(x, y, i)$, *where* $\eta(x, e, 0) = x$, *and*

$$\begin{aligned} \eta(x, e, i+1) &= \eta(x, e, i) \\ &\sqcup \bigsqcup \{b \mid (a, b) \in \Delta \;\&\; a \sqsubseteq \eta(x, e, i) \;\&\; b \uparrow e\} \end{aligned}$$

for all $i \geq 0$.

We now recapitulate "update semantics" in general for the Smyth powerdomain.

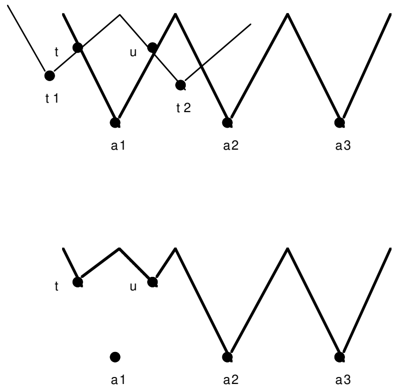

Figure 2. Updating $a1 \in A$ by the set θ.

Definition 3.3 *A semantic default constraint over the Smyth powerdomain $P^\sharp(X)$ of a Scott domain $(X, \sqsubseteq)$ is a pair (d, θ), where $d \in X$ and θ is a (possibly empty) compact open subset of X.*

The idea of this definition has been explained in the introductory remarks. In the setting of three-valued logic, these constraints are derived from a syntactic presentation Σ of defaults of the form $\tau \rightsquigarrow \theta$.

Some notation: if A is a nonempty compact open set, we let $\mu(A)$ be the finite set of generators of A. Thus $A = \uparrow \mu(A)$.

Definition 3.4 *Let A be a non-empty compact open subset of X. If θ is a compact open set, and $a \in \mu(A)$, we say that*

$$A[a \leftarrow a \cap \theta] = \uparrow (\mu(A) \setminus \{a\}) \cup (\uparrow a \cap \theta)$$

*is an **update** of A, provided the whole result is not empty.*

Example 3.2

A general picture of updating is presented in Figure 2. In this picture, the set A is the upward closure of the three element set $\{a1, a2, a3\}$. The set θ is indicated by lighter lines in the top half of the figure. It is generated by the set $\{t1, t2\}$. To update $a1$, we remove this element from the set of generators of A, and take the upward closure of $\{a1, a2\}$. This leaves the other two dark "cones" intact. Then we add back in the set $(\uparrow a1) \cap \theta$. This results in the two new generators t and u shown above $a1$ in the figure.

The next definition provides the actual lifting of default constraints to the powerdomain setting. Recall that a normal default set in any domain has to be a relation whose field is the set of finite elements of that domain.

Definition 3.5 *Let Γ be a set of default constraints over $P^\sharp(X)$. The default set Δ_Γ determined by Γ is the set of pairs*

$$\{(A, A[a \leftarrow a \cap \theta]) \\ \mid A \in \mathsf{K}(P^\sharp(X));\ (d, \theta) \in \Gamma,\ \textit{and}\ d \sqsubseteq a\}.$$

In the case of 3-valued logic, Γ is determined as in the remarks at the beginning of this section.

Finally, we can state a general notion of (skeptical) non-monotonic consequence over an arbitrary Scott domain, equipped with a default set Δ.

Definition 3.6 *Let d be a finite element and x an arbitrary element of a Scott domain $(X, \sqsubseteq)$. We say that x is a non-monotonic consequence of d, and write $d \mid\!\sim x$, if $x \sqsubseteq e$ for every Δ-extension e of d.*

Technically we should subscript $\mid\!\sim$ with the default set Δ, but the default set will always be clear from context.

3.2 SNIP

We first recast Definition 3.6 in the Smyth power ordering. Let Σ be a set of default constraints, and ϕ, ψ be formulas in L. The triple $S = (\phi, \Sigma, \psi)$ is called a *system*. The set V_S is the set of propositional variables appearing anywhere in S; D is the set of partial truth assignments over this finite set, and $P(D)$ is its Smyth powerdomain.

Definition 3.7 ***(Non-monotonic Consequence)*** *We say that $\phi \mid\!\sim_\Sigma \psi$ if (i) ϕ is unsatisfiable, or (ii) ϕ is satisfiable, and in $P(D)$, we have that $Mod(\psi) \supseteq E$ for any extension E of $Mod(\phi)$.*

We can now state our complexity problems and major result.

The Skeptical Non-monotonic Inference Problem (SNIP)

INSTANCE: A system $S = (\phi, \Sigma, \psi)$ with ϕ not containing any subformula **true**.

