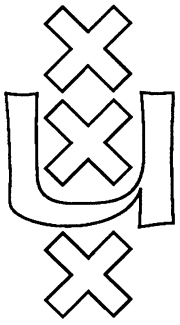
Institute for Language, Logic and Information

GREATEST FIXED POINTS OF LOGIC PROGRAMS

Kees Doets

ITLI Prepublication Series
for Computation and Complexity Theory CT-90-07



University of Amsterdam

The ITLI Prepublication Series

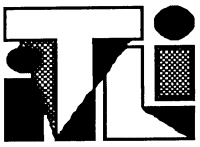
1986
86-01 The Institute of Language, Logic and Information
86-02 Peter van Emde Boas — A Semantical Model for Integration and Modularization of Rules
86-03 Johan van Benthem — Categorial Grammar and Lambda Calculus
86-04 Reinhard Muskens — A Relational Formulation of the Theory of Types
86-05 Kenneth A. Bowen, Dick de Jongh — Some Complete Logics for Branched Time, Part I Well-founded Time,
86-06 Johan van Benthem — Logical Syntax Forward looking Operators

1987 87-01 Jeroen Groenendijk, Martin Stokhof — Type shifting Rules and the Semantics of Interrogatives
87-02 Renate Bartsch — Frame Representations and Discourse Representations
87-03 Jan Willem Klop, Roel de Vrijer — Unique Normal Forms for Lambda Calculus with Surjective Pairing
87-04 Johan van Benthem — Polyadic quantifiers
87-05 Víctor Sánchez Valencia — Traditional Logicians and de Morgan's Example
87-06 Eleonore Oversteegen — Temporal Adverbials in the Two Track Theory of Time
87-07 Johan van Benthem — Categorial Grammar and Type Theory
87-08 Renate Bartsch — The Construction of Properties under Perspectives
87-09 Herman Hendriks — Type Change in Semantics: The Scope of Quantification and Coordination

1988 LP-88-01 Michiel van Lambalgen — *Logic, Semantics and Philosophy of Language:* Algorithmic Information Theory
LP-88-02 Yde Venema — Expressiveness and Completeness of an Interval Tense Logic
LP-88-03 — Year Report 1987
LP-88-04 Reinhard Muskens — Going partial in Montague Grammar
LP-88-05 Johan van Benthem — Logical Constants across Varying Types
LP-88-06 Johan van Benthem — Semantic Parallels in Natural Language and Computation
LP-88-07 Renate Bartsch — Tenses, Aspects, and their Scopes in Discourse
LP-88-08 Jeroen Groenendijk, Martin Stokhof — Context and Information in Dynamic Semantics
LP-88-09 Theo M.V. Janssen — A mathematical model for the CAT framework of Eurotra
LP-88-10 Anneke Kleppe — A Blissymbolics Translation Program
ML-88-01 Jaap van Oosten — *Mathematical Logic and Foundations:* Lifschitz' Realizabiility
ML-88-02 M.D.G. Swaen — The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination
ML-88-03 Dick de Jongh, Frank Veltman — Provability Logics for Relative Interpretability
ML-88-04 A.S. Troelstra — On the Early History of Intuitionistic Logic
ML-88-05 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics
CT-88-01 Ming Li, Paul M.B.Vitanyi — *Computation and Complexity Theory:* Two Decades of Applied Kolmogorov Complexity
CT-88-02 Michiel H.M. Smid — General Lower Bounds for the Partitioning of Range Trees
CT-88-03 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas — Maintaining Multiple Representations of Dynamic Data Structures
CT-88-04 Dick de Jongh, Lex Hendriks, Gerard R. Renardel de Lavalette — Computations in Fragments of Intuitionistic Propositional Logic
CT-88-05 Peter van Emde Boas — Machine Models and Simulations (revised version)
CT-88-06 Michiel H.M. Smid — A Data Structure for the Union-find Problem having good Single-Operation Complexity
CT-88-07 Johan van Benthem — Time, Logic and Computation
CT-88-08 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas — Multiple Representations of Dynamic Data Structures
CT-88-09 Theo M.V. Janssen — Towards a Universal Parsing Algorithm for Functional Grammar
CT-88-10 Edith Spaan, Leen Torenvliet, Peter van Emde Boas — Nondeterminism, Fairness and a Fundamental Analogy
CT-88-11 Sieger van Denneheuvel, Peter van Emde Boas — Towards implementing RL
X-88-01 Marc Jumelet — *Other prepublications:* On Solovay's Completeness Theorem

