# Towards Nonmonotonic Reasoning on Hierarchical Knowledge

Pascal Hitzler

Artificial Intelligence Institute, Department of Computer Science
Dresden University of Technology, Dresden, Germany
phitzler@inf.tu-dresden.de, www.wv.inf.tu-dresden.de/~pascal/

Technical Report WV–02–09

**Abstract**

W.C. Rounds and G.-Q. Zhang have recently proposed to study a form of disjunctive logic programming generalized to algebraic domains [RZ01]. This system allows reasoning with information which is hierarchically structured and forms a (suitable) domain. We extend this framework to include reasoning with *negative information*, i.e. the implicit or explicit *absence* of bits of information. These investigations will naturally lead to a form of default reasoning which is strongly related to logic programming with answer sets or stable models, which has recently created much interest amongst artificial intelligence researchers concerned with knowledge representation and reasoning.

## 1 Introduction

In [RZ01], Rounds and Zhang propose to study a form of clausal logic generalized to algebraic domains, in the sense of domain theory [SHLG94, AC98]. In essence, they propose to interpret finite sets of compact elements as clauses, and develop a theory which links corresponding logical notions to topological notions on the domain. Amongst other things, they establish a sound and complete resolution rule and a form of disjunctive logic programming in domains, based on material implication. A corresponding semantic operator turns out to be Scott-continuous.

In this report, we will extend this paradigm to include reasoning with default negation. We are motivated by the gain in expressiveness through the use of negation in artificial intelligence paradigms related to nonmonotonic reasoning. This approach, using ideas from default logic [Rei80], treats negation as a meta-logical supplement: The negation of an item is believed if there is no reason to believe the item itself. This perspective on negation has recently led to the development of applications in the form of nonmonotonic reasoning systems known as *answer set programming* (cf. [Lif99, MT99] for accounts of this).

The paper is structured as follows. In Section 2 we review and study Rounds' and Zhang's *clausal logic and resolution in algebraic domains*, as laid out in [RZ01]. We will also provide an equivalent system which is simpler and easier to work with. In Section 3, we move on to study logic programming in algebraic domains as proposed in [RZ01]. In particular, we will adjoin a form of default negation to this programming paradigm. The exposition will naturally lead to establishing a form of well-founded semantics for these programs, as well as a notion of stable models, both very strongly related to their counterparts in the classical logic programming paradigm.

The emphasis of this report is to propose definitions and constructions, not to present deep results, although we have taken care to include just enough examples and results as to justify the constructions. Proofs have been omitted, and can be found in [Hit02], which is available from the author's web page.

*Acknowledgements.* I'm very grateful for discussions with Achim Jung, William C. Rounds, Anthony K. Seda, Pawel Waszkiewicz, and Guo-Qiang Zhang, some of which took place at a very pleasant stay at Schloss Dagstuhl, during Seminar 02221, in May 2002.

## 2 Clausal Logic and Resolution in Algebraic Domains

A *partially ordered set* is a pair $(D, \sqsubseteq)$, where $D$ is a nonempty set and $\sqsubseteq$ is a reflexive, antisymmetric, and transitive relation on $D$. A subset $X$ of a partially ordered set is *directed* if for all $x, y \in X$ there is $z \in X$ with $x, y \sqsubseteq z$. An *ideal* is a directed and downward closed set. A *complete partial order*, *cpo* for short, is a partially ordered set $(D, \sqsubseteq)$ with a least element $\bot$, called the *bottom element* of $(D, \sqsubseteq)$, and such that every directed set in $D$ has a least upper bound, or supremum, $\bigsqcup D$. An element $c \in D$ is said to be *compact* or *finite* if whenever $c \sqsubseteq \bigsqcup L$ with $L$ directed, then there exists $e \in L$ with $c \sqsubseteq e$. The set of all compact elements of a cpo $D$ is written as $\mathsf{K}(D)$. An *algebraic cpo* is a cpo such that every $e \in D$ is the directed supremum of all compact elements below it. For $a, b \in D$ we write $a \not\uparrow b$ if $a$ and $b$ are *inconsistent*, i.e. if there does not exist a common upper bound of $a$ and $b$.

A set $U \subseteq D$ is said to be *Scott open*, or just *open*, if it is upward closed and for any directed $L \subseteq D$ we have $\bigsqcup L \in U$ if and only if $U \cap L \neq \emptyset$. The *Scott topology* on $D$ is the topology whose open sets are all Scott open sets. An open set is *compact open* if it is compact in the Scott topology. A *coherent algebraic cpo* is an algebraic cpo such that the intersection of any two compact open sets is compact open. We will not make use of many topological notions in the sequel. So let us just note that coherency of an algebraic cpo implies that the set of all minimal upper bounds of a finite number of compact elements is finite, i.e. if $c_1, \ldots, c_n$ are compact elements, then the set $\mathsf{mub}\{c_1, \ldots, c_n\}$ of minimal upper bounds of these elements is finite. As usual, we set $\mathsf{mub}\,\emptyset = \{\bot\}$, where $\bot$ is the least element of $D$.