PROPERTY: $\phi \mid\!\sim_\Sigma \psi$.

The omission of the special formula **true** is for technical purposes; the full paper discusses the reasons for this, which have to do with its special status as the only way to introduce tautologies into 3-valued logic.

4 Characterizing extensions

Our NP(3) completeness result depends heavily on the structure of extensions in the Smyth powerdomain. This section is devoted to the analysis of that structure. A crucial role is played by the concept of a *safe extension*, and we look at this case first. Later, we find that non-safe extensions must be generated by singleton elements, and we exploit this to arrive at our final algorithms. (These appear in the next section.) Some proofs have been shortened or omitted for lack of space.

4.1 Safe extensions

Definition 4.1 *A partial truth assignment e is safe iff (i) $e \models \phi$, and (ii) for every syntactic constraint $\tau \rightsquigarrow \theta$ in Σ, if $e \models \tau$, then $e \models \theta$.*

Corollary 4.1 *The problem of determining, for a formula ϕ and set Σ of syntactic constraints, whether there is a safe element with respect to these, is NP-complete.*

Proof. To check whether there is a safe element, we guess a partial truth assignment e. Then, by Definition 4.1, we can check if e is safe in polynomial time. The hardness proof is a simple reduction from CNF-SAT.

□

Definition 4.2 *An extension E of a compact open set $M \in P(D)$ is said to be safe if E contains a safe element e.*

We want to show that if an extension is safe, then all of its generators are safe elements. To do this we need the definition of the extension operator stated for the Smyth powerdomain ordering.

Lemma 4.1 *The set $\eta(M, W)$ is the inclusion-largest set Y such that (i) $M \supseteq Y$; and (ii) whenever $(K, K[a \leftarrow a \cap \theta])$ is a default with $K \supseteq Y$ and $K[a \leftarrow a \cap \theta] \cap W \neq \emptyset$, then $Y = K[a \leftarrow a \cap \theta] \cap Y$.*

We refer to (ii) in Lemma 4.1 as W*-invariance* of Y under W-consistent rule application, or simply W-invariance.

Lemma 4.2 *If E is a safe extension determined by a system of constraints, then every generator g of E is safe.*

Proof sketch: Show that if a generator is unsafe, then Lemma 4.1 is violated.

□

Lemma 4.3 *If there is a safe element e under a system of constraints, then for any powerdomain element W, we have $e \in \eta(M, W)$.*

Proof sketch: Use Proposition 3.1, the "iterative" characterization of Reiter's extension operator, and induction.

□

Corollary 4.2 *If there is a safe element e under a system of constraints, then the set M of satisfiers of (a satisfiable) ϕ has a unique extension, which is safe.*

Proof sketch. A general fact about extensions in any Scott domain is that distinct extensions must be inconsistent. In the Smyth order, if E and F are inconsistent, then they have an empty intersection. This contradicts the previous lemma.

□

The preceding lemmas lead us to our first important theorems.

Theorem 4.1 ***(Dichotomy Theorem).*** *For a system of constraints Σ and satisfiable formula ϕ, either there is a unique safe extension E of $M = Mod(\phi)$, or no extensions are safe.*

Theorem 4.2 *It is an NP-complete problem to decide if a system has a (unique) safe extension.*

4.2 Non-safe extensions

Now we turn to the analysis of non-safe (we also say unsafe) extensions. This is a little more subtle than the safe case, but also a little more surprising. We remind the reader that throughout this section, we will assume that our starting formula ϕ is not equivalent to **true**.

Our first result is easy, but a bit unexpected. It uses Lemma 4.1.

Lemma 4.4 *If E is a non-safe extension of M, then $E = \uparrow g$ for a single element g of D.*

Proof. Suppose E has at least two generators g and h. Then since these generators are not safe, one of them can be properly updated, and the result will still be E-consistent (we do not update the other). This violates invariance of E under E-consistent rule application.

□

Because of this result, we can turn attention to criteria for deciding if a system has a "singleton" extension $\uparrow g$.

Refer again to the conditions in Lemma 4.1. We are interested in $W = \uparrow g$, and we seek a characterization of the condition $\eta(M, \uparrow g) = \uparrow g$. To get at this, say that a set Y is *g-invariant* if it satisfies item (ii) of Lemma 4.1, taking $W = \uparrow g$.

