A theory of nonmonotonic rule systems

W. Marek,[1] A. Nerode[2] and J. Remmel[3]

1 Introduction

In mathematics, a consequence drawn by a deduction from a set of premises can also drawn by the same deduction from any larger set of premises. The deduction remains a deduction no matter how the axioms are increased. This is monotonic reasoning, much imitated in other, less certain, disciplines. The very nature of monotonic reasoning makes mathematical proofs permanent, independent of new information. Thus it has been since Euclid and Aristotle. Theorems with complete proofs are never withdrawn due to later knowledge. It is little exaggeration to say that mathematicians never reject the completed proofs of their predecessors, except to complain about their constructivity.

Mathematicians build directly on the works of their forebearers stretching back two and a half millenia to Euclid. Our current mathematical reasoning is merely a fleshed-out version of Euclid's. Monotonic reasoning marks theoretical mathematics as a discipline. The traditional systems of mathematical logic are monotonic since they simple reflect mathematical usage. Tarski [43] described a calculus of deductive systems and captured in a simple way the general concept of a monotonic formal system. His formulation includes all logics traditionally studied, intuitionistic, modal, and classical. He did not qualify his definition, as we do, with the adjective "monotone," because there were no other systems studied at that time.

Minsky [32] suggested that there is another sort of reasoning which is not monotonic. This is reasoning in which we deduce a statement based on the absence of any evidence against the statement. Such a statement is in the category of beliefs rather than in the category of truths. Modern science offers

[1]Department of Computer Science, University Kentucky, Lexington, KY 40506–0027. Currently in Mathematical Sciences Institute at Cornell University. Work partially supported by N.S.F. grant RII 8610671 and Kentucky EPSCoR program and the ARO contract DAAL03-89-K-0124

[2]Mathematical Sciences Institute, Cornell University, Ithaca, NY 14853. Work partially supported by NSF grant MCS-83-01850 and ARO contract DAAG629-85-C-0018.

[3]Department of Mathematics, University of California at San Diego, La Jolla, CA 92903. Work partially supported by NSF grant DMS-87-02473.

as a tool for establishing provisional beliefs statistics, but in many instances we have no basis for applying statistics, due to a lack of governing distributions or samples for the problem at hand.

What role does belief play in our affairs? Often we must make sharp "yes or no" decisions between alternative actions. There may be no deductive or statistical base which justifies our choice, but we may not be able to wait for missing information, it may never materialize anyway. Often all we have as a basis for decision is surmise; that is, deductions from beliefs as well as truths and statistically derived statements. These beliefs are often accepted and used as premises for deduction and choice of action due to an unquantified lack of evidence against them.

A philosopher's much-quoted example is about Tweety. We observe only birds that can fly, and accept the belief that all birds can fly from the absence of evidence for the existence of non-flying birds. We are told that Tweety is a bird, and conclude that Tweety can't fly using our belief as premise. Later, we observe that Tweety is a pet ostrich and clearly can't fly. We reject our previous belief set and conclusions as a basis for decision making, and are forced to choose a new belief set. The new set of beliefs may also include equally uncertain statements, accepted due to a lack of evidence against. But we blithly draw consequences from the new belief set and make decisions on that basis till contrary evidence on some accepted belief is garnered, at which time we again have to acquire a new set of beliefs.

This has happened in the history of practically every subject except mathematics. The principles of physics, or biology, have been changed with every scientific revolution, even though unreflective practitioners of each age think that final principles have been found. Even for mathematics, the Dutch mathematician and philosopher L.E.J. Brouwer would have argued that the belief in theorems established by "non-constructive methods" was unjustified, and that a new belief set based on constructive principles should be adopted in its place. Other mainstream mathematicians such as Hilbert did not agree with this position. Some philosophers of mathematics living now would argue that, even within classical mathematics, the independence proofs for propositions of set theory, such as the continuum hypothesis or the axiom of choice, indicate there are several incompatible axiomatic systems which, as belief sets, could be the foundation of mathematics.

One can envisage making up non-monotone logics describing the mathematical nature of belief. The exact result depends on the definition chosen for "lack of evidence against". McCarthy [30], initiated the study of non-monotonicity with his notion of circumscription. With all relation symbols but one, $\bar{R}$, of a model (the world we are discussing) held fixed, and given axioms $\varphi(\bar{R})$ relating that $\bar{R}$ to the other (fixed) relations of the model, the belief should be that, lacking further evidence to the contrary, we should posit that $\bar{R}$ denotes the least relation R, if any, satisfying $\varphi(\bar{R})$. If further evidence in the form of an axiom $\psi(\bar{R})$ becomes available, then we should believe that $\bar{R}$ denotes the least R satisfying $(\varphi \wedge \psi)(\bar{R})$, if any, instead, in a changed belief set.

There are now many different non-monotonic system, abstracted from different questions in computer science and AI. Among the other systems that have been studied are

The theory of multiple believers of Hintikka, [18].

Truth Maintenance systems of Doyle, [8]

Default logic of Reiter, [40]

Autoepistemic logic of Moore, [34]

Theory of individual and common knowledge and belief of Halpern and Moses [17]

Logic programming with negation as failure [**?**].

This, by no means, exhausts the list. What issues in artificial intelligence or computer science motivates these systems?

Suppose that we build a robot in a "blocks world" to navigate in a room and avoid obstacles and perform simple tasks, such as crossing the room with variable obstacles. We want the robot to learn principles from experience as to how to cross the room. At any given point, one may imagine that the robot should have a consistent deductively closed set of beliefs which are the current basis for its actions, including such provisional beliefs as "I can always traverse the left edge of the room since there has never been anything in the way there". But when such a principle is contradicted by new obstacles, the robot has to choose another belief set. So an important problem is to define what a belief set is and how to compute them and how to update them based on new evidence. Moore's autoepistemic logic [34], is really a first try at this problem, mostly for propositional logic.

In computers, the operating system and program obey rules which compute how to change state. In the absence of exceptional behaviour, such as error conditions or failures to access resources, there is a system of decision rules (beliefs) computing how to change the state of the machine in this "normal behavior", or "default" case. But when an exceptional behavior happens, we are thrown to a different set of decision rules for change of state, a different set of "beliefs". One wants to be able to deduce what is true of the machine in states when it is a particular such "belief set". A logic for dealing with one such belief set at a time is Reiter's default logic.

In databases, facts and rules are stored as entries (the PROLOG model). Often also the database computes and stores conclusions, such as summary statistics or rules or tables computed from the database. These act as a deductive base for the set of current beliefs. When we query the database, we are asking for consequences of this belief set. When we update the database, all old entries that have changed have to be replaced, every consequence that uses these entries has to be recomputed and changed too. This is the process of replacing an old belief set by a new one. One often makes decisions on the basis of the absence of information in the database as well. A logic appropriate for describing a single such belief set is Doyle's truth-maintenance system [8]. See also de Kleer [6]. Also stable models for logic programming with negation as failure ([10]) arise in this way.

We expressed these examples informally in terms of the anthropomorphic notion of belief so as to bring out their common features. The actual non-monotonic logics have much in common, and a number of translations between them have been proposed ([23], [12],[13], [39],[28]). They have been investigated principally for propositional logic. Predicate versions suitable for actual applications are, up to now, pretty minimal.

Study of monotonic rule systems can be traced to the work of Post on "production systems" and to work of Tarski on the abstract properties of consequence relation for classical logic systems. The investigation of nonmonotonic component is of much more recent nature and seems to appear first in the work of Reiter on default logic. Reiter's investigations involved finding a natural extension of classical logic which allows one to handle the negative information.

Independently Clark, and subsequently Apt, Blair and Walker, and also (extending their work) Gelfond and Lifschitz, studied "negation as failure" in logic programming. It has turned out that these investigations are in a common direction. The mutual relations were uncovered by Bidoit and Froidevaux and Marek and Truszczynski, who exhibited the precise nature of the connection between logic programming and default logic. The reevaluation of default extensions in terms of "context-dependent proofs" by Marek and Truszczynski, which has its roots in the Apt, Blair and Walker's "elementary interpreter", for which it may serve as a clarifying definition, is a point of departure for the investigations of this paper. Here, drawing on all the research mentioned above for inspiration, we present a coherent unified theory of nonmonotonic formal systems.

At our level of abstraction we finally saw that non-monotone systems pervade ordinary mathematical practice. There is no sign of any realization of the existence of such mathematical examples in the previous non-monotonic logic literature. Perhaps these connections can only be seen by having a common abstract notion. What this commonality does for us is to make available known mathematical techniques from other areas of conventional mathematics for constructing and classifying belief sets (extensions), and simultaneously make evident a common thread among disparate parts of mathematics and disparate non-monotonic systems from artificial intelligence and computer science.

On the level of Mathematical Philosophy there is a connection worth stating as well. Non-monotone reasoning takes place during the process of discovery of mathematical theorems, when one posits temporarily some proposition on the basis that there is no evidence against it, and explores the consequences of such a belief until new mathematical facts force their abandonment. These non-monotone belief sets have their traces eradicated when final belief sets are achieved and demonstrative proofs are finished and published. The only hint of provisional belief sets left in mathematical papers is in the motivational remarks explaining what obstacles were overcome and by what changes in viewpoint the proof was achieved.

Here is the main definition. A non-monotone rule system consists of a set U and a set of triples (α, β, γ) called rules, where $\alpha = (\alpha_1, \ldots, \alpha_k)$ is a finite sequence of elements of U, called premises, $\beta = (\beta_1, \ldots, \beta_1)$ is a finite sequence of elements from U, called guards, and γ is an element of U. This is written, generalizing a notation of default logic, as

$$\frac{\alpha_1 \; \ldots, \alpha_n \colon \beta_1, \ldots, \beta_k}{\gamma}$$

The informal reading is: From $\alpha_1, \ldots, \alpha_k$ being established, and $\beta_1, \ldots, \beta_k$ not being established now or ever, conclude γ. You may substitute "computed" for "established" for an informal reading in many applications. A subset S of U is called *deductively closed* if for every rule of the system, whenever $\alpha_1, \ldots, \alpha_k$ are in S and $\beta_1, \ldots, \beta_n$ are not in S, then γ is in S. There are no variables here, these are not schema, this version is not the one appropriate for non-monotone predicate logics. Nonmonotonic predicate logics cannot be exposited in a few lines and we defer that to a later paper.