1989 LP-89-01 Johan van Benthem — *Logic, Semantics and Philosophy of Language:* The Fine-Structure of Categorial Semantics
LP-89-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Predicate Logic, towards a compositional, non-representational semantics of discourse
LP-89-03 Yde Venema — Two-dimensional Modal Logics for Relation Algebras and Temporal Logic of Intervals
LP-89-04 Johan van Benthem — Language in Action
LP-89-05 Johan van Benthem — Modal Logic as a Theory of Information
LP-89-06 Andreja Prijatelj — Intensional Lambek Calculi: Theory and Application
LP-89-07 Heinrich Wansing — The Adequacy Problem for Sequential Propositional Logic
LP-89-08 Víctor Sánchez Valencia — Peirce's Propositional Logic: From Algebra to Graphs
LP-89-09 Zhisheng Huang — Dependency of Belief in Distributed Systems
ML-89-01 Dick de Jongh, Albert Visser — *Mathematical Logic and Foundations:* Explicit Fixed Points for Interpretability Logic
ML-89-02 Roel de Vrijer — Extending the Lambda Calculus with Surjective Pairing is conservative
ML-89-03 Dick de Jongh, Franco Montagna — Rosser Orderings and Free Variables
ML-89-04 Dick de Jongh, Marc Jumelet, Franco Montagna — On the Proof of Solovay's Theorem
ML-89-05 Rineke Verbrugge — Σ-completeness and Bounded Arithmetic
ML-89-06 Michiel van Lambalgen — The Axiomatization of Randomness
ML-89-07 Dirk Roorda — Elementary Inductive Definitions in HA: from Strictly Positive towards Monotone
ML-89-08 Dirk Roorda — Investigations into Classical Linear Logic
ML-89-09 Alessandra Carbone — Provable Fixed points in $I\Delta_0+\Omega_1$
CT-89-01 Michiel H.M. Smid — *Computation and Complexity Theory:* Dynamic Deferred Data Structures
CT-89-02 Peter van Emde Boas — Machine Models and Simulations
CT-89-03 Ming Li, Herman Neuféglise, Leen Torenvliet, Peter van Emde Boas — On Space Efficient Simulations
CT-89-04 Harry Buhrman, Leen Torenvliet — A Comparison of Reductions on Nondeterministic Space
CT-89-05 Pieter H. Hartel, Michiel H.M. Smid, Leen Torenvliet, Willem G. Vree — A Parallel Functional Implementation of Range Queries
CT-89-06 H.W. Lenstra, Jr. — Finding Isomorphisms between Finite Fields
CT-89-07 Ming Li, Paul M.B. Vitanyi — A Theory of Learning Simple Concepts under Simple Distributions and Average Case Complexity for the Universal Distribution (Prel. Version)
CT-89-08 Harry Buhrman, Steven Homer, Leen Torenvliet — Honest Reductions, Completeness and Nondeterminstic Complexity Classes
CT-89-09 Harry Buhrman, Edith Spaan, Leen Torenvliet — On Adaptive Resource Bounded Computations
CT-89-10 Sieger van Denneheuvel — The Rule Language RL/1
CT-89-11 Zhisheng Huang, Sieger van Denneheuvel, Peter van Emde Boas — Towards Functional Classification of Recursive Query Processing
X-89-01 Marianne Kalsbeek — *Other Prepublications:* An Orey Sentence for Predicative Arithmetic
X-89-02 G. Wagemakers — New Foundations: a Survey of Quine's Set Theory
X-89-03 A.S. Troelstra — Index of the Heyting Nachlass
X-89-04 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar, a first sketch
X-89-05 Maarten de Rijke — The Modal Theory of Inequality
X-89-06 Peter van Emde Boas — Een Relationele Semantiek voor Conceptueel Modelleren: Het RL-project