In the following, $(D, \sqsubseteq)$ will always be assumed to be a coherent algebraic cpo. We will also call these spaces *domains*. All of the above notions, except perhaps that of coherency, are standard and can be found e.g. in [SHLG94, AC98].

**2.1 Definition** Let $D$ be a coherent algebraic cpo with set $\mathsf{K}(D)$ of compact elements. A *clause* is a finite subset of $\mathsf{K}(D)$. We denote the set of all clauses over $D$ by $\mathcal{C}(D)$. If $X$ is a clause and $w \in D$, we write $w \models X$ if there exists $x \in X$ with $x \sqsubseteq w$, i.e. $X$ contains an element below $w$. A *theory* is a set of clauses, which may be empty. An element $w \in D$ is a *model* of a theory $T$, written $w \models T$, if $w \models X$ for all $X \in T$ or, equivalently, if every clause $X \in T$ contains an element below $w$. A clause $X$ is called a *logical consequence* of a theory $T$, written $T \models X$, if $w \models T$ implies $w \models X$. If $T = \{E\}$, then we write $E \models X$ for $\{E\} \models X$. Note that this holds if and only if for every $w \in E$ there is $x \in X$ with $x \sqsubseteq w$. For two theories $T$ and $S$, we say that $T \models S$ if $T \models X$ for all $X \in S$. In order to avoid confusion, we will throughout denote the empty clause by $\{\}$, and the empty theory by $\emptyset$. A theory $T$ is *closed* if $T \models X$ implies $X \in T$ for all clauses $X$. It is called *consistent* if $T \not\models \{\}$ or, equivalently, if there is $w$ with $w \models T$.

A main technical result from [RZ01], where the notions from Definition 2.1 were introduced, shows that the set of all consistent closed theories over $D$, ordered by inclusion, is isomorphic to the collection of all non-empty Scott-compact saturated subsets of $D$, ordered by reverse inclusion. This result rests on the Hofmann-Mislove theorem, and we refer the reader to [RZ01] for details. It follows as a corollary that a theory is logically closed if and only if it is an ideal,[1] and also that a clause is a logical consequence of a theory $T$ if and only if it is a logical consequence of a finite subset of $T$. The latter is a compactness theorem for clausal logic in algebraic domains.

**2.2 Example** In [RZ01], the following running example was given. Consider Kleene's strong three-valued logic in the propositional case[2], with the usual (knowledge)-ordering on the set $\mathbb{T} = \{\mathbf{f}, \mathbf{u}, \mathbf{t}\}$ of truth values given by $\mathbf{u} \leq \mathbf{f}$ and $\mathbf{u} \leq \mathbf{t}$. This induces a pointwise ordering on the space $\mathbb{T}^{\mathcal{V}}$ of all interpretations (or *partial truth assignments*), where $\mathcal{V}$ is the (countably infinite) set of all propositional variables in the language under consideration. The partially ordered set $\mathbb{T}^{\mathcal{V}}$ is a coherent algebraic cpo[3]. Compact elements in $\mathbb{T}^{\mathcal{V}}$ are those interpretations which map all but a finite number of propositional variables to $\mathbf{u}$. We denote compact elements by strings such as $pq\overline{r}$, which indicates that $p$ and $q$ are mapped to $\mathbf{t}$ and $r$ is mapped to $\mathbf{f}$.

We note that $\{e \mid e \models \phi\}$ is upward-closed for any formula $\phi$. A clause in $\mathbb{T}^{\mathcal{V}}$ is a formula in disjunctive normal form, e.g. $\{pq\overline{r}, \overline{p}q, r\}$ translates to $(p \wedge q \wedge \neg r) \vee (\neg p \wedge q) \vee r$.

In [RZ01], a sound and complete resolution rule, called *clausal hyperresolution*, was given as follows, where $\{X_1, \dots, X_n\}$ is a clause set and $Y$ a clause, and $\mathsf{mub}\{a_1, \dots, a_n\}$ denotes the set of all minimal upper bounds of all the $a_i$'s, which is a finite set of compact elements by algebraic coherence, i.e. a clause.

$$\frac{X_1 \quad X_2 \quad \dots \quad X_n; \qquad a_i \in X_i \text{ for } 1 \leq i \leq n; \qquad \mathsf{mub}\{a_1, \dots, a_n\} \models Y}{Y \cup \bigcup_{i=1}^{n} (X_i \setminus \{a_i\})} \qquad (\mathsf{hr})$$

This rule is sound in the following sense: Whenever $w \models X_i$ for all $i$, then for any admissible choice of the $a_i$ and $Y$ in the antecedent, we have $w \models Y \cup \bigcup_{i=1}^{n} (X_i \setminus \{a_i\})$.

For completeness, it is necessary to adjoin to the above clausal hyperresolution rule a special rule which allows the inference of any clause from the empty clause. We indicate this rule as follows.

$$\frac{\{\}; \qquad Y \in \mathcal{C}(D)}{Y} \qquad (\mathsf{spec})$$

With this addition, given a theory $T$ and a clause $X$ with $T \models X$, we have that $T \vdash^* X$, where $\vdash^*$ stands for a finite number of applications of the clausal hyperresolution rule together with the special rule.