The intersection of all deductively closed sets containing a set I is generally not deductively closed. But the intersection of a descending chain of deductively closed sets is deductively closed, and I may be contained in many minimal deductively closed sets over I. In the context of nonmonotone logic the intersection of all deductively closed sets containing I is a (non-deductively closed) set, called the the set of *secure consequences* of I. These are the propositions a "skeptical reasoner" would take as beliefs based on I. The most important notion of contemporary nonmonotonic logic is that of *extension.* For a fixed subset S of U, one defines (finite) derivations from I, where all guards encountered are outside S, all premises encountered are conclusions of previous rules or in I. This defines the set $C_S(I)$ of S-consequences of I. Extensions are those S such that $S = C_S(I)$. These are minimal deductively closed sets containing I, but not conversely. These represent the "deductively closed, grounded, belief sets" that contain I. In these sets, if the negative guards are all obeyed, we are reduced to monotone reasoning. See Section 2 for the exact definition.

These simple definitions capture the common content of the several theories of non-monotonicity listed above, and of many mathematical theories as well. For example, the set of all marriages of the "marriage problem" can be formulated as exactly the set of all extensions in a non-monotone rule system; similarly for the set of all k-colorings of graphs, the set of chain covers of a partial order, the Stone space of all maximal ideals of a Boolean algebra, etc. Similarly, for a commutative ring with unit there is a non-monotone rule system such that the deductively closed sets are the prime ideals, the McCoy radical (the set of nilpotents) is the set of secured consequences of $\{0\}$, etc. There are similar non-monotonic systems associated with virtually every algebraic systremn for which radicals of some sort have been defined and characterized. These mathematical examples have suggested a whole new set of techniques for finding extensions because of the availability of algorithms already investigated in the mathematical literature on one or another of these problems, not previously known to be relevant in the artificial intelligence community. They not arise in logic, but really in operations research. Finally, in recursion theory, prioric constructions can be construed as non-monotone systems, sets constructed by the priority argument as extensions. These ideas give many constructions of recursively enumerable extensions.

We spend a lot of effort in both this and subsequent papers to answer the following question. Exactly how complicated is the set of extensions of a recursively enumerable nonmonotonic system, and what is its structure? This is the analogue of the classical logic question, how complicated is the set of complete theories containing a recursively enumerable theory, and what is its structure? In classical logic, this leads to analyzing the character of the set of maximal ideals containing given recursively enumerable ideal in a recursively presented free Boolean Algebra, a subject in which two of the authors have a lot of experience (see [42], [35]). The simplest case covering many nonmonotonic systems arising from mathematics is that of "highly recursive" nonmonotone rule systems. There, it turns out that extensions can be, up to a one-to-one recursive map, exactly any bounded Π^0_1 class os sets of natural numbers. So, even in this case, the computational problems are of the same level of difficulty as (say) solving "marriage problem" for highly recursive societies, or finding of recursively presented formally real field ([31]), or finding an abcissa between 0 and 1 where a given recursive continuous function on [0,1] takes a maximum value ([19]). Now this recursion-theoretic methodology can be refined to give complexity-theoretic results on the same problems about extensions, as has been done in algebra by Nerode and Remmel in [36], [37], and [38]. Since this is a more delicate matter than recursion theory, these developments are deferred again to a later paper.

Next, we turn to investigations of the semantics for nonmonotonic rule systems. The fundamental common semantics we have found comes from $L_{\omega_1\omega}$, and generalizes the Clark completion of logic programming. It is perfectly general, and gives systematic semantics and completeness for all the nonmonotonic logics discussed above. Such uniform semantics are new. Some of the subjects never before had a decent semantics. We find semantical representations of extensions, weak extensions and deductively closed sets. This representation requires creation of an additional infinitary language $\mathcal{L}_S$ which properly encodes not only rules as "first order objects", but also additional (infinitary) objects allowing characterization of the intended structures (extensions, weak extensions etc.). The previously established characterization of default logic, in terms of nonmonotonic rule systems, provides us with a semantics for default logic. This semantics, in opposition to the attempt of Etherington, satisfies Tarski's conditions. That is, it allows us to introduce for defaults (virtual) negations, conjunctions etc, and also a natural entailment relation. Finally, the $L_{\omega_1,\omega}$ proof procedures also yield new algorithms based on recursive well-founded trees.

This short summary indicates that there is a great wealth of problems and results which naturally arise from nonmonotonic rule systems. Our study delineates the role of (parametized) deducibility in nonmonotonic logics. This, in turn, connects our work naturally with studies of inductive definability. The latter indicates that logic programming will profit by less emphasis on predicate calculus, and more emphasis on inductive definability. Although this is a paradigm different from Kowalski's, we do not claim that this is the only "correct" position. But we do claim that it leads to a new direction for research.

The predicate logic case is not treated in this paper. It will come out from a schematic version of our theory, analogous to Post production systems. There, U is the set of all strings of an alphabet. There are typed "metavariables" ranging over specific subsets of U called "types", there are "metastrings". These are built from the alphabet of U and string variables, as sequence of elements of U and variables. Rules are of the same form as before, but use metastrings instead of strings. This point of view gives rise not only to a general theory, but also gives outright syntax, semantics, and completeness for new predicate versions of all the logics mentioned above. It also gives non-monotone classical, or intuitionistic, or modal predicate and propositional logics.

2 Nonmonotonic formal systems

Inspired by Reiter [40], and Apt [?], we introduce the notion of a nonmonotonic formal system $< U, N >$. A *nonmonotonic rule of inference* is a triple $< P, G, \varphi >$, where $P = \{\alpha_1, \ldots, \alpha_n\}$, $G = \{\beta_1, \ldots, \beta_m\}$ are finite lists of objects from U and $\varphi \in U$. Each such rule is written in form

$$r = \frac{\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m}{\varphi} \tag{1}$$

Here $\{\alpha_1, \ldots, \alpha_n\}$ are called the *premises* of rule r, $\{\beta_1, \ldots, \beta_m\}$ are called the *guards* of rule r.

Either, or both, lists P, G may be empty. If $P = G = \emptyset$ then the rule r is called an axiom.

A *nonmonotonic formal system* is a pair $< U, N >$, where U is a non-empty set and N is a set of nonmonotonic rules.

A monotonic formal system is a nonmonotonic system in which each rule has no guards. That is, each monotonic formal system can be identified with the nonmonotonic system in which every monotonic rule is given an empty set of guards.

A subset $S \subseteq U$ is called *deductively closed* if for every rule of N, if all premises $\alpha_1, \ldots, \alpha_n$ are in S and all guards $\beta_1, \ldots, \beta_m$ are not in S then the conclusion φ belongs to S.

In nonmonotonic systems, deductively closed sets are not generally closed under arbitrary intersections as in monotone case. But deductively closed sets are closed under intersections of *descending* chains. By the Kuratowski-Zorn Lemma, any $I \subseteq U$ is contained in at least one *minimal* deductively closed set. The intersection of all the deductively closed sets containing I is called the set of *secured consequences* of I. This set is also the intersection of all minimal deductively closed sets containing I. Deductively closed sets are thought of as representing possible "points of view". The intersection of all deductively closed sets containing I represents the common information present in all such "points of view", containing I. (Generally in the literature, if we assign to a given I a collection $\mathcal{M}$ of subsets of U , then assigning to I the intersection of $\mathcal{M}$ is called the *skeptical reasoning* associated with $\mathcal{M}$ and I.) Depending on the context we may talk about deductively closed sets containing I, weak extensions of I or extensions of I.

Example 2.1 *Let* $U = \{\alpha, \beta, \gamma\}$.
(a) Consider U *with* $N_1 = \{\frac{:}{\alpha}, \frac{\alpha:\beta}{\beta}\}$. *there is only one minimal deductively closed set* $S = \{\alpha, \beta\}$. *Then* $\{\alpha, \beta\}$ *is the set of secured consequences of* $< U, N_1$.
(b) Consider U *with* $N_2 = \{\frac{:}{\alpha}, \frac{\alpha:\beta}{\gamma}, \frac{\alpha:\gamma}{\beta}\}$,
then there are two minimal deductively closed sets, $S_1 = \{\alpha, \beta\}, S_2 = \{\alpha, \gamma\}$. $\{\alpha\}$ *is the set of secured consequences of* $< U, N_2 >$.

Example 2.1, (b) shows that the set of all secured consequences is not, in general, deductively closed.

Given a set S and an $I \subseteq U$, an S-deduction of φ from I in $< U, N >$ is a finite sequence $< \varphi_1, \ldots, \varphi_k >$ such that $\varphi_k = \varphi$ and, for all $i \leq k$, and each φ_i is in I or is an axiom, or is the conclusion of a rule $r \in N$ such that all the premises of r are included in $\{\varphi_1, \ldots, \varphi_{i-1}\}$ and all guards of r are in $U \setminus S$ (see [28], also [39]).

An S-consequence of I is an element of U occurring in some S-deduction from I. Let $C_S(I)$ be the set of all S-consequences of I in $< U, N >$.

Generally, $C_S(I)$ is not deductively closed in $< U, N >$. It is perfectly possible that all premises of a rule are in $C_S(I)$, the guards of that rule are outside $C_S(I)$, but a guard of that rule is in S, preventing the conclusion from being put into $C_S(I)$.