1990 SEE INSIDE BACK COVER



Instituut voor Taal, Logica en Informatie
Institute for Language, Logic and Information

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Plantage Muidergracht 24
1018TV Amsterdam

Faculteit der Wijsbegeerte
(Department of Philosophy)
Nieuwe Doelenstraat 15
1012CP Amsterdam

GREATEST FIXED POINTS OF LOGIC PROGRAMS

Kees Doets
Department of Mathematics and Computer Science
University of Amsterdam

ITLI Prepublication Series
for Computation and Complexity Theory
ISSN 0924-8374

Received November 1990

Greatest Fixed Points of Logic Programs

Kees Doets
Dept. of Mathematics and Computer Science
University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
the Netherlands

Contents.

0. Summary and introduction.

Consider a logic program P and its associated immediate consequence operator T over the corresponding Herbrand base HB. (Cf. Apt [1988] and Lloyd [1987] for unexplained notions.)
We have $T\uparrow=T\uparrow\omega\subset T\downarrow\subset T\downarrow\omega\subset HB$.
For the problem "when is the ground-atom A false?" at least three possible answers have been considered:

1. $A\notin T\uparrow$ (taking the least model seriously as the intended one)
2. $A\notin T\downarrow\omega$ ("finite failure")
3. $A\notin T\downarrow$ ("Herbrand's rule").

So, the following questions are natural: (a) when do these answers coincide, and (b) what is their computational complexity?

Most of this report has been motivated by these questions.
Here is a short description of its contents.
Part 1 gives an upper bound for the downward closure ordinal of a monotone operator. Here and in part 3, the notions of recurrency and weak recurrency and some related results of Bezem [199?] are generalized.
Part 2 describes an infinite game connected with $T\uparrow$, $T\downarrow$ and the corresponding hierarchies.
Part 4 presents a couple of results on weak recurrency of "natural" computing programs; it slightly strengthens Blair's result [Blair 1986] that every total recursive function can be computed by a program with $T\uparrow=T\downarrow\omega$ and presents a more direct proof of Bezem's [199?] strengthening thereof.
Part 5 contains a short, but nevertheless quite complete, proof for Blair's [1982] theorem that $T\downarrow$ can be Σ^1_1-complete.
Part 6 introduces the *canonical mgu-tree* associated with a (fair) SLD-tree. Using it, we extend the finite-failure characterization.
Part 7 discusses non-standard Herbrand universes obtained as direct limits and quotients of free algebras induced by substitutions; it shows that the results of [Blair/Brown 199?] follow from known facts in the theory of recursive saturation.
Note that we deviate from general usage in not writing the implication-sign backwards (and in goals, we often omit it).

1. The downward closure ordinal and recurrency.

Suppose that P is a logic program. Let $T=T_P:\mathcal{P}(HB)\to\mathcal{P}(HB)$ be its associated immediate consequence operator over the Herbrand base HB consisting of all atomic sentences.
Define, for a ground-atom A, $\alpha(A)$ to be the set of all literals occurring in bodies C of ground-instances $C\to A$ of P-rules of which A is the head.
Then obviously, for $X\subset HB$:

[α] $\quad A\in T(X) \Rightarrow A\in T(\alpha(A)\cap X)$.

For the rest of this section we make the

Assumption: $T:\mathcal{P}(U)\to\mathcal{P}(U)$ is an arbitrary monotone operator over U (*not* necessarily associated with a logic program) and $\alpha:U\to\mathcal{P}(U)$ is a function satisfying [α].