Furthermore, [RZ01, Remark 4.6] shows that binary hyperresolution, together with (spec), is already complete, i.e. the system consisting of the *binary clausal hyperresolution* rule

$$\frac{X_1 \quad X_2; \qquad a_1 \in X_1 \quad a_2 \in X_2; \qquad \mathsf{mub}\{a_1, a_2\} \models Y}{Y \cup (X_1 \setminus \{a_1\}) \cup (X_2 \setminus \{a_2\})} \qquad (\mathsf{bhr})$$

together with the special rule is sound and complete.

[1] An ideal with respect to the *Smyth preorder* $\sqsubseteq^\sharp$, where $X \sqsubseteq^\sharp Y$ if and only if for every $y \in Y$ there exists some $x \in X$ with $x \sqsubseteq y$.

[2] Cf. [Fit85] for a discussion of this in the context of logic programming semantics and [Plo78] for a domain-theoretic context.

[3] In fact it is also consistently complete.

If the set $\{a_1,\dots,a_n\}$ is inconsistent, then $\mathsf{mub}\{a_1,\dots,a_n\} = \{\}$. Since $\{\} \models \{\}$, clausal hyperresolution generalizes the usual notion of resolution, given by the following rule.

$$\frac{X_1 \quad X_2; \qquad a_1 \in X_1 \quad a_2 \in X_2; \qquad a_1 \not\uparrow a_2}{(X_1 \setminus \{a_1\}) \cup (X_2 \setminus \{a_2\})} \qquad (\mathsf{r})$$

It is our desire to give a sound and complete system which is as simple as possible. Consider the following rule, which we call *simplified hyperresolution*. It is easy to see that it is an instance of $(\mathsf{hr})$ and more general than $(\mathsf{r})$.

$$\frac{X_1 \quad X_2; \qquad a_1 \in X_1 \quad a_2 \in X_2}{\mathsf{mub}\{a_1,a_2\} \cup (X_1 \setminus \{a_1\}) \cup (X_2 \setminus \{a_2\})} \qquad (\mathsf{shr})$$

Also note the following rules, the first of which is a special instance of the clausal hyperresolution rule, while the second can be obtianed using $(\mathsf{hr})$ and $(\mathsf{spec})$. We call them the *weakening rule* and the *extension rule*, respectively.

$$\frac{X; \qquad a \in X; \qquad y \sqsubseteq a}{\{y\} \cup (X \setminus \{a\})} \quad (\mathsf{w}), \qquad \frac{X; \qquad y \in \mathsf{K}(D)}{\{y\} \cup X} \quad (\mathsf{ext})$$

Indeed, the first rule follows from $(\mathsf{hr})$ since $a \in X$ and $\{a\} \models \{y\}$. The second rule follows from $(\mathsf{hr})$ for $X \neq \{\}$, since $\{a\} \models \{a, y\}$ for all $y \in \mathsf{K}(D)$, and from $(\mathsf{spec})$ for $X = \{\}$.

**2.3 Theorem** The system consisting of $(\mathsf{shr})$, $(\mathsf{ext})$ and $(\mathsf{w})$ is complete.

Note that resolution $(\mathsf{r})$ together with $(\mathsf{ext})$ and $(\mathsf{w})$ is not complete. In order to see this, we refer to Example 2.2. Let $T = \{\{p\},\{q\}\}$ and $X = \{pq\}$. Then $T \models X$ but there is no way to produce $X$ from $T$ using $(\mathsf{r})$, $(\mathsf{w})$ and $(\mathsf{ext})$ alone. Indeed, it is easy to show by induction that any $X$ which can be derived from $T$ by using only $(\mathsf{r})$, $(\mathsf{w})$ and $(\mathsf{ext})$, contains either $p$ or $q$, which suffices.

We call the system consisting of the rules $(\mathsf{w})$, $(\mathsf{ext})$ and $(\mathsf{shr})$ the RAD system, from *Resolution in Algebraic Domains*. We interpret the RAD rules in the setting of Example 2.2. We already know that clauses correspond to formulas in disjunctive normal form (DNF), and theories to sets of DNF formulas. The weakening rule acts on single clauses and replaces a conjunction contained in a DNF formula by a conjunction which contains a subset of the propositional variables contained in the original conjunction, e.g. $(p \wedge q) \vee r$ becomes $p \vee r$. The extension rule disjunctively extends a DNF formula by a further conjunction of propositional variables, e.g. $(p \wedge q) \vee r$ becomes $(p \wedge q) \vee r \vee (s \wedge q)$. The simplified hyperresolution rule finally takes two DNF formulas, deletes one conjunction from each of them, and forms a disjunction from the resulting formulas together with the conjunction of the deleted items, e.g. $(p \wedge q) \vee r$ and $\neg p \vee (s \wedge r)$ can be resolved to $(p \wedge q) \vee (r \wedge \neg p) \vee (s \wedge r)$.