Example 2.2 $U = \{\alpha, \beta, \gamma\}, N = \{\frac{:}{\alpha}, \frac{\alpha:\beta}{\gamma}\},\ S = \{\beta\}$. *Then* $S_1 = C_S(\emptyset) = \{\alpha\}$ *is not deductively closed.*

However, the following holds:

Proposition 2.1 *If* $S \subseteq C_S(I)$ *then* $C_S(I)$ *is deductively closed.*

We say that $S \subseteq U$ is *grounded* in I if $S \subseteq C_S(I)$.
We say that $S \subseteq U$ is an *extension* of I if $C_S(I) = S$.
Finally, we say that$S \subseteq U$ is a *weak extension* of I if $C_S(I \cup R) = S$,
where $R = \{\varphi$: for some $r \in N,\ r = \frac{\alpha_1,\ldots,\alpha_n:\beta_1,\ldots,\beta_m}{\varphi}$,
$\alpha_1, \ldots, \alpha_n \in S, \beta_1, \ldots, \beta_m \notin S\}$ Thus S is a weak extension if S is generated by I and the conclusions of rules that are applicable. The notion of weak extension is related to Clark's completion and will be investigated below. The notion of groundedness is related to the phenomenon called "reconstruction". S is grounded in I if all elements of S are S-deducible from I (remember that S influences only the negative side of the rule). S is an extension of I if two things happen. First of all, every element of S is deducible from I, that is, S is grounded in I (this is analogue of adequacy). Second, the converse holds: all the S-consequences of I belong to S (this is analogue of completeness). Thus extensions are analogues for a nonmonotonic systems of the set of all consequences for monotonic systems. Both properties (adequacy and completeness) need to be satisfied.
The third concept, weak extension, is a closure property. In the process of constructing $C_S(I)$, S is used to generate only negatively as a restraint. But we can relax our requirements and allow deductions that use S also on the positive side. That is, S is not included, but is allowed to be used to generate objects from U by also testing the positive side of a rule for membership in S. This concept is closely related with the fixpoints of the operator T_P in logic programming, and Clark's completion, see [**?**].

The notion of extension is related to that of minimal deductively closed set.

Lemma 2.2 *If* S *is an extension of* I*, then:*
(1) S *is a minimal deductively closed superset of* I*.*
(2) For every I' *such that* $I \subseteq I' \subseteq S$, $C_S(I') = S$.

Proposition 2.3 *The set of extensions of* I *forms an antichain. That is, if* S_1, S_2 *are extensions of* I *and* $S_1 \subseteq S_2$*, then* $S_1 = S_2$.

Proposition 2.4 *An extension of* I *is a weak extension of* I.

Given an $S \subseteq U$, a rule r is called S-*applicable* if all the guards of r are outside of S and all the premises of r are in S. The collection $N(S)$ consists of all S-applicable rules.

With a nonmonotonic system $\mathcal{S} = < U, N >$ we associate the operator $T = T_{\mathcal{S}}\colon \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ defined by formula $T_{\mathcal{S}}(I) = \{\varphi \in U\colon \exists_{r\in N} r = \frac{\alpha_1,\ldots,\alpha_n:\beta_1,\ldots,\beta_m}{\varphi}$, $\alpha_1, \ldots, \alpha_n \in I, \beta_1, \ldots, \beta_m \notin I\}$ This operator is closely related to the operator T_P as considered in logic programming, see [**?**].

Proposition 2.5 *Let $< U, N >$ be a nonmonotonic rule system. Let T be its associated operator, and let $S \subseteq U$. Then:*
(1) $T(S) \subseteq S$ if and only if S is deductively closed.
(2) $T(S) = S$ if and only if S is a weak extension of $\emptyset$ in $< U, N >$.

Proposition 2.6 *Let $S \subseteq U$. Then S is a weak extension of $\emptyset$ in $< U, N >$ if and only if the following conditions are met:*
(i) S is closed under rules of N. That is, if there is a rule $r \in N$ such that $r = \frac{\alpha_1,\ldots,\alpha_n:\beta_1,\ldots,\beta_m}{\varphi}$, and $\alpha_1, \ldots, \alpha_n \in S, \beta_1, \ldots, \beta_m \notin S\}$ then φ belongs to S.
(ii) Whenever $\varphi \in S$ then there is a rule $r \in N$ such that $r = \frac{\alpha_1,\ldots,\alpha_n:\beta_1,\ldots,\beta_m}{\varphi}$, with $\alpha_1, \ldots, \alpha_n \in S, \beta_1, \ldots, \beta_m \notin S\}$.

Deductively closed sets here play the role that Herbrand models play in logic programming, Weak extensions play a role similar to that of supported models of programs, that is, models of Clark's completion, in logic programming. (see also Section 6).

A set S such that $T(S) \subseteq S$ is called a prefixpoint of T. There is no guarantee that T possesses a fixpoint.

Corollary 2.7 *For every system $< U, N >$, for every $S \subseteq U$ which is a prefixpoint of T, there is a minimal prefixpoint S' of T, $S' \subseteq S$.*

With each rule r we associate a monotonic rule

$$r' = \frac{\alpha_1, \ldots, \alpha_n}{\varphi} \tag{2}$$

obtained from r by dropping all guards. The rule r' is called the *projection* of rule r. The collection $M(S)$ is the collection of all projections of all rules from $N(S)$. The projection $< U, N >|_S$ is the monotone system $< U, M(S) >$. Thus $< U, N >|_S$ is obtained as follows: First, non-S applicable rules are eliminated. Then, the guards are dropped altogether. We have the following characterization theorem:

Theorem 2.8 *A subset $S \subseteq U$ is an extension of I, if and only if S is the deductive closure of I in $< U, N >|_S$.*

Theorem 2.8 tells us how to test if a collection $S \subseteq U$ is an extension of I in $< U, N >$. In case U and N are finite this leads to an algorithm.

(1) Compute $N(S)$.
(2) Project $N(S)$ by dropping guards to get $M(S)$.
(3) Compute the deductive closure T of I in $< U, M(S) >$, say T.
(4) Test whether $T = S$.

Proposition 2.9 *If S is a extension of I, then S consists entirely of elements of I and conclusions of certain rules in N.*

3 Examples and Applications in Logic, Logic Programming, and Commonsense Reasoning

3.1 Classical Implicational Propositional Logic

Here the set U is the collection of all well-formed formulas of propositional logic, over some collection *At* of atoms with binary connective $\Rightarrow$ and constant $\perp$. The standard Lukasiewicz axiomatization is represented as a collection of rules of the form:

$$\frac{}{\varphi \Rightarrow (\psi \Rightarrow \varphi)}$$

$$\frac{}{(\varphi \Rightarrow (\psi \Rightarrow \vartheta)) \Rightarrow ((\varphi \Rightarrow \psi) \Rightarrow ((\varphi \Rightarrow \vartheta))}$$

$$\frac{}{((\varphi \Rightarrow \perp) \Rightarrow (\psi \Rightarrow \perp)) \Rightarrow (((\varphi \Rightarrow \perp) \Rightarrow \psi) \Rightarrow \varphi)}$$

$$\frac{\varphi\ ,\ \varphi \Rightarrow \psi}{\psi}$$

The collection of derivable elements of U is the set of tautologies of propositional logic.

Propositional logic may be represented in other ways as well, for instance in the language with the usual connectives $\neg, \wedge, \vee, \Rightarrow$.

3.2 Default logic

Again let U be the collection of all formulas of propositional logic. A default theory $< D, W >$ is a pair where D a collection of default rules, that is, rules of form

$$\frac{\alpha\colon \mathrm{M}\beta_1, \ldots, \mathrm{M}\beta_m}{\omega}, \tag{3}$$

with W a collection of formulas of $\mathcal{L}$.
Represent such a default theory as a rule system consisting of three lists:
(i) Elements of $\omega \in W$ are represented as rules:

$$\frac{:}{\omega}$$

(ii) Rules of form (3) are represented as

$$\frac{\alpha\colon \neg\beta_1, \ldots, \neg\beta_m}{\omega}$$

(That is, the guards of the rule representing a default rule r have an additional negation in front).
(iii) Processing rules of logic. That is, all the monotonic rules of the system of classical logic.

We then have the following proposition:

Proposition 3.1 *A collection $S \subseteq U$ is an extension of a system consisting of rules of type* (i), (ii)*, and* (iii) *if and only if S is a default extension of $< D, W >$.*

3.3 Propositional logic programming, general case

A general logic program is a list of *general clauses*, of the form:

$$p \leftarrow q_1, \ldots, q_n, \neg r_1, \ldots, \neg r_m$$

We refer to [10] for the definition of a stable model of such a program. That concept is a generalization of the *perfect models* as introduced in [3].

Let U be the collection of atoms under consideration. Represent a general clause as a rule:

$$\frac{q_1, \ldots, q_n : r_1, \ldots r_m}{p}$$

The translation $tr(P)$ of a program P is the set of translations of its individual clauses.

The following result was proved in [4] and [29]:

Proposition 3.2 *A subset $M \subseteq U$ is a stable model of P if and only if M is an extension of $tr(P)$.*

Proposition 3.3 *A subset $M \subseteq U$ is a supported model of P if and only if M is a weak extension of $tr(P)$.*

3.4 Logic programming with classical negation

We now discuss so-called "logic programming with classical negation" of [11] as a chapter in the theory of nonmonotonic formal systems.
Recall the basic notions introduced in [11]. The collection of objects appearing in heads or bodies of clauses is the set of all literals, that is, atoms or negated

atoms. In particular, a negated atom may appear in the head of a clause. Consider first Horn clauses in which literals appear in an arbitrary place. To each set P of such clauses assign its *answer set*, the least collection A of literals satisfying the following two conditions:
(1) If $a \leftarrow b_1, \ldots, b_m$ is in P and $b_1, \ldots, b_m \in A$ than $a \in A$.
(2) If for some atom p, p and $\neg p$ are both in A, then A is the whole collection Lit of all literals.