Lemma 4.5 *Assume that there are no safe elements. Let $g \in M$, and assume M is not $\uparrow \perp$. Then the set $\uparrow g$ is g-invariant if and only if g is a maximal element of the finite domain D.*

Proof. Assume first that there are two or more variables in V_S. Let $g \in M$ be an (unsafe) element which is not maximal. We have also that $g \neq \perp$, because $g \in M$. Since g is unsafe, there is a constraint (d, θ) (derived from some syntactic constraint $\tau \rightsquigarrow \theta$) with $d \sqsubseteq g$, but $g \notin \theta$. Since g is not maximal, and not $\perp$, then at least one variable x must be set to a 0 or 1 value in g, and another variable y to $\perp$. Define a $g' \sqsupset g$ by setting y to 0 (1 would also do); and then let a be obtained from g' by setting x to $\perp$. Then the set $K = \{a, g\}$ has two incomparable generators, and contains g. The set $K[g \leftarrow g \cap \theta]$ furthermore is consistent with $\uparrow g$, and properly updates $\uparrow g$. Thus $\uparrow g$ is not g-invariant.

Conversely, suppose that g is maximal in M. Then clearly $\{g\}(=\uparrow g)$ is g-invariant because if $K[b \leftarrow b \cap \theta] \cap \{g\} \neq \emptyset$ then $g \in K[b \leftarrow b \cap \theta]$.

It remains to consider the case where there is only one propositional variable p in V_S. But then all elements which are not $\perp$ are maximal.

□

At this point, in analyzing possibilities for unsafe extensions, we know that if $\phi \neq$ **true**, then any extension of $M = Mod(\phi)$ must be a singleton maximal point $g \models \phi$. We now analyze the condition that such a g-invariant set $\{g\}$ is the largest g-invariant Y with $Y \subseteq M$.

Lemma 4.6 *Assume there are no safe elements, and let $g \in M \neq (\uparrow \perp)$ be maximal and $Y \subseteq M$ be g-invariant. Then Y is not a subset of $\{g\}$ if and only if all generators of Y strictly subsume g.*

Proof. Assume Y is as stated, with a generator y_0 not subsuming g. We show that y_0 must be safe.

Consider a constraint $\tau \rightsquigarrow \theta$ with $y_0 \models \tau$. Let $K = Y \cup \{g\}$. We claim that K is still g-invariant. This is clear if $g \in Y$. If not, then we have $\{g\} \cap Y = \emptyset$. So any updates whose result is g-consistent have to apply either to generators of Y or to g itself. But since g is maximal, we have that $\{g\}$ is g-invariant. So these updates cannot affect g either. This establishes the claim.

Now consider the generator y_0 of K. The set $K[y_0 \leftarrow y_0 \cap \theta]$ is certainly consistent with $\{g\}$. But since $K = Y \cup \{g\}$ is g-invariant, we have $y_0 \models \theta$. So y_0 is safe. The converse is trivial.

□

The next lemma reduces the search for g-invariant sets even more.

Lemma 4.7 *Under the conditions stated above, if Y is a g-invariant set, then it must be singleton-generated.*

Proof. If Y is a subset of $\{g\}$, then there is nothing to prove. So suppose $Y \not\subseteq \{g\}$. By the previous lemma, all generators of Y must (strictly) subsume g. Suppose there are more than one generators for Y. Fix one of the generators y_0 of Y. Since y_0 is unsafe, there is some $\tau \leadsto \theta \in \Sigma$ such that $y_0 \models \tau$, but $y_0 \notin \theta$. Now applying $(Y, Y[y_0 \leftarrow y_0 \cap \theta])$ to Y (note that g is still a member of $Y[y_0 \leftarrow y_0 \cap \theta]$), and we get a proper update – a contradiction.

□

Our next lemma is the real surprise.

Lemma 4.8 *If there is no safe element, and $\{g\}$ is a g-invariant set with g maximal and $g \in M \neq (\uparrow \bot)$, then there is no $y_0 \sqsubset g$ and $y_0 \in M$ such that $\uparrow y_0$ is g-invariant.*

Proof. Let $y_0 \in M$, and $y_0 \sqsubset g$. Once again we have that $y_0 \neq \bot$. Since y_0 is not safe, there is a constraint (d, θ) with $d \sqsubseteq y_0$, such that $y_0 \notin (\uparrow y_0 \cap \theta)$. Choose another element $r \in D$ with $r \sqsubseteq g$ but r incomparable with y_0. Since g is maximal and $y_0 \neq \bot$, this is certainly possible. Consider the default pair

$$(\{r, y_0\}, \{r, y_0\}[y_0 \leftarrow y_0 \cap \theta]).$$

The second element of this pair still contains g because $r \sqsubseteq g$. But the default properly updates y_0.