A more abstract interpretation of the RAD system comes from a standard intuition underlying domain theory. Elements of the domain $D$ are interpreted as pieces of information, and if $x \sqsubseteq y$, this represents that $y$ contains more information than $x$. Compact elements are understood as items which are computationally accessible. From this point of view, RAD gives a calculus for reasoning about disjunctive information in computation, taking a clause, i.e. a finite set of computationally accessible information items as disjunctive knowledge about these items. The rules from RAD yield a system for deriving further knowledge from the given disjunctive information. The weakening rule states that we can replace an item by another one which contains less information. The extension rule states that we can always extend our knowledge disjunctively with further bits of information. Both rules decrease our knowledge.

The simplified hyperresolution rule states that we can disjunctively merge two collections of disjunctive information, while strengthening our knowledge by replacing two of the items from the collections by an item which contains both pieces of information, and deleting the original items.

The RAD system alone already deserves a more thorough study. It provides a framework for reasoning with disjunctive information on a lattice which encodes background knowledge. The system, however, can be extended naturally by a disjunctive logic programming paradigm, as presented in the next section.

## 3 Logic Programming in Algebraic Domains

Following the lead from [RZ01], we now move on to study disjunctive logic programming in algebraic domains. Our aim is to extend this paradigm with a kind of default negation.

**3.1 Definition** A (*disjunctive logic*) *program* over $D$ is a set $P$ of *rules* of the form $Y \leftarrow X$, where $X, Y$ are clauses over $D$. An element $e \in D$ is said to be a *model* of $P$ if for every rule $Y \leftarrow X$ in $P$, if $e \models X$, then $e \models Y$. A clause $Y$ is a *logical consequence* of $P$ if every model of $P$ satisfies $Y$. We write $\mathsf{cons}(P)$ for the set of all clauses which are logical consequences of $P$. If $T$ is a theory, we write $\mathsf{cons}(T)$ for the set of all clauses which are logical consequences of $T$. Note that the notions of logical consequence are substantially different for theories and programs.

The following (*clause*) *propagation*[4] *rule*, denoted by $\mathsf{CP}(P)$, for given program $P$, was studied in [RZ01].

$$\frac{X_1 \quad \ldots \quad X_n; \quad a_i \in X_i \quad (i = 1, \ldots, n); \quad Y \leftarrow Z \in P; \quad \mathsf{mub}\{a_1, \ldots, a_n\} \models Z}{Y \cup \bigcup_{i=1}^{n} (X_i \setminus \{a_i\})}$$

Applying this rule, we say that $Y \cup \bigcup_{i=1}^{n} (X_i \setminus \{a_i\})$ is a $\mathsf{CP}(P)$-consequence of a theory $T$ if $X_1, \ldots, X_n \in T$. The following operator is based on the notion of $\mathsf{CP}(P)$-consequence and acts on logically closed theories. Let $T$ be a logically closed theory over $D$ and let $P$ be a program and define

$$\mathcal{T}_P(T) = \mathsf{cons}\left(\{Y \mid Y \text{ is a } \mathsf{CP}(P)\text{-consequence of } T\}\right).$$

In [RZ01], it was shown that $\mathcal{T}_P$ is a Scott-continuous function on the space of all logically closed theories, i.e. has a least fixed point $\mathsf{fix}(\mathcal{T}_P)$. It was also shown that $\mathsf{fix}(\mathcal{T}_P) = \mathsf{cons}(P)$.

### 3.1 Inference of Negative Information

Using the notation of Example 2.2, consider the program $P$ consisting of the following rules. This program is in fact a propositional version of the well-known *even numbers* program, which can be found e.g. in [Fit94] or [HS0x].

$$\begin{aligned} \{p_0\} &\leftarrow \{\bot\} \\ \{p_{n+1}\} &\leftarrow \{\overline{p_n}\} \qquad \text{for all } n \in \mathbb{N}. \end{aligned}$$

Note that $\mathsf{cons}(\emptyset) = \mathsf{cons}(\{\{\bot\}\})$, so we obtain $\mathsf{cons}(P) = \mathsf{fix}(\mathcal{T}_P) = \mathsf{cons}(\{\{p_0\}\})$. If we understand $P$ as a logic program in the classical sense, however, then all major approaches to

[4]This rule was called the *hyperresolution rule determined by P* in [RZ01]. We prefer the notion *propagation* since in our opinion resolution, when talking about programs, is better thought of as a process which yields the antecedent from a given consequent.

declarative semantics, e.g. the *Clark completion semantics* [Cla78] (also known as the *supported model semantics*), the *Fitting semantics* [Fit85] (also called *Kripke-Kleene semantics*), the *perfect model semantics* [Prz88], the *stable model semantics* [GL88] (also called *answer set semantics*, which is motivated by default logic), and the *well-founded semantics* [GRS91], agree on $M = \{\{p_{2n}\} \mid n \in \mathbb{N}\}$ as the intended model. We refer to [Sub99] for a survey of these issues.