Introduce a collection Str of *structural processing* rules over the set $U = Lit$. These are all monotone rules of the form:

$$\frac{p, \neg p}{a}$$

for all atoms p and literals a.
Translate the clause: $a \leftarrow b_1, \ldots, b_n$ as rule:

$$\frac{b_1, \ldots, b_n}{a}$$

and let tr(P) be the collection of translations of clauses in P plus the structural rules Str. Then we have

Proposition 3.4 *A subset $A \subseteq Lit$ is an answer set for P if and only if A is an extension of $tr(P)$. Since $tr(P)$ is a set of monotonic rules, such an answer set is the least fixpoint of the (monotonic) operator associated with the translation.*

Gelfond and Lifschitz then introduce general rules. Since the negation used in literals is not the "negation-as-failure" of general logic programming, Gelfond and Lifschitz introduce another negation symbol "*not*" and a general logic clause with classical negation in the form:

$$a \leftarrow b_1, \ldots, b_n, not(c_1), \ldots, not(c_m)$$

Then the *answer set* for a set P of clauses of this form is introduced by merging the operational procedure for the construction of stable models for a program (as introduced in [10]) with the procedure above. They define the answer set for a program with classical negation as follows:
Let $M \subseteq Lit$ and P be a general program. Define P/M as a collection of clauses lacking *not* and obtained as follows:
(1) If a clause C contains a substring $not(a)$ and $a \in M$, then eliminate C altogether.
(2) In remaining clauses eliminate all substrings of form $not(a)$.
The resulting program P/M lacks the symbol *not*, so the answer set is well defined. Let M' be the answer set for P/M. We call M an answer set for P precisely when $M' = M$.

Gelfond and Lifschitz give a computational procedure for finding such answer sets, and subsequently reduce computing them to computing default logic extensions. We show that the construction of Gelfond and Lifschitz is faithfully represented within nonmonotonic rule systems. Define U to be Lit, and translate the clause:

$$a \leftarrow b_1, \ldots, b_n, not(c_1), \ldots, not(c_m)$$

as the rule:

$$\frac{b_1, \ldots, b_n : c_1, \ldots, c_m}{a}$$

The translation of the program P then consists of the translations of individual clauses C of P, incremented by the structural rules Str. We get the following result:

Proposition 3.5 *Let P be a general logic program with classical negation and N_P be the translation described above. Then a collection M is an answer set for P if and only if M is an extension for the rule system $< U, N_P >$.*

4 Solutions to Combinatorial and Algebraical Problems as Extensions

4.1 The Marriage Problem

A *society*, $\mathcal{S} =< B, G, K >$ is a set B of boys, a set G of girls such that $B \cap G = \emptyset$, and a relation $K \subseteq B \times G$, the intended meaning of $< b, g > \in K$ being b knows g. A marriage for a society $\mathcal{S}$ is a map $M \colon B \rightarrow G$. A marriage M is *proper* if M is one-to-one and for all $b \in B$, $M(b) = g$ implies $K(b, g)$. That is, in a proper marriage each boy marries a girl he knows. A marriage M is *symmetric* if M maps B onto G. For a symmetric marriage, every girl is married.

For finite societies Philip Hall ([15]) gave a necessary and sufficient condition for the existence of a proper marriage, namely:
(*) For every finite set of boys $B' \subseteq B$, the set of girls that the boys of B' know altogether has cardinality greater or equal than that of B'.

Marshall Hall ([16]) showed that condition (*) is also a necessary and sufficient condition for the existence of marriages in an infinite society $\mathcal{S}$ as long as each boy knows only finitely many girls. Philip Hall's theorem is a special case of the more general problem of finding transversals (see [33]).

We claim that if $\mathcal{S} =< B, G, K >$ is a society satisfying (*) in which each boy knows only finitely many girls, then there is a nonmonotonic rule system $\mathcal{Z} =< U(\mathcal{S}), N(\mathcal{S}) >$ such that the collection of extensions of $\mathcal{Z}$ correspond exactly to the set of proper marriages of $\mathcal{S}$. To this end let us consider a collection of strings $U(\mathcal{S}) = \{Mbg \colon b \in B, g \in G, \text{ and } K(b, g) \text{ holds}\}$, where

M is a new symbol. Then for each boy $b \in B$ if $\{g_1, \ldots, g_n\}$ is the set of girls b knows, we add the following set of rules to $N(\mathcal{S})$.

$$\frac{: Mbg_1, \ldots, \widehat{Mbg_k}, \ldots, Mbg_n}{Mbg_k} \tag{4}$$

where we adopt the convention that for any sequence $s_1, \ldots, s_n$, $s_1, \ldots, \hat{s_k}, \ldots, s_n$ is the sequence that results from $s_1, \ldots, s_n$ by removing s_k.
For any girl g and any two boys $b_1 \neq b_2$, each of who knows g, add the following rules to $N(\mathcal{S})$:

$$\frac{Mbg_1, Mbg_2 :}{\varphi} \tag{5}$$

for every $\varphi \in U(\mathcal{S})$. Let $N(\mathcal{S})$ consists of all the rules of the form (4) or (5).

Theorem 4.1 *Let $\mathcal{S} =< B, G, K >$ be a society satisfying (*), for which each boy knows only finitely many girls. Then E is an extension for $\mathcal{Z} =< U(\mathcal{S}), N(\mathcal{S}) >$ if and only if $M_E = \{< b, g >: Mbg \in E\}$ is a proper marriage for $\mathcal{S}$.*

By expanding our set of rules $N(\mathcal{S})$, we can ensure that extensions correspond to proper symmetric marriages. That is, suppose that $\mathcal{S} =< B, G, K >$ is a society in which every boy knows only finitely many girls, and every girl knows only finitely many boys, and there is a symmetric marriage for $\mathcal{S}$. Let U be defined as before. In addition to all rules of form (4) and (5), add a set of rules for each $g \in G$.

If $g \in G$ and $\{b_1, \ldots, b_n\}$ is the set of boys that g knows, then add the following set of rules:

$$\frac{: Mb_1g, \ldots, \widehat{Mb_kg}, \ldots, Mb_ng}{Mb_kg} \tag{6}$$

Let $N_{Sym}(\mathcal{S})$ be the collection of rules of form (4), (5), and (6) and let $U_{Sym}(\mathcal{S}) = U$. By a proof which is similar to that of Theorem 4.1, we can prove the following:

Theorem 4.2 *Let $\mathcal{S} =< B, G, K >$ be a society such that each boy knows only finitely many girls and each girl knows only finitely many boys, and there is a proper symmetric marriage for $\mathcal{S}$. Then E is an extension for $\mathcal{Z}_{Sym} =< U_{Sym}(\mathcal{S}), N_{Sym}(\mathcal{S}) >$ if and only if $M_E = \{< b, g >: Mbg \in E\}$ is a proper symmetric marriage for $\mathcal{S}$.*

4.2 Complementary Subspaces of Vector Spaces

Let V_∞ be an countably infinite dimensional vector space over a finite field F, and let $B = \{b_0, b_1, \ldots\}$ be a basis for V_∞. If $S \subseteq V_\infty$, we let $(S)^\star$ denote the space generated by S. Let $V_n = (\{b_0, \ldots, b_n\})^\star$ for $n \geq 1$. Given two subspaces A and B of V_∞, we write $A+B$ for $(A \cup B)^\star$ and $A \bigoplus B$ for $A+B$ if $A \cap B = \{0\}$, where 0 is zero vector of V_∞.

Now, suppose that W is a subspace of V_∞. We can define a nonmonotonic rule system $< U, N >=< U_W(V_\infty), N_W(V_\infty) >$ so that extensions of $< U, N >$ correspond to the complementary subspaces for W that arise from the most natural construction of such spaces. That is, if one were going to construct a subspace A such that $A \bigoplus W = V_\infty$, a natural way to proceed would be to construct a sequence of subspaces $A_0 \subseteq A_1 \subseteq \ldots$ in stages as follows:
Stage 0 Let $A_0 = \{0\}$.
Stage s+1 Having defined a subspace $A_s \subseteq V_s$ such that $A_s \bigoplus W_s = V_s$ where $W_n = W \cap V_n$ for $n \geq 1$, we proceed according to one of two cases.
Case 1 W_s is properly included in W_{s+1}.
In this case, it is easy to show that $A_s \bigoplus W_{s+1} = V_{s+1}$, so we let $A_{s+1} = A_s$.
Case 2 $W_s = W_{s+1}$.
In this case it is easy to show that if $x_{s+1} \in V_{s+1} \setminus V_s$, $A_{s+1} = (A_s \cup \{x_{s+1}\}^\star$, then $A_{s+1} \subseteq V_{s+1}$ and $A_{s+1} \bigoplus W_{s+1} = V_{s+1}$.

Then $A = \bigcup_s A_s$ will be the desired complementary subspace of W. Note that we can get many different such complementary subspaces depending on choice of x_{s+1} at those stages in which Case 2 occurs at stage $s+1$.

Define our nonmonotonic rule system $< U, N >=< U_W(V_\infty), N_W(V_\infty) >$ as follows. We let $U = V_\infty$. Then we let N consists of the following five classes of rules:

$$\frac{:}{0}, \tag{7}$$

$$\frac{x_1, \ldots, x_k :}{\sum_{i=1}^{k} \lambda_i x_i}, \tag{8}$$

for all $x_1, \ldots, x_i \in V_\infty$, and $\lambda_1, \ldots, \lambda_k \in F$

$$\frac{w:}{y}, \tag{9}$$

for all $x \in W \setminus \{0\}$ and $v \in V_\infty$.

$$\frac{:}{x}, \tag{10}$$

where $x \in V_{s_0}$ and s_0 is the largest s such that $W_s = \{0\}$ and $W_{s+1} \neq \{0\}$.

$$\frac{a_1, \ldots, a_n : b_1, \ldots, b_m}{x}, \tag{11}$$

where $\{a_1, \ldots, a_n\}$ is a subspace and for some $s > s_0$, $A \bigoplus W_s = V_s$; $\{0\} \neq W_s = W_{s+1}$, $x \in V_{s+1} \setminus V_s$, and $\{b_1, \ldots, b_m\} = V_{s+1} - (A \cup \{x\})^\star$.