It is well-known that if λ is an infinite regular cardinal such that

$$a\in T(X) \Rightarrow \exists Y\subset X[|Y|<\lambda \wedge a\in T(Y)]$$

then $T\uparrow=T\uparrow\lambda$.

However, it need not be true at all that also $T{\downarrow}=T{\downarrow}\lambda$, as operators associated with logic programs emphatically demonstrate.
But, assuming a uniform condition helps:

1.1 Theorem. If λ is regular and $|\alpha(a)|<\lambda$ for all $a\in U$ then $T{\downarrow}=T{\downarrow}\lambda$.
Proof. In order that $T{\downarrow}=T{\downarrow}\lambda$, it suffices that $T{\downarrow}\lambda\subset T(T{\downarrow}\lambda)$ $(=T{\downarrow}\lambda+1)$.
So, assume that $a\in T{\downarrow}\lambda$. Now, $T{\downarrow}\lambda=\bigcap_{\xi<\lambda}T{\downarrow}\xi=\bigcap_{\xi<\lambda}T(T{\downarrow}\xi)$.
Hence, by $[\alpha]$, $a\in\bigcap_{\xi<\lambda}T(\alpha(a)\cap T{\downarrow}\xi)$.
Define $h:\alpha(a)\setminus T{\downarrow}\lambda\to\lambda$ by $h(x):=\min\{\xi|x\notin T{\downarrow}\xi\}$.
Since $|\alpha(a)\setminus T{\downarrow}\lambda|\leqslant|\alpha(a)|<\lambda$ and λ is regular,
$\gamma<\lambda$ exists such that $h:\alpha(a)\setminus T{\downarrow}\lambda\to\gamma$.
Then $\alpha(a)\cap T{\downarrow}\lambda=\alpha(a)\cap T{\downarrow}\gamma$.
Hence, for $\xi\geqslant\gamma$: $\alpha(a)\cap T{\downarrow}\xi\subset T{\downarrow}\lambda$;
therefore $T(\alpha(a)\cap T{\downarrow}\xi)\subset T(T{\downarrow}\lambda)$ and $\bigcap_{\xi<\lambda}T(\alpha(a)\cap T{\downarrow}\xi)\subset T(T{\downarrow}\lambda)$. ⊠

1.2 Remark. For T associated with a logic program, the hypothesis of 1.1 is satisfied with $\lambda=\omega$ when each variable in the body of a rule occurs as well in its head. Cf. Lloyd [1987] p. 67 exercise 7b.

For any well-founded relation $\prec$ on a set U we can define its rank-function $\varrho:U\to OR$ by the equation $\varrho(b)=\sup\{\varrho(a)+1|a\prec b\}$, using recursion along $\prec$.
Then $\sup\{\varrho(b)+1|b\in U\}$ is called the **heigth** of $\prec$.
In the following, we concentrate on the relation $\prec$ defined by

$$a\prec b:\equiv a\in\alpha(b).$$

1.3 Definition.

1. α is **∞-recurrent** iff $\prec$ is well-founded;
2. it is **γ-recurrent** iff $\prec$ is well-founded of height γ.
3. (Bezem [199?]) α is **recurrent** iff it is ω-recurrent.

1.4 Theorem.

1. If $\prec$ is well-founded then $T{\uparrow}=T{\downarrow}$.
2. Precisely, if α is γ-recurrent then $T{\uparrow}\gamma=T{\downarrow}\gamma$.

Proof.
1. Assume that $\prec$ is well-founded. We show that, if $T(I)\subset I$ and $J\subset T(J)$ (which is satisfied, in particular, for $I=T{\uparrow}$ and $J=T{\downarrow}$) then $J\subset I$.
In fact, we show that $a\in J\Rightarrow a\in I$ using induction along $\prec$.
So, assume by way of induction hypothesis that $\forall b\in J[b\prec a\Rightarrow b\in I]$.
This means that $\alpha(a)\cap J\subset I$.
So, $T(\alpha(a)\cap J)\subset T(I)$.
Now if $a\in J$, then $a\in T(J)$ as well (since $J\subset T(J)$).