One way of justifying the latter model as the intended one would be the following: Since we obtain $\{p_0\}$ as a consequence (in some natural, naive sense) of the program, we are inclined to dismiss the possibility that $\{\overline{p_0}\}$ could hold, since it is inconsistent with the knowledge of $\{p_0\}$. So we infer "not $\{\overline{p_0}\}$", meaning that $\{\overline{p_0}\}$ can be dismissed as possible consequence. It follows that there is no way to justify $\{p_1\}$ as a consequence of the program. Common practice in nonmonotonic reasoning semantics is to therefore conclude "not $\{p_1\}$", and to identify this with $\{\overline{p_1}\}$, allowing to conclude $\{p_2\}$, and so on.

We attempt to lift this line of reasoning to the general setting of logic programs in algebraic domains. In place of the notion of negation, which is not available in the general setting, we can try to use inconsistency. From this perspective, and refering again to the above *even numbers* program, we can indeed dismiss $\{\overline{p_0}\}$ as a possible consequence from the observation that $\{p_0\}$ can be derived. Again we conclude that there is no way to justify $\{p_1\}$ as a consequence of the program, hence we obtain "not $\{p_1\}$", i.e. the absence of $\{p_1\}$ as a possible conclusion. In general algebraic domains, however, without a notion of negation, there may be many compact elements inconsistent with $p_1$. While in the case of the domain $\mathbb{T}^{\mathcal{V}}$ we can justify to derive $\overline{p_1}$ from the absence of provability of $p_1$, i.e. taking $\overline{p_1}$ as a kind of default, it is unclear, in the general case, which of the elements inconsistent with $p_1$ should be taken.

In the absence of an involutive notion of negation, we therefore should distinguish between two kinds of "negation", as follows. Assume that we believe in some items, i.e. compact elements of a domain, and that the collection of these items is consistent. We then say (1) that a compact element is *refuted by contradiction* if if is inconsistent with a compact element which belongs to our believe and (2) that a compact element is *refuted by default* if it is not believed, and not refuted by contradiction. Let us finally call a compact element *refuted*, if it is refuted by contradiction or refuted by default.

Let us again review the above *even numbers* program. We refuted $\overline{p_0}$ by contradiction, while we refuted $p_1$ by default, leading us to assuming $\overline{p_1}$, i.e. $\overline{p_1}$ was interpreted as the statement "$p_1$ is refuted". It relies entirely on the existence of an involutive negation, that we are able to identify "$p_1$ is refuted" with $\overline{p_1}$. For algebraic domains, we should be able to abstract from an involutive negation, and this is facilitated by the following definition.

**3.2 Definition** Let $D$ be a coherent algebraic domain. An *extended clause* is a finite set $\{(c_1, N_1), \ldots, (c_n, N_n)\}$ where for all $i \in \{1, \ldots, n\}$ we have that $N_i$ is a clause in $D$ and $c_i \in \mathsf{K}(D)$. We call $(c, N) = (c, \{d_1, \ldots, d_n\})$ an *extended precondition* and abreviate it by $(c; d_1, \ldots, d_n)$, or by $c\neg d_1 \ldots \neg d_n$. In the latter notation, we omit $c$ if $c = \bot$ and $N \neq \{\}$. If $N = \{\}$ we abreviate $(c, N)$ by $c$. Note that $(\bot, \{\})$ can be abbreviated to $\bot$, in which case $\bot$ may not be omitted. An extended clause $\{(c_1, N_1), \ldots, (c_n, N_n)\}$ with $N_i = \{\}$ for all $i$ is called a *trivially extended clause.* A (*trivially*) *extended rule* is of the form $Y \leftarrow X$, where $Y$ is a clause and $X$ is a (trivially) extended clause. An (*extended disjunctive*) *program* consists of a set of extended rules.

We note that an extended disjunctive program which consists of trivially extended rules only, can be identified with a (non-extended) disjunctive logic program.

**3.3 Example** The following extended program $P$ is a more suitable representation of the *even numbers* program above. We use again the notation from Example 2.2.

$$\begin{aligned} \{p_0\} &\leftarrow \{(\bot,\{\})\} \\ \{p_{n+1}\} &\leftarrow \{(\bot,\{p_n\})\} \qquad \text{for all } n \in \mathbb{N} \end{aligned}$$

In abbreviated form, this program may be written as

$$\begin{aligned} \{p_0\} &\leftarrow \{\bot\} \\ \{p_{n+1}\} &\leftarrow \{\neg p_n\} \qquad \text{for all } n \in \mathbb{N}. \end{aligned}$$

We now seek a reasonable notion of logical consequence for this extended program. Consider some candidate theory $T$ which forms our belief. We next remove from the program all $\neg p_n$ for which $p_n$ is not a logical consequence of $T$, i.e. we consider these $p_n$ to be *refuted by default.* Then we remove all extended preconditions $(c,N)$ for which there is $p \in N$ with $T \models \{p\}$. The remaining program is no longer extended, and we call it $P/T$. From $P/T$ we can obtain its set of logical consequences, e.g. as $T' = \mathsf{fix}\left(\mathcal{T}_{P/T}\right)$. However, since $T'$ is in general different from $T$, the set of elements which are refuted by default using $T'$ is different from the set of elements refuted by default using $T$. But this means, that we are rather searching for a theory $S$ with $S = \mathsf{fix}\left(\mathcal{T}_{P/S}\right)$, or in other words, if we define the operator $\mathcal{D}_P$ on theories (i.e. sets of clauses) by $\mathcal{D}_P(A) = \mathsf{fix}\left(\mathcal{T}_{P/A}\right)$, then we are searching for fixed points of the operator $\mathcal{D}_P$. It is in fact easy to see that the desired theory $\mathsf{cons}(\{\{p_{2n}\} \mid n \in \mathbb{N}\})$ is a fixed point of $\mathcal{D}_P$. It is indeed its unique fixed point, which is rather satisfactory.