Theorem 4.3 *Let V_∞ be a countably infinite dimentional vector space over a finite field F, let $B = \{b_1, b_2, \ldots\}$ be a basis for V_∞ and let W be a subspace of V_∞. Then E is an extension of $< U, N >=< U_W(V_\infty), N_W(V_\infty) >$ if and only if E is a subspace such that $E \bigoplus W = V_\infty$ and for all $s \leq 1$, $E_s \bigoplus W_s = V_s$.*

5 Extensions of Highly Recursive Rule Systems

In this section we define the notions of recursive and highly recursive nonmonotonic rule systems. We show that the problem of finding an extension in a highly recursive nonmonotonic rule system is effectively equivalent to finding an infinite path through a recursive binary tree. That is, we prove that given any highly recursive nonmonotonic rule system $\mathcal{S} =< U, N >$, there is a recursive binary tree $T_{\mathcal{S}}$ and an effective one-to-one degree-preserving correspondence between the set of extensions of $\mathcal{S}$ and the set of infinite paths through $T_{\mathcal{S}}$. Conversely, we show that given any recursive binary tree T, there is a highly recursive nonmonotonic rule system $\mathcal{S}_T =< U_T, N_T >$ such that there is an effective one-to-one degree preserving correspondence between the set of infinite paths through T and the set of extensions of $\mathcal{S}_T$. It follows from the result of [21] that any recursively bounded Π^0_1-class can be coded as the set of infinite paths through a recursive binary tree.

We transfer all the results about degrees of elements of recursively bounded Π^0_1-classes to results about degrees of extensions in highly recursive nonmonotonic rule systems.

5.1 Paths through the Binary Trees and Extensions

To make the program outlined above precise, we first need some notation. Let $\omega = \{0, 1, 2, \ldots\}$ denote natural numbers and let $<,>: \omega \times \omega \to \omega$ be some fixed one-to-one and onto recursive pairing function such that the projection functions π_1 and π_2 defined by $\pi_1(< x, y >) = x$ and $\pi_2(< x, y >) = y$ are also recursive. We extend our pairing function to code n-tuples for $n > 2$ by usual inductive definition, that is $< x_1, \ldots, x_n >=< x_1, < x_2, \ldots, x_n >>$ for $n \geq 2$. We let $\omega^{<\omega}$ denote the set of all finite sequences from ω and $2^{<\omega}$ denote the set of all finite sequences of 0's and 1's. Given $\alpha =< \alpha_1, \ldots, \alpha_n >$ and $\beta =< \beta_1, \ldots, \beta_k >$ in $\omega^{<\omega}$, we write $\alpha \sqsubseteq \beta$ if α is initial segment of β, that is if $n \leq k$ and $\alpha_i = \beta_i$ for $i \leq n$. For the rest of this paper, we identify a finite sequence $\alpha =< \alpha_1, \ldots, \alpha_n >$ with its code $c(\alpha) =< n, < \alpha_1, \ldots, \alpha_n >>$ in ω. We let O be the code of the empty sequence $\emptyset$. Thus, when we say a set $S \subseteq \omega^{<\omega}$ is recursive, recursively enumerable, etc., we mean the set $\{c(\alpha): \alpha \in S\}$ is recursive, recursively enumerable, etc. A *tree* T is a nonempty subset of $\omega^{<\omega}$ such that T is closed under initial segments. A function $f: \omega \to \omega$ is an infinite *path* through T if for all n, $< f(0), \ldots, f(n) >\in T$. We let $\mathcal{P}(T)$ denote the set of all infinite paths through T. A set A of functions is a Π^0_1-class if there is a recursive predicate R such that $A = \{f: \omega \to \omega : \forall_n(R((f(0), \ldots, f(n)))\}$. A Π^0_1-class A is *recursively bounded* if there is a recursive function $g: \omega \to \omega$ such that $\forall_{f \in A} \forall_n (f(n) \leq g(n))$. It is not difficult to see that if A is a Π^0_1-class, then $A = \mathcal{P}(T)$ for some recursive tree $T \subseteq \omega^{<\omega}$. We sat that a tree $T \subseteq \omega^{<\omega}$ is *highly recursive* if T is a recursive, finitely branching tree such that there is a recursive procedure which, given $\alpha =< \alpha_1, \ldots, \alpha_n >$ produces a canonical index of the set of immediate successors of α in T, that is produces a canonical index of $\{\beta =< \alpha_1, \ldots, \alpha_n, k >: \beta \in T\}$. Here we say the canonical indes, $can(X)$, of the finite set $X = \{x_1 < \ldots < x_n\} \subseteq \omega$ is $2^{x_1} + \ldots + 2^{x_n}$ and canonical index of $\emptyset$ is 0. We let D_k denote the finite set whose canonical index is k, that is $can(D_k) = k$. It is then the case that if A is a recursively bounded Π^0_1-class, then $A = \mathcal{P}(T)$ for some highly recursive tree $T \subseteq \omega^{<\omega}$, see [21]. We note that if T is a tree contained in $2^{<\omega}$, then $\mathcal{P}(T)$ is a collection of $\{0,1\}$-valued functions and by identifying each $f \in \mathcal{P}(T)$ with the set A_f, $A_f = \{x: f(x) = 1\}$ of which f is the characteristic function, we can think of $\mathcal{P}(T)$ as a Π^0_1 class of sets.

Next we need to define the notions of recursive and highly recursive nonmonotonic rule systems $\mathcal{S} =< U, N >$. For the rest of this section we shall assume that $U \subseteq \omega$ and we shall identify a rule $r = \frac{\alpha_1, \ldots, \alpha_n : \beta_1, \ldots, \beta_m}{\varphi}$ in N with its code $c(r) =< k, l, \varphi >$ where $D_k = \{\alpha_1, \ldots, \alpha_n\}$ and $D_l = \{\beta_1, \ldots, \beta_m\}$. In this way, we think of N as a subset of ω. We say that $\mathcal{S} =< U, N >$ is *recursive* if U and N are recursive subsets of ω. To define the notion of highly recursive nonmonotonic rule system $\mathcal{S} =< U, N >$, we must first introduce the concept of *proof scheme* for φ in $< U, N >$. An (annotated) proof scheme for φ is a finite sequence

$$p =<< \varphi_0, r_0, can(G_0) >, \ldots, \tag{12}$$

$< \varphi_m, r_m, can(G_m) >>$
such that $\varphi_m = \varphi$ and
(1) If $m = 0$ then:
(a) φ_0 is an axiom (that is there exists a rule $r \in N, r = \frac{:}{\varphi_0}$) and $r_0 = r$, $G_0 = \emptyset$ or
(b) φ is a conclusion of a rule $r = \frac{:\beta_1, \ldots, \beta_r}{\varphi_0}$ and $r_0 = r$, $G_0 = \{\beta_1, \ldots, \beta_r\}$,
(2) $m > 0$, $<< \varphi_i, r_i, can(G_i >>_{i=0}^{m-1}$ is a proof scheme of length m and φ_m is a conclusion of $r = \frac{\varphi_{i_0}, \ldots \varphi_{i_s} : \beta_1, \ldots, \beta_r}{\varphi_m}$ $i_0, \ldots, i_s < m$, $r_m = r$ and $G_m = G_{m-1} \cup \{\beta_1, \ldots, \beta_r\}$
(3) $\{\varphi_1, \ldots, \varphi_m\} \cap u_m = \emptyset$.
The formula φ_m is called the *conclusion* of p and denoted $cln(p)$, the set G_m is called *support* of p and denoted $supp(p)$.

The idea behind this concept is this: we really care about schemata for proofs, and one scheme of proof is good for a large collection of sets S, as long as they satisfy natural constraints. A proof scheme brings all these constraints together.

A proof scheme with the conclusion φ may include a number of rules irrelevant to the enterprise of deriving φ. There is a natural preordering $\prec$ on proof schemes namely we say that $p \prec p_1$ if every rule appearing in p appears in p_1 as well. The relation $\prec$ is not a partial ordering, and it is not a partial ordering if we restrict ourselves to proof schemes with a fixed conclusion φ. Yet it is a well-founded relation, namely, for every proof scheme p there exists a proof scheme p_1 such for every p_2, if $p_2 \prec p_1$ then $p_1 \prec p_2$. Moreover we can, if desired, request the conclusion of p_1 to be the same as p.

Moreover, setting $p \sim p_1 \equiv (p \prec p_1 \wedge p_1 \prec p)$ we see that $\sim$ is an equivalence relation and that its cosets are finite.

We say that the system $< U, N >$ is locally finite if for every $\varphi \in U$ there are finitely many $\prec$-minimal proof schemes with conclusion φ. This concept is motivated by the fact that for locally finite systems for every φ there is a finite set of derivations Dr_φ, such that all the derivations of φ are inessential extensions of derivations in in Dr_φ. Finally, we say that $\mathcal{S}$ is *highly recursive* if $\mathcal{S}$ is recursive, locally finite, and the map $\varphi \mapsto can(Dr_\varphi)$ is partial recursive, that is there exists an effective procedure which, given any $\varphi \in U$, produces a canonical index of the set of all $\prec$-minimal proof schemes with conclusion φ. Also, we let $\mathcal{E}(\mathcal{S})$ denote the set of *extensions* of $\mathcal{S}$.

Formally, when we say that there is an effective, one-to-one degree preserving correspondence between the set of extensions $\mathcal{E}(\mathcal{S})$ of a highly recursive nonmonotonic rule system $\mathcal{S} =< U, N >$ and the set of infinite paths $\mathcal{P}(T)$ through a highly recursive tree T, we mean that there are indices e_1 and e_2 of oracle Turing machines such that
(i) $\forall_{f \in \mathcal{P}(T)} \{e_1\}^{gr(f)} = E_f \in \mathcal{E}(\mathcal{S})$,
(ii) $\forall_{E \in \mathcal{E}(\mathcal{S})} \{e_2\}^{E} = f_E \in \mathcal{P}(T)$, and
(iii) $\forall_{f \in \mathcal{P}(T)} \forall_{E \in \mathcal{E}(\mathcal{S})} (\{e_1\}^{gr(f)} = E$ if and only if $\{e_2\}^E = f)$.
where $\{e\}^B$ denotes the function computed by the e^{th} oracle machine with oracle B. Also, we write $\{e\}^B = A$ for a set A if $\{e\}^B$ is a characteristic function of A, and for function $f \colon \omega \to \omega$, $gr(f) = \{< x, f(x) >\colon x \in \omega\}$. As concerns our conditions (i)-(iii), the first ones say that the branches of the tree T uniformly produce extensions (via an algorithm with the index e_1), and that extensions of $\mathcal{S}$ uniformly produce branches of the tree T (via an algorithm with the index e_2). The condition (iii) asserts that if $\{e_1\}^{gr(f)} = E_f$ then f is Turing equivalent to E_f. In what follows, we shall not explicitely construct the indices e_1 and e_2 but it will be clear that such indices exist in each case.