□

Theorem 4.3 *Assume that there are no safe elements for the system (ϕ, Σ, ψ). If $\bot \not\models \phi$ (i.e., ϕ is not equivalent to **true**), then the extensions of $M = Mod(\phi)$ are those singleton sets $\{g\}$ such that g is maximal and $g \models \phi$.*

Proof. The result follows immediately from Lemma 4.8.

□

5 Algorithms and completeness

Theorem 5.1 *SNIP is in co-NP(3).*

Proof. We present an NP(3) algorithm for the complementary problem. This algorithm will accept those systems $S = (\phi, \Sigma, \psi)$ such that it is not the case that $\phi \mid\!\sim_\Sigma \psi$. This happens if and only if (i) ϕ is satisfiable and (ii) there is an extension E of $M = Mod(\phi)$ not contained in $Mod(\psi)$. To do this it only has to verify that some generator of E does not satisfy ψ.

Given a system $S = (\phi, \Sigma, \psi)$, our algorithm carries out the following steps.

1. Use – as an oracle – the nondeterministic polynomial algorithm that, determines for S, whether or not there is a safe element e with $e \in M = Mod(\phi)$. This just uses Def. 4.1. If the answer returned is "yes", go to step 2, else to step 3.

2. Using Def. 4.1 once again, guess an element e, verify that it is safe (this entails satisfying ϕ), and further does not satisfy ψ. If this branch of the algorithm succeeds, then there is a safe extension with a generator not satisfying ψ, since the element e must be above a generator of a safe extension. If the generator satisfied ψ, then e would too, by persistence. Conversely, if there is a safe extension, with a generator not satisfying ψ, we know by Lemma 4.2 that every generator is safe. Our algorithm guesses all possible safe elements in $M = Mod(\phi)$, so it must succeed on such a generator. Note that it does not have to check that what it has found is actually a generator!

3. At this point, we know that there are no safe extensions. We now check for the existence of a non-safe extension (necessarily of the form $\{g\}$) such that $g \not\models \psi$. Given the triple (ϕ, Σ, ψ), we continue with the following steps.

4. First, nondeterministically guess a maximal element g of $D(V_S)$.

5. Check that $g \models \phi$. If not, fail. Otherwise go to the next step.

6. When this step is reached, we know that g is an extension of M. Check that $g \not\models \psi$. If so, succeed, otherwise fail.

□

The hardness proof uses a standard complete language for co-NP(3):

$$\text{co-SAT}(3) = \{(P, Q, R) : \\ (P \in \text{SAT} \;\&\; Q \in \text{TAUT}) \;\text{ or }\; R \in \text{TAUT}\},$$

where SAT stands for the collection of satisfiable boolean expressions, and TAUT stands for the collection of tautologies in the classical two-valued case.

Theorem 5.2 (Co-NP(3) Completeness) *SNIP is co-NP(3)-complete.*

Proof sketch. We only have to establish that the problem is hard for co-NP(3). Given any boolean expression triple (P, Q, R), we construct an instance of skeptical non-monotonic consequence as follows. First, *rename* the boolean variables so that P, Q and R do not share common variables.

Now add two default constraints

$$\Sigma = \{\textbf{true} \rightsquigarrow P, \textbf{true} \rightsquigarrow \bigwedge_{i \in I}(x_i \vee \neg x_i)\},$$

where $\{x_i \mid i \in I\}$ is the set of all variables appearing in P, Q, or R (after the renaming of variables).

We can then show that $P \in$ SAT and $Q \in$ TAUT, or $R \in$ TAUT if and only if $\neg R \mid\!\sim_\Sigma P \wedge Q$.

□

6 Further theorems

Here are some special types of systems for which the safety test is simple; so our overall complexity is reduced.

Definition 6.1 *A system (ϕ, Σ, ψ) is **conjunctive** iff the formula ϕ and all of the formulas θ appearing as default conclusions contain no occurrences of $\vee$.*

Theorem 6.1 *There is a polynomial-time algorithm to decide whether or not a conjunctive system is safe.*

The proof shows how to inductively construct a (singleton) safe extension if there is one.

□

Corollary 6.1 *The skeptical entailment problem for conjunctive systems is in co-NP.*

Corollary 6.2 *The previous corollary holds for systems which are conjunctive, except that the starting formula is allowed to be a disjunction of conjunctions.*

Corollary 6.3 *For a conjunctive system, the question of whether the system is safe and $\phi \mid\!\sim \psi$ can be answered in polynomial time.*

Corollary 6.4 *If in a conjunctive system (ϕ, Σ, ψ), the formula ψ is in DNF and is not equal to **true**, and ϕ is in DNF, then the skeptical inference problem is in P.*

We close the section with a final theorem, which was discovered as a by-product of our work on complexity. The Cautious Monotony law is one of the patterns of inference said to form a "core" collection of rules for any reasonable logical non-monotonic reasoning system. It reads as follows, where now $\mid\!\sim$ is some sort of general relation between formulas.