The reader familiar with the stable model semantics for logic programming [GL88] may recognize the constructions made in Example 3.3: It is the original approach to stable models. This can be carried over to logic programs with disjunctions as consequents of their rules and containing two kinds of negation, namely classical negation and default negation. Such programs are called *extended disjunctive logic programs*, and we refer to [GL91] for the stable model semantics for these programs, which we will now lift to logic programming on coherent algebraic domains.

**3.4 Definition** Let $D$ be a coherent algebraic domain, let $P$ be an extended disjunctive program, and let $T$ be a theory. We define $P/T$ to be the (non-extended) program obtained by applying the following two transformations: (1) Delete from $P$ all $\neg d$ for which $d$ is not a logical consequence of $T$. (2) Delete all extended preconditions $(c,N)$ for which there is $d \in N$ with $T \models \{d\}$. We define the *Gelfond-Lifschitz operator* or *default operator* $\mathcal{D}_P$ as a function on theories as $\mathcal{D}_P(T) = \mathsf{fix}\left(\mathcal{T}_{P/T}\right)$. A *stable model* of $P$ is a fixed point of $\mathcal{D}_P$, i.e. a theory $T$ such that $\mathcal{D}_P(T) = \mathsf{fix}\left(\mathcal{T}_{P/T}\right) = T$.

We obtain immediately from the definition that stable models are logically closed. Indeed, $\mathcal{D}_P$ maps logically closed theories to logically closed theories.

In order to justify our terminology, we have to explain what a *model* of an extended disjunctive program is.

**3.5 Definition** Consider a pair $(T,S)$ of theories, which we call an *interpretation*, and let $(c,N)$ be an extended precondition. We write $(T,S) \models (c,N)$ if $T \models \{c\}$ and for all $d \in N$ we have $S \not\models \{d\}$. If $X$ is an extended clause, then we write $(T,S) \models X$ if $(T,S) \models C$ for some extended precondition $C$ in $X$. The pair $(T,S)$ is called a *model* of $P$ if for every extended rule $Y \leftarrow X$ we have that $(T,S) \models X$ implies $T \models Y$. An interpretation $(T,S)$ is called *consistent* if $\mathsf{cons}(T) \subseteq \mathsf{cons}(S)$. It is called *total* if $T = \mathsf{cons}(T) = \mathsf{cons}(S) = S$.

We can now identify every theory $T$ with the total interpretation $(\mathsf{cons}(T), \mathsf{cons}(T))$. From this point of view, fixed points of the default operator are indeed models, as is easily verified.

**3.6 Proposition** Let $T$ and $S$ be logically closed theories and $S \subseteq T$. Then $\mathcal{D}_P(T) \subseteq \mathcal{D}_P(S)$, i.e. $\mathcal{D}_P(T)$ is antitonic. In particular, $\mathcal{D}_P^2$ is monotonic with respect to set-inclusion on the set of all logically closed theories.

Since $\mathcal{D}_P^2$ is antitonic, by the well-known Tarski fixed-point theorem, we obtain that $\mathcal{D}_P^2$ has a least fixed point, $L_P = \mathsf{lfp}\left(\mathcal{D}_P^2\right)$, and a greatest fixed point, $G_P = \mathsf{gfp}\left(\mathcal{D}_P^2\right)$.

**3.7 Proposition** We have $L_P = \mathcal{D}_P(G_P)$ and $G_P = \mathcal{D}_P(L_P)$. Furthermore, $(L_P, G_P)$ is a consistent model of $P$.

We call $(L_P, G_P)$ the *well-founded model* of $P$, borrowing terminology from nonmonotonic reasoning [Sub99].

**3.8 Theorem** For every stable model $S$ we have $L_P \subseteq S \subseteq \mathcal{D}_P(L_P)$. Furthermore, if $L_P = \mathcal{D}_P(L_P)$ for some program $P$, i.e. if the well-founded model is total, then $P$ has unique stable model $L_P$.

It can now easily be shown that the *even numbers* program from Example 3.3 has a total well-founded model, and hence a unique stable model.

## 3.2 Implicit and Explicit Knowledge

Extended disjunctive logic programming in algebraic domains enables us to represent knowledge in a variety of ways. Causal dependence may be encoded in the structure of the domain, i.e. implicitly, or explicitly by rules consituting a logic program. Likewise, negative information may be encoded implicitly in the domain, by facilitating inconsistency, or explicitly by using default negation. We give an example for this using a new representation of a classical problem.