Theorem 5.1 *Given a highly recursive nonmonotonic rule system $\mathcal{S} =< U, N >$, there is a highly recursive tree $T \subseteq 2^{<\omega}$ such that there is an effective one-to-one degree preserving correspondence between $\mathcal{E}(\mathcal{S})$ and $\mathcal{P}(T)$.*

We get several immediate consequences about the degrees of extensions in highly recursive nonmonotonic rule systems from Theorem 5.1, based on results of [21]. For any set $A \subseteq \omega$, let $A' = \{e \colon \{e\}^A(e) \text{ is defined}\}$ denote the jump of A and $0'$ denote the jump of the empty set $\emptyset$. We write $A \leq_T B$ if A is Turing reducible to B and $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T B$. We say that A is *low* if $A' \equiv_T 0'$. Thus A is low if the jump of A is as small as possible with respect to the Turing degrees.

Corollary 5.2 *Let $\mathcal{S} = < U, N >$ be a highly recursive nonmonotonic rule system such that $\mathcal{E}(\mathcal{S}) \neq \emptyset$. Then*

(i) There exists an extension E of $\mathcal{S}$ such that E is low.
(ii) If $\mathcal{S}$ has only finitely many extensions, then every extension E of $\mathcal{S}$ is recursive.

5.2 Highly Recursive Marriage Problems

We say that a society $\mathcal{S} =< B, G, K >$ in which every boy knows only finitely many girls is *highly recursive* if B and G are recursive subsets of ω, K is a recursive relation, and there is a recursive procedure which, given any $b \in B$, produces a canonical index of the finite set of girls known by b. If, in addition, each girl $g \in G$ knows only finitely many boys in B and there is a recursive procedure which, given any $g \in G$, produces a canonical index of the finite set of boys known by by g, then we say that $\mathcal{S}$ is *symmetrically highly recursive.* Now, it is easy to see that if $\mathcal{S}$ is a highly recursive society and we identify Mbg with its code $c(Mbg) =< b, g >$, then $< U(\mathcal{S}), N(\mathcal{S}) >$ is a recursive nonmonotonic rule system. However, as it stands, $< U(\mathcal{S}), N(\mathcal{S}) >$ is not highly recursive rule system because the rules of the form (5) which allow for infinitely many minimal derivations of φ.

However, if $\mathcal{S}$ is symmetrically highly recursive, then a slight modification of rules (5) will produce a highly recursive nonmonotonic rule system with the same extensions. That is, suppose $\mathcal{S} =< B, G, K >$ is a symmetrically highly recursive society which has a proper marriage. Let $U(\mathcal{S}) = \{Mbg\colon b \in B, g \in G, \text{ and } < b, g >\in K\}$ as before. Now suppose $b_1 \neq b_2$ are boys which know the same girl. Then clearly one of boys b_1 and b_2 must know at least two girls, since otherwise there can be no proper marriage for $\mathcal{S}$. Since $\mathcal{S}$ is highly recursive, $B_2 = \{b \in B\colon b \text{ knows at least two girls }\}$ is a recursive set. Now consider rules of the form

$$\frac{Mb_1g, Mb_2g\colon}{Mb_3g'} \tag{13}$$

for all $b_1, b_2 \in B, g \in G$ where $b_3 = max(\{b_1, b_2\} \cap B_2)$ and $g' \neq g$.
Let $\overline{U(\mathcal{S})} = U(\mathcal{S})$ and $\overline{N(\mathcal{S})}$ consists of rules of the form (4) and (13). Then we have the following

Theorem 5.3 *Let $\mathcal{S} =< B, G, K >$ be a symmetrically highly recursive society such that $\mathcal{S}$ possesses a proper marriage. Then*
(i) $< \overline{U(\mathcal{S})}, \overline{N(\mathcal{S})} >$ is a highly recursive nonmonotonic rule system and
(ii) E is an extension of $< \overline{U(\mathcal{S})}, \overline{N(\mathcal{S})} >$ if $M_E = \{< b, g >\colon Mbg \in E\}$ is a proper marriage of $\mathcal{S}$.

The same modification can be applied to the symmetric marriage problem. That is, suppose that $\mathcal{S} =< B, G, K >$ is a symmetrically highly recursive society. Let $\overline{U_{sym}(\mathcal{S})} = \overline{U(\mathcal{S})}$, and $\overline{N_{sym}(\mathcal{S})}$ be all rules of form (4), (13), and (5), Then we have the following.

Theorem 5.4 *Let $\mathcal{S} =< B, G, K >$ be a symmetrically recursive society such that $\mathcal{S}$ has a proper symmetric marriage. Then*
(i) $< \overline{U_{sym}(\mathcal{S})}, \overline{N_{sym}(\mathcal{S})} >$ is a highly recursive nonmonotonic rule system and
(ii) E is an extension of $< \overline{U_{sym}(\mathcal{S})}, \overline{N_{sym}(\mathcal{S})} >$ if and only if the mapping $M_E = \{< b, g >: Mbg \in E\}$ is a proper marriage of $\mathcal{S}$.

5.3 Recursion-theoretic results for extensions

The results of Manaster and Rosenstein, and Remmel combined with Theorem 5.4 yield the following.

Theorem 5.5 *Let C be any recursively bounded Π^0_1-class. Then there is a highly recursive nonmonotonic rule system $< U, N >$ and an effective one-to-one degree preserving correspondence between the elements of C and the set of all extensions of $< U, N >$.*

Theorem 5.5 now allows us to transfer many results about possible degrees of elements of recursively bounded Π^0_1-classes to results about degrees of extensions of highly recursive nonmonotonic rule systems. Below we shall list a few examples of such results.

Corollary 5.6 *There is a highly recursive nonmonotonic rule system $< U, N >$ such that $< U, N >$ has $2^{\aleph_0}$ extensions but no recursive extensions.*

Corollary 5.7 *There is a highly recursive nonmonotonic rule system $< U, N >$ such that $< U, N >$ has $2^{\aleph_0}$ extensions and any two extensions $E_1 \neq E_2$ of $< U, N >$ are Turing incomparable.*

Corollary 5.8 *If $\mathbf{a}$ is any Turing degree that $\mathbf{0} <_T \mathbf{a} \leq_T 0'$, then there is a highly recursive nonmonotonic rule system $< U, N >$ such that $< U, N >$ has $2^{\aleph_0}$ extensions but no recursive extensions and $< U, N >$ has an extension of degree $\mathbf{a}$. (Here 0 is the degree of recursive sets.)*

Corollary 5.9 *If $\mathbf{a}$ is any Turing degree that $\mathbf{0} <_T \mathbf{a} \leq_T 0'$, then there is a highly recursive nonmonotonic rule system $< U, N >$ such that $< U, N >$ has $\aleph_0$ extensions, $< U, N >$ has an extension E of degree $\mathbf{a}$ and if $E' \neq E$ is an extension of $< U, N >$, then E' is recursive.*

Corollary 5.10 *There is a highly recursive nonmonotonic rule system $< U, N >$ such that $< U, N >$ has $2^{\aleph_0}$ extensions and if $\mathbf{a}$ is the degree of any extension E of $< U, N >$ and $\mathbf{b}$ is any recursively enumerable degree such that $\mathbf{a} <_T \mathbf{b}$, then $\mathbf{b} \equiv_T 0'$.*

Corollary 5.11 *If* **a** *is any recursively enumerable Turing degree, then there is a highlly recursive nonmonotonic rule system* $< U, N >$ *such that* $< U, N >$ *has* $2^{\aleph_0}$ *extensions and the set of recursively enumerablr degrees* **b** *which contain an extension of* $< U, N >$ *is precisely the set of all recursively enumerable degrees* $\mathbf{b} \geq_T \mathbf{a}$.

All of the above results follow from Theorem 5.5 plus the corresponding results for recursively bounded Π^0_1-classes due to Jockusch and Soare [21] [22] except of Corollary 5.10 which follows from the corresponding result for recursively bounded Π^0_1-classes due to Jockusch and McLaughlin [20].

Now we give a construction of a rule system $< U, N >$ whose extensions directly code infinite paths through a binary tree T and provide us with a more direct route to Theorem 5.5 which avoids using the results of [24] or [41].

Example 5.1 *Paths through binary trees.*

Let $\mathcal{T}$ be a recursive binary tree contained in $2^{<\omega}$. Let $U(\mathcal{T}) = \{P_i, \overline{P_i} : i \in \omega\}$. Our idea is to have a set π such that $| \pi \cap \{P_i, \overline{P_i}\} |= 1$ for all i correspond to a path $f_\pi : \omega \to \omega$ through the complete binary tree $B_\omega = 2^{<\omega}$ where

$$x = \begin{cases} 1 & \text{if } P_i \in \pi \\ 0 & \text{if } \overline{P_i} \in \pi \end{cases}$$

$P_i \in \pi$ says that we branch right at level i, and $\overline{P_i} \in \pi$ says that we branch left at the level i. Now, for any node $\sigma =< \sigma(0), \ldots, \sigma(n) >$, let $\vec{P_\sigma} = \{\sigma(P_0), \ldots, \sigma(P_n)\}$ where

$$\sigma(P_i) = \begin{cases} P_i & \text{if } \sigma(i) = 1 \\ \overline{P_i} & \text{if } \sigma(i) = 0 \end{cases}$$

We say that $\sigma =< \sigma(0), \ldots, \sigma(n) >$ is a terminal node of $\mathcal{T}$ if both $< \sigma(0), \ldots, \sigma(n), 0 > \notin \mathcal{T} < \sigma(0), \ldots, \sigma(n), 1 > \notin \mathcal{T}$.
Then we consider the following set of rules.

$$\frac{: P_i}{\overline{P}_i} \quad \frac{: \overline{P}_i}{P_i} \tag{14}$$

$$(a) \qquad \frac{\sigma(P_0), \ldots, \sigma(P_n):}{P_n} \tag{15}$$

for all σ which are terminal nodes of $\mathcal{T}$ where $\sigma(P_n) = \overline{P}_n$

$$\text{(b)} \qquad \frac{\sigma(P_0), \ldots, \sigma(P_n):}{\overline{P}_n}$$

for all σ which are terminal nodes of $\mathcal{T}$ where $\sigma(P_n) = P_n$.
Let $N(\mathcal{T})$ consists of all rules of the form (14) and (15). Then we have the following (if we identify P_i with its code $2i$ and $\overline{P}_i$ with its code $2i + 1$).