If $\phi \mid\!\sim \psi$ and $\phi \mid\!\sim \gamma$, then $\phi \wedge \gamma \mid\!\sim \psi$.

Theorem 6.2 *The CM law holds (with $\mid\!\sim$ taken to be $\mid\!\sim_\Sigma$) as long as ϕ is not equivalent to $\mathbf{true}$.*

The proof is by the Dichotomy Theorem, analazying the safe and unsafe cases separately. As we have mentioned, the Or law (reasoning by cases) is non-trivial; the proof appears elsewhere.

7 Conclusion

The results reported here show that domain theory can be a useful tool in the investigation of phenomena not directly arising from programming languages. There are, however, many connections between non-monotonic reasoning and logic programming. It seems that powerdomain techniques shold be considered to see if they provide a foundation for disjunctive (non-deterministic) logic programs. In this way, one might understand more about negation as failure.

We should also mention that our own techniques apply to other domains beside that of propositional logic. First-order domains can be defined [19], for example. We need to extend these complexity studies to the first-order case.

Our complexity results are only for normal defaults; more general kinds of defaults should be investigated. Finally, we have not as yet implemented our semantics, though work is under way on such a project. We hope that our results will in fact enable a feasible and useful implementation of propositional non-monotonic reasoning.

References

[1] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Clarendon Press, 1994.

[2] H. Aït-Kaci. *A Lattice-Theoretic Approach to Computation based on a Calculus of Partially Ordered Type Structures*. PhD thesis, University of Pennsylvania, 1984.

[3] P. Buneman. Functional programming and databases. In D. Turner, editor, *Research Topics in Functional Programming*, pages 155–169. Addison-Wesley, 1990.

[4] J. Y. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and K. Wechsung. The boolean hierarchy. i: Structural properties. *SIAM Journal on Computing*, 17:1232–1252, 1988.

[5] J. Y. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and K. Wechsung. The boolean hierarchy. ii: Applications. *SIAM Journal on Computing*, 18:95–111, 1989.

[6] M. Gelfond, V. Lifschitz, H. Przymuzińska, and M. Truszczyński. Disjunctive defaults. In *Proceedings of Second Annual Conference on Knowledge Representation*, pages 230–237. Morgan Kaufmann, 1991.

[7] G. Gottlob. Complexity results for non-monotonic logics. *J. Logic Comput*, 2(3):397–425, 1992.

[8] G. Gottlob. The complexity of reasoning under the stationary fixed point semantics. *Information and Computation*, 121(1), 1995.

[9] C. A. Gunter, T-H. Ngair, P. Panangaden, and D. Subramanian. The common order-theoretic structure of version spaces and atms's. In *Ninth National Conference on Artificial Intelligence*, pages 500–505. AAAI Press, 1991.

[10] C. A. Gunter and D. S. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics*, chapter 12, pages 633–674. Elsevier, 1990.

[11] H. Kautz and B. Selman. Hard problems for simple default logics. In *Proceedings of KR 89*, pages 189–197. Morgan Kaufmann, 1989.

[12] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models, and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[13] L. Libkin. *Aspects of Partial Information in Databases*. PhD thesis, University of Pennsylvania, 1994.

[14] W. Marek, A. Nerode, and J. Remmel. A theory of nonmonotonic rule systems. In *Proceedings of 5th IEEE Symposium on Logic in Computer Science*, pages 79–94, 1990.

[15] F. Pereira and S. Shieber. The semantics of grammar formalisms seen as computer languages. In *Proceedings of 10th International Conference on Computational Linguistics: COLING 84*, 1984.

[16] C. Pollard and M. A. Moshier. Unifying partial descriptions of sets. In P. Hansen, editor, *Vancouver Studies in Cognitive Science: vol. I*. University of British Columbia Press, 1990.

[17] H. Przymusinska and T. Przymusinski. Stationary default extensions. *Fund. Informat.*, 21(12):67–87, 1994.

[18] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[19] W. Rounds and G. Q. Zhang. Logical considerations on default semantics. To appear,*Annals of Mathematics and Artificial Intelligence*, 1997.

[20] J. Stillman. The complexity of propositional default logics. In *Proceedings of AAAI 1992*, pages 794–799, 1992.