**3.9 Example** We want to represent the following knowledge: (1) Tweety is a penguin. (2) Bob is a bird. (3) Birds fly or are penguins. (4) Birds always fly, unless the opposite can be shown. (5) Pengunins don't fly. (6) Penguins are birds.

We choose to represent (5) and (6) implicitly using the domain, and the remaining statements by a program. We first describe the domain $D$. Consider the set of items $A = \{p(T), p(B), b(T), b(B), f(T), f(B), n(T), n(B)\}$, where $T$ stands for "Tweety", $B$ stands for "Bob", $p(X)$ stands for "$X$ is a penguin", $b(X)$ for "$X$ is a bird", $f(X)$ for "$X$ can fly", $n(X)$ for "$X$ cannot fly". Now define $D$ to be the set of all subsets $c$ of $A$ which satisfy the following conditions for all $X \in \{B, T\}$: (i) $c$ does not contain both $f(X)$ and $n(X)$. (ii) $c$ does not contain both $b(X)$ and $p(X)$. (iii) $c$ does not contain both $p(X)$ and $n(X)$. (iv) $c$ does not contain both $p(X)$ and $f(X)$.

For $c, d \in D$ let $c \leq d$ if and only if one of the following holds: (i) $c \subseteq d$, (ii) $p(X) \in d$ and $c = (d \setminus p(X)) \cup b(X)$ for some $X \in \{B, T\}$, (iii) $p(X) \in d$ and $c = (d \setminus p(X)) \cup n(X)$ for some $X \in \{B, T\}$. We consider the domain $(D, \sqsubseteq)$, where $\sqsubseteq$ is the reflexive and transitive closure of $\leq$.

We note that in $D$, for all $X \in \{B, T\}$, sets containing both $n(X)$ and $f(X)$ are inconsistent, as are sets containing both $p(X)$ and $f(X)$. Now consider the following extended disjunctive program $P$

$$\begin{aligned} \{p(T)\} &\leftarrow \{\bot\} \\ \{b(B)\} &\leftarrow \{\bot\} \\ \{f(X), p(X)\} &\leftarrow \{b(X)\} \qquad \text{for all } X \in \{B, T\} \\ \{f(X)\} &\leftarrow \{b(X) \neg n(X)\} \qquad \text{for all } X \in \{B, T\} \end{aligned}$$

and the interpretation $S = \mathsf{cons}(\{p(T), b(T), n(T), b(B), f(B)\})$. The reader will easily verify that $S$ is a stable model of $P$. In particular, we notice that in this model Tweety does not fly, but Bob does.

Let us now analyse how knowledge is represented in Example 3.9. The sentences (1) to (4) are certainly being represented by the clauses of the program $P$, while (5) and (6) are satisfactorily represented by the structure of the underlying domain. We can regard (5) and (6) as background knowledge, and thus obtain a conceptually clean way of distinguishing between background or domain knowledge, and the explicit knowledge given by the program rules. Likewise, negated knowledge is treated. "Flying" and "not flying" are opposite properties, and can not hold of a single object at the same time. This knowledge is encoded in the structure of the domain, by making them inconsistent. Default negation was used explicitly in the program, and in the tradition of default logic was used for representing rules which "normally" hold, i.e. to which there may be exceptions.

## 4 Conclusions and Further Work

We introduced reasoning with default negation to domain theory, in the form of logic programming in coherent algebraic domains. Many possible lines of investigation open up from our first observations, and we want to name just a few. All these matters are under investigation by the author.

(1) Logic of domains. In the recent past, it became apparent that extended disjunctive logic programming, and its appropriate semantics, provides a very powerful tool for knowledge representation and reasoning, see eg. [Lif99, MT99]. It is therefore reasonable to expect that the well-established research area concerned with the relationships and the interplay between logic and domain theory could profit from extensions along these lines. Generalized approaches to the well-founded and the stable semantics, as in [DMT00], could lead the way. For the approach presented here it would be fortunate if the restriction to algebraic domains could be disposed of, mainly because e.g. the interval domain, and the subsumption lattice [NCdW97], fail to be algebraic.

(2) Domain-theory based logic programming. In classical logic programming, as implemented for example in Prolog, the use of negation is still unsatisfactory from a theoretical point of view, and it will probably remain so, since it can be argued that negation, as generally implemented in these systems, is not a clean declarative concept. Investigations on logic programming in algebraic domains may at some stage lead to a clean programming paradigm, including negation, which may be as powerful and applicable as modern Prolog systems. How this can be achieved is yet unclear, but first steps along these lines have already been performed, e.g. in [KRZ98]. Yet another line of research may be concerned with the machine learning paradigm known as *inductive logic programming* (ILP), see [MdR94], which still lacks a broad theoretical foundation. How this paradigm could be connected to domain theory proper is an open question, in particular since the *subsumption lattice*, which

features prominently in ILP [NCdW97], fails to be a domain. As we have seen in Example 3.9 above, however, logic programming in algebraic domains provides a very natural way for a conceptually clean distinction between background knowledge and programs. For a domain-theory-based ILP paradigm one would attempt to encode the background knowledge in the domain and learn program rules.