Theorem 5.12 *Let $\mathcal{T} \subseteq 2^{<\omega}$ be a recursive tree.*
(i) $< U(\mathcal{T}), N(\mathcal{T}) >$ is a highly recursive nonmonotonic rule system and
(ii) E is an extension of $< U(\mathcal{T}), N(\mathcal{T}) >$ if and only if the map $f_E : \omega \to \omega$ defined by

$$f_E(i) = \begin{cases} 1 & \text{if } P_i \in E \\ 0 & \text{if } \overline{P}_i \in E \end{cases}$$

is an infinite path through $\mathcal{T}$.

Given Theorems 5.1 and 5.5, it is natural to ask if there are analogous results for locally finite nonmonotonic rule systems which are recursive, but not highly recursive. The answer is "yes". That is, we say that a tree $T \subseteq \omega^{<\omega}$ is *highly recursive* in $0'$ if T is recursive in $0'$, T is finitely branching, and there is a procedure which is recursive in $0'$ and which given any node $\eta \in T$ will produce the canonical index of the set of immediate successors of η in T. Then the analogues of Theorems 5.1 and 5.5 hold for recursive nonmonotonic rule systems if we replace highly recursive trees by trees which are highly recursive in $0'$.

Moreover, by relativization to the code of the collection of rules $< U, N >$ we are able to deal with the case of *arbitrary* locally finite nonmonotonic system $\mathcal{S}$. The distinction between the form of function that computes the canonical index of the collection of prooof schemes for elements of U remains, if this function is recursive in (the code of) $< U, N >$, then the tree $\mathcal{T}$ whose branches code extensions of $< U, N >$ is recursive in (the code of) $< U, N >$. Otherwise it is recursive in its jump.

These results will be proved in a subsequent paper.

5.4 Some applications to Logical Systems

The results of Sections 5.1 and 5.3 can be interpreted, using Sections 3.2 and 3.3, as (new) results about default logic and logic programming. The relationship between stable semantics for logic programs and default logic and the results of Section 3.3 show the relevance of proof schemes to the construction of stable models for logic programs. As far as we know, programs with the local finiteness property have not been discussed previously in the literature, although this covers most practical programs. The definition of proof scheme with a "forbidden" set of atoms (corresponding to the definition of support of a proof scheme above) is perfectly natural and can be lifted from definition (12) in an obvious fashion. The ordering $\prec$ has the same meaning as before. This way, we get a natural concept of a locally finite (propositional) program. When the program P involves variables, we interpret P as the collection of its Herbrand constant substitutions. This gives rise to a definition of locally finite program. The rule systems of Sections 5.1 and 5.3 can be rewritten following *reverse* translations of Section 3.3 (notice that we deal there only with atoms!). That is, the rule $\frac{q_1,\ldots,q_n : r_1,\ldots,r_m}{p}$ is translated to: $p \leftarrow q_1, \ldots, \neg q_n, r_1, \ldots, \neg r_m$. Using Proposition

3.2 we get stable models from extensions. It is easy to see that the concept of proof scheme is preserved, locally finite systems generate locally finite programs. Then, in an analogous manner, we introduce the notion of a "highly recursive program" as one that is recursive, locally finite, and for which a function assigning to p the code of its finite collection of its $\prec$-minimal proof schemes is recursive. Let $Stab(P)$ be the collection of stable models of the program P. We then get

Theorem 5.13 *Given a highly recursive program P there is a highly recursive tree $T \subseteq 2^{<\omega}$ and an effective one-to-one degree preserving correspondence between $Stab(P)$ and $\mathcal{P}(T)$.*

Exactly the same lifting may be done for default logic. We leave the details to the reader.

So the results of Jockusch and Soare apply both to logic programming and to default logic, and we get a series of results on recursion theory of stable models of logic programs by lifting Corollary 5.2, Theorem 5.5, Corollaries 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, and Theorem 5.12.

It is appropriate to compare the results of this section with those of [2]. They construct, for a given natural number $n \geq 1$, a stratified finite program P (in particular its Herbrand expansion is a recursive propositional program) whose perfect model is a complete Σ^0_n set of natural numbers. Since the perfect model is stable, and stratified programs possess the unique stable model (as pointed by [10]), the collection $Stab(P)$ is a one element class. Then this is a Π^0_2-class, whose only element is a Σ^0_n set. Our results show that it is impossible to find a recursive program possessing a unique stable model which is Π^1_1-complete; as the unique element of an arithmetical singleton class in 2^ω must be hyperarithmetical.

6 Semantics

Let $< U, N >$ be a deductive system and assume that $|U| = \omega$. Without loss of generality we may identify the set U with the set ω of natural numbers, and N, which consists of finite objects, with a subset of ω.

Let us recall that we wish to characterize three classes: minimal sets closed under N, weak extensions, and extensions of $< U, N >$. We shall provide a semantic characterization of these concepts. These characterization use the infinitary logic.

The logic $\mathcal{L}_S$ is defined as the closure of a collection of atoms of form "$\varphi \in S$" (φ ranging over U) under negation, arbitrary denumerable conjunctions and arbitrary denumerable disjunctions.

Given $T \subseteq U$, and φ a formula of $\mathcal{L}_S$, define the satisfaction relation $T \models \varphi$

by induction in the most natural fashion:
(1) $T \models \alpha \in S$ if and only if $\alpha \in T$.
(2) $T \models \neg\psi$ if and only if not$(T \models \psi)$.
(3) $T \models \bigwedge_{i \in J} \psi_i$ if and only if for all $i \in J$, $T \models \psi_i$.
(4) $T \models \bigvee_{i \in J} \psi_i$ if and only if there exists $i \in J$, such that $T \models \psi_i$.
The connectives $\Rightarrow$ and $\Leftrightarrow$ are abbreviations.
Associate with each rule:

$$r = \frac{\alpha_1, \ldots \alpha_n : \beta_1, \ldots, \beta_m}{\varphi} \tag{16}$$

a finitary formula of $\mathcal{L}_S$,

$$t(r) = [\alpha_1 \in S \wedge \ldots \wedge \alpha_n \wedge \neg(\beta_1 \in S) \wedge \ldots \tag{17}$$

$\wedge\neg(\beta_m \in S)] \Rightarrow \varphi \in S$.
The object φ is denoted by $c(r)$.

Proposition 6.1 *A subset T of U is deductively closed if and only if for all $r \in N$, $T \models t(r)$.*

Generalizing *Clark's completion* from logic programming, we define "Clark's completion" of a deductive system $< U, N >$. This is a theory in $\mathcal{L}_S$, possibly infinitary. To define it, assume that r is a rule of form (16). Set

$$A(r) = \alpha_1 \in S \wedge \ldots \wedge \alpha_n \in S \wedge \neg(\beta_1 \in S) \wedge \tag{18}$$

$\ldots \wedge \neg(\beta_m \in S)$.
Thus $t(r) = A_r \Rightarrow (c(r) \in S)$. Now, given $\alpha \in U$, let F_α be

$$\alpha \in S \Leftrightarrow \bigvee \{A_r : r \in N \wedge c(r) = \alpha \in S\} \tag{19}$$

F_α says that α belongs to T exactly if it is supported by a formula of the form A_r for some $r \in N$.

The formulas F_r can be used to characterize weak extensions.

Theorem 6.2 *A collection $T \subseteq U$ is a weak extension of $< U, N >$ if and only if for all $\alpha \in U$, $T \models F_\alpha$.*

Now, identifing U with the set of natural numbers ω, the collection of all subsets of U is identified with 2^ω, this is the Cantor space. Then:

Proposition 6.3 *For every formula $\Phi \in \mathcal{L}_S$, $\{T : T \models \Phi\}$ is a Borel subclass of 2^ω in the Cantor topology.*

Corollary 6.4 *(a) Let $< U, N >$ be a deductive system, $U = \omega$. The collection W of weak extensions of $< U, N >$ is a Borel subclass of 2^ω.*
(b) Consequently, $|W|$ is finite, or $|W| = \omega$ or $|W| = 2^{\aleph_0}$.

When $< U, N >$ is recursive, then the formula $\bigvee\!\!\!\bigvee\{A_r : \psi = c(r)\}$ is representable as a recursively enumerable set of natural numbers. it follows that:

Proposition 6.5 *If $< U, N >$ is recursive, then the collection of weak extensions of $< U, N >$ is a Π^0_2 subclass of 2^ω.*

The collection of all extensions of a deductive system has a model-theoretical characterization. Using the idea behind proof schemes we can introduce an infinitary description of provability. Fix $< U, N >$.

Proposition 6.6 *For every $\psi \in U$ there exists a formula $pr_\psi \in \mathcal{L}_S$ such that for every $T \subseteq U$, $T \models pr_\psi$ if and only if ψ possesses a T-derivation. (pr_ψ depends on N)*

Corollary 6.7 *Let $< U, N >$ be a deductive system. Then $T \subseteq U$ is an extension of $< U, N >$ if and only if:*
(1) For all $\psi \in T$, $T \models pr_\psi$, and
(2) For all $\psi \notin T$, $T \models \neg pr_\psi$.