(3) Theoretical foundations of answer set programming and deductive databases. Although there is a broad base of theoretical work on answer set programming ([Lif99, MT99]) and deductive databases ([Min97]), domain-theoretic foundations have, to our knowledge, not yet been studied for these paradigms — apart from some investigations concerning fixed-point semantics, e.g. [KKM93, HS99, DMT00, Hit01]. Extended disjunctive logic programming as presented in this report may provide an important link.

(4) Application to reasoning on concept lattices [GW99b, GW99a]. Although concept lattices are structurally different from algebraic domains, we expect that reasoning techniques developed for the latter can be applied to the former, and indeed the pursuit of possible applications to concept lattices may also drive the theoretical development on algebraic domains. We forsee that an important matter which will have to be addressed in order to allow a cross-transfer is that of paraconsistency, i.e. the research will almost automatically shed light onto the question how negation should be handled for concept lattices.

## References

[AC98] R.M. Amadio and P.-L. Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1998.

[Cla78] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[DMT00] M. Denecker, V.W. Marek, and M. Truszynski. Approximating operators, stable operators, well-founded fixpoints and applications in non-monotonic reasoning. In J. Minker, editor, *Logic-based Artificial Intelligence*, chapter 6, pages 127–144. Kluwer Academic Publishers, Boston, 2000.

[Fit85] M. Fitting. A Kripke-Kleene-semantics for general logic programs. *Journal of Logic Programming*, 2:295–312, 1985.

[Fit94] M. Fitting. Metric methods: Three examples and a theorem. *Journal of Logic Programming*, 21(3):113–127, 1994.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R.A. Kowalski and K.A. Bowen, editors, *Logic Programming. Proceedings of the 5th International Conference and Symposium on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[GW99a] B. Ganter and R. Wille. Contextual attribute logic. In W.M. Tepfenhart and W.R. Cyre, editors, *Conceptual Structures: Standards and Practices, 7th International Conference on Conceptual Structures, ICCS'99, Blacksburg, Virginia, USA, July 1999*, volume 1640 of *Lecture Notes in Computer Science*, pages 377–388. Springer, Berlin, 1999.

[GW99b] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin, 1999.

[Hit01] P. Hitzler. *Generalized Metrics and Topology in Logic Programming Semantics.* PhD thesis, Department of Mathematics, National University of Ireland, University College Cork, 2001.

[Hit02] Pascal Hitzler. Resolution and logic programming in algebraic domains: Negation and defaults. Technical Report WV-02-05, Knowledge Representation and Reasoning Group, Department of Computer Science, Dresden University of Technology, Dresden, Germany, 2002.

[HS99] P. Hitzler and A.K. Seda. Some issues concerning fixed points in computational logic: Quasi-metrics, multivalued mappings and the Knaster-Tarski theorem. In *Proceedings of the 14th Summer Conference on Topology and its Applications: Special Session on Topology in Computer Science, New York*, volume 24 of *Topology Proceedings*, pages 223–250, 1999.

[HS0x] P. Hitzler and A.K. Seda. Generalized metrics and uniquely determined logic programs. *Theoretical Computer Science*, 200x. To appear.

[KKM93] M.A. Khamsi, V. Kreinovich, and D. Misane. A new method of proving the existence of answer sets for disjunctive logic programs: A metric fixed-point theorem for multivalued mappings. In C. Baral and M. Gelfond, editors, *Proceedings of the Workshop on Logic Programming with Incomplete Information, Vancouver, B.C., Canada*, pages 58–73, 1993.

[KRZ98] E. Klavins, W. Rounds, and G.-Q. Zhang. Experimenting with power default reasoning. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 1998.

[Lif99] V. Lifschitz. Answer set planning. In D. De Schreye, editor, *Logic Programming. Proceedings of the 1999 International Conference on Logic Programming*, pages 23–37, Cambridge, Massachusetts, 1999. MIT Press.

[MdR94] S. Muggleton and L. de Raedt. Inductive logic programming: Theory and applications. *Journal of Logic Programming*, 19–20:629–679, 1994.

[Min97] J. Minker. Logic and databases: Past, present, and future. *AI Magazine*, 18(3):21–47, 1997.

[MT99] V.M. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In K.R. Apt, V.W. Marek, M. Truszczyński, and D.S. Warren, editors, *The Logic Programming Paradigm: A 25 Year Persepective*, pages 375–398. Springer, Berlin, 1999.

[NCdW97] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence.* Springer, Berlin, 1997.

[Plo78] G. Plotkin. $\mathbb{T}^\omega$ as a universal domain. *Journal of Computer and System Sciences*, 17:209–236, 1978.

[Prz88] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, Los Altos, CA, 1988.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[RZ01] W.C. Rounds and G.-Q. Zhang. Clausal logic and logic programming in algebraic domains. *Information and Computation*, 171(2):156–182, 2001.

[SHLG94] V. Stoltenberg-Hansen, I. Lindström, and R. Griffor. *Mathematical Theory of Domains.* Cambridge University Press, 1994.

[Sub99] V.S. Subrahmanian. Nonmonotonic logic programming. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):143–152, January/February 1999.