Corollary 6.8 *(a) Let $< U, N >$ be a deductive system, $U = \omega$. The collection E of extensions of $< U, N >$ is $\Pi^{0,N}_2$ subclass of 2^ω.*
(b) Consequently, $|E|$ is finite, or $|E| = \omega$ or $|E| = 2^{\aleph_0}$.

6.1 Applications in Default Logic and Logic Programming

Proposition 6.9 *Let $< D, W >$ be a default theory. $< U, N >$ be its translation as a nonmonotonic rule system, and let S be a subset of U satisfying translation of $< D, W >$. Then:*
(1) S is a weak default extension of $< D, W >$ if and only if for all $\varphi \in \mathcal{L}$, $S \models F_\varphi$.
(2) S is a default extension of $< D, W >$if and only if:
(i) for all $\vartheta \in S$, $S \models pr_\vartheta$.
(ii) for all $\vartheta \notin S$, $S \models \neg pr_\vartheta$.

Compare this result with the earlier results of Etherington ([9]) who characterized default extensions by means of "most preferred models". Here we use a different device. A careful inspection of our method indicates the following: First, imbed the language $\mathcal{L}$ into a new language $\mathcal{L}_S$. This language $\mathcal{L}_S$ possesses a new *atom* for every *formula* of $\mathcal{L}$. Thus, $\mathcal{L}_S$ is a much richer language. Second, formulas of $\mathcal{L}$ are translated as atoms of $\mathcal{L}_S$. The relationship between various formulas of $\mathcal{L}$ is enforced in $\mathcal{L}_S$ by means of translations of rules. Default rules of $\mathcal{L}$ are translated to certain finitary clauses of $\mathcal{L}_S$. Checking satisfaction for these clauses refers only to simpler formulas of $\mathcal{L}_S$. Some of these are not images of formulas of $\mathcal{L}$. The semantic characterization of extensions and weak extensions refers to formulas of $\mathcal{L}_S$ which are not images of formulas of $\mathcal{L}$. In addition, these formulas used in characterization are infinitary. This indicates an infinitary character of the concept of extension and weak extension.

One needs to notice that when the theory $< D, W >$ is finite, our description of it, via the translation to nonmonotonic rule systems is finitary. Also, the characterization formulas pr_ψ are finitary. The reason for this is that, in addition to rules in D, we also have all the rules of logic. The schemes of proof of logic result in infinitely many rules which are, however, monotonic. There are infinitely many proof schemes, but the collection of formulas of form $k(p)$ is finite anyway! This fact results in the finitary algorithm described in [25].

Our translation of propositional logic programs as rule systems provides us with an infinitary characterization of stable models of logic programs. The reason this is important is that the definition of stable model of logic program, as introduced in [10], is purely operational. Let P be a logic program, Π its propositional version, that is the collection of all the Herbrand substitutions of P. Let H be the Herbrand base of P and let $M \subseteq H$. Gelfond and Lifschitz give an algorithm for testing whether M is stable structure for P and prove that a stable structure is, actually, a minimal model of P. It should be clear from this description that this definition is purely operational. Here, using the infinitary language $\mathcal{L}_S$ we give a purely logical, although infinitary, description of stability.

Proposition 6.10 *Let P be a logic program, Π its propositional version, H, Herbrand base of P, and finally $< H, T >$ the translation of Π described in Section 3.3. Then $M \subseteq H$ is a stable model of P if and only if for every $\vartheta \in M$, ϑ is M-derivable in $< H, T >$. and for every $\vartheta \notin M$, ϑ is not M-derivable in $< H, T >$.*
Thus $M \subseteq H$ is a stable model of P if and only if
(i) for every $\vartheta \in M$, $M \models pr_\vartheta$.
(ii) for every $\vartheta \notin M$, $M \models \neg pr_\vartheta$.

7 Conclusion

In a sequel we deal with rule systems containing variables in the rules. We shall deal with predicate logics. We shall prove results related to the properties of recursive systems that are not necessarily highly recursive. We also explore connections with $L_{\omega_1,\omega}$.

We acknowledge helpful conversations and discussions with Krzysztof Apt, Howard Blair, Michael Gelfond, John Schlipf, V.S. Subrahmanian, and Miroslaw Truszczyński.

References

[1] K.R. Apt. Introduction to Logic Programming. TR-87-35, University of Texas, 1988.

[2] K.R. Apt, H.A. Blair. Classification of Perfect Models of Stratified Programs. To appear in *Fundamenta Informaticae.*

[3] K.R. Apt, H.A. Blair, A. Walker. Towards a theory of declarative knowledge. In: J. Minker ed. *Foundations of Deductive Databases and Logic Programming*, pp. 89-142, Morgan Kaufmann, Los Altos, CA.

[4] N. Bidoit, C. Froixdevaux. General logical databases and programs, default logic semantics, and stratification. *J. Information and Comput.*, to appear.

[5] H.A. Blair, A.L. Brown, V.S. Subrahmanian. Monotone Logic Programming. Technical Report CS-TR-2375, University of Maryland.

[6] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence* 28:127 – 162, 1986.

[7] R.P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics* 51 (1950) pp. 161-165.

[8] J. Doyle. A Truth Maintenance System. *Artificial Intelligence Journal* 12:231–272, 1979.

[9] D.W. Etherington. Formalizing Nonmonotonic Reasoning Systems. *Artificial Intelligence Journal* 31:41–85, 1987.

[10] M. Gelfond, V. Lifschitz. Stable Semantics for Logic Programs. In: *Proceedings of 5th International Symposium Conference on Logic Programming*, Seattle, 1988.

[11] M. Gelfond, V. Lifschitz. Logic Programming with Classical Negation. Unpublished Manuscript.

[12] M. Gelfond, H. Przymusińska. On the Relationship between Circumscription and Autoepistemic Logic. In: *Proceedings of the ISMIS Conference*, 1986.

[13] M. Gelfond, H. Przymusińska. Inheritance Reasoning in Autoepistemic Logic, Manuscript, 1989.

[14] D. Gries. *The Science of Programming.* Springer-Verlag 1981.

[15] P. Hall On representatives of subsets. *Journal of London Mathematical Society* 10 (1935) pp. 26-30.

[16] M. Hall. Distinct representatives of subsets. *Bulletin of American Mathematical Society* 54 (1948), pp. 922-926.

[17] J.Y. Halpern, Y.O. Moses, Knowledge and Common Knowledge in a Distributed Environment, 3rd ACM Conference on the Principles of Distributed Computing, pp. 50-61.

[18] J. Hintikka *Knowledge and Belief.* Cornell University Press.

[19] W-Q. Huang, A. Nerode. Applications of Pure Recursion Theory to Recursive Analysis. *Acta Sinica* 28.

[20] C.G. Jockusch, T.G. McLaughlin. Countable retracing functions and Π^0_2 predicates. *Pacific Journal of Mathematics* 30 (1972) pp. 69-93.

[21] C.G. Jockusch, R.I. Soare. Π^0_1 classes and degrees of theories. *Transactions of American Mathematical Society* 173 (1972) pp. 33-56.

[22] C.G. Jockusch, R.I. Soare. Degrees of members of Π^0_1 classes. *Pacific Journal of Mathematics* 40 (1972) pp. 605-616.

[23] K. Konolige. On the Relation between Default and Autoepistemic Logic. *Artificial Intelligence* 35:343–382, 1988.

[24] A. Manaster, J. Rosenstein. Effective matchmaking. *Proceedings of the London Mathematical Society* 25 (1972) pp. 615-654.

[25] W. Marek, A. Nerode. Decision procedure for default logic Mathematical Sciences Institute Reports, Cornell University.

[26] W. Marek, V.S. Subrahmanian. The Relationship Between Logic Program Semantics and Non-Monotonic Reasoning. In: *Proceedings of the Fifth International Conference on Logic Programming*, M.I.T. Press.

[27] W. Marek and M. Truszczyński. Autoepistemic Logic, to appear.

[28] W. Marek and M. Truszczyński. Relating Autoepistemic and Default Logics. In: *Principles of Knowledge Representation and Reasoning*, Morgan Kaufman, San Mateo, 1989. (Full version available as Technical Report 144-89, Computer Science, University of Kentucky, Lexington, KY 40506-0027, 1989.)

[29] W. Marek and M. Truszczyński. Stable models for logic programs and default logic. In: *Proceedings of North American Conference on Logic Programming*, MIT Press, 1989. (Full version available as Technical Report, Computer Science Department, University of Kentucky, Lexington, KY 40506-0027, 1989.)

[30] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.

[31] G. Metakides, A. Nerode. Effective Content of Field Theory. *Annals of Mathematical Logic*, 17:289–320.

[32] M. Minsky. A framework for representing knowledge. In: *The Pychology of Computer Vision*, pp.211-272. McGrow Hill.

[33] L. Mirsky. *Transversal Theory.* Academic Press, New York.

[34] R.C. Moore. Semantical Considerations on Non-Monotonic Logic. *Artificial Intelligence*, 25:75–94, 1985.

[35] A. Nerode, J.B. Remmel. A Survey of r.e. Substructures. Proc. Symp. Math. 42, Amer. Math. Soc.

[36] A. Nerode, J.B. Remmel. Complexity-theoretic Algebra I: Vector Spaces over Finite Fields. In: Structures in Complexity. pp. 218-241.

[37] A. Nerode, J.B. Remmel. Complexity-theoretic Algebra II: Boolean Algebras. *Annals of Pure and Applied Logic* 44:71–99.

[38] A. Nerode, J.B. Remmel. Complexity-theoretic Algebra III: Bases of Vector Spaces. In: Feasible Mathematics, Springer Verlag.

[39] M. Reinfrank, O. Dressler. On the Relation between Truth Maintenance and Non-Monotonic Logics. In: *Proceedings of International Joint Conference on Artificial Iintelligence*, 1989.

[40] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[41] J.B. Remmel. Graph colorings and recursively bounded Π^0_1 classes. *Annals of Pure and Applied Logic* 32 (1986) pp. 185-194.

[42] J.B. Remmel. Recursive Boolean Algebras. In: *Handbook of Boolean Algebras*

[43] A. Tarski. *Logic, Semantics, Metamathematics* Oxford, 1956.