Hence, if $a \in J$ then $a \in T(\alpha(a) \cap J) \subset T(I)$, using $[\alpha]$.
Part 2. immediately follows from the

1.5 Lemma. If $\prec$ is well-founded then
$\varrho(b) < \xi$ and $b \in T{\downarrow}\xi$ imply $b \in T{\uparrow}\xi$.

Proof. Induction w.r.t. ξ.
The cases $\xi = 0$ and ξ a limit are trivial.
So, assume $\varrho(b) < \xi + 1$ and $b \in T{\downarrow}\xi + 1$.
Now, $T{\downarrow}\xi + 1 = T(T{\downarrow}\xi)$, so, by $[\alpha]$, $b \in T(\alpha(b) \cap T{\downarrow}\xi)$.
But, $\alpha(b) \cap T{\downarrow}\xi \subset T{\uparrow}\xi$ by induction hypothesis.
So, $b \in T(\alpha(b) \cap T{\downarrow}\xi) \subset T(T{\uparrow}\xi) = T{\uparrow}\xi + 1$. ⊠

1.6 Remark. When P is the natural program computing a primitive recursive function (cf. part 4) then the associated relation $\prec$ is easily seen to be well-founded.
Example.
The addition-program

$$\rightarrow +(x,0,x)$$
$$+(x,y,z) \rightarrow +(x,y+1,z+1)$$

has $\prec$ of height ω.
Then the addition of $\times$-computing rules

$$\rightarrow \times(x,0,0)$$
$$\times(x,y,u),\ +(u,x,z) \rightarrow \times(x,y+1,z)$$

preserves height ω (for, the height of $\times(n,m,p)$ is $n+m+1$)
and so $T{\uparrow} = T{\downarrow}\omega$.
However other $\times$-computing rules could be used as well.
Replacing the last one by

$$\times(x,y,u),\ +(x,u,z) \rightarrow \times(x,y+1,z)$$

will also compute $\times$, since we can compute all instances of the commutativity of $+$. However, the height of the corresponding $\prec$ is no longer ω (it is $\omega+\omega$), and we can no longer use 1.4 to prove the fact that, still, $T{\uparrow} = T{\downarrow}\omega$.

1.7 Remark. If $T{\uparrow} = T{\downarrow}\omega$, then P is called **determinate** by Blair [1986], who also proves that every total recursive function can be determinately computed (cf. also part 4 below).
Thus, recurrency implies determinacy (1.4.2 for $\gamma = \omega$), but the converse fails. (For examples, cf. Bezem [199?]. The easiest is $\{p \rightarrow p;\ \rightarrow p\}$; more complicated ones are the second $\times$-computing program above and the natural program which computes the Ackermann-function.)
Note that Bezem [199?] supersedes Blair [1986] by proving that every total recursive function can be computed by a recurrent program (again, cf. part 4).

2. Games related to T↑ and T↓.

We refine condition [α] of part 1.

Again, let P be a logic program and $T:\mathcal{P}(HB)\to\mathcal{P}(HB)$ its immediate consequence operator. Define, for $A\in HB$, σ(A) to be the set of all clauses C (considered as the finite set of its literals) such that $C\to A$ is a ground-instance of a rule of P.

Then

[σ] 1. $C\in\sigma(A) \Rightarrow$ C is finite

2. $C\in\sigma(A) \Rightarrow A\in T(C)$

3. $A\in T(X) \Rightarrow \exists C\in\sigma(A)[C\subset X]$.

Assumption. In the following, $T:\mathcal{P}(U)\to\mathcal{P}(U)$ can be an arbitrary monotone operator and $\sigma:U\to\mathcal{P}(U)$ is a map satisfying [σ].

2.1 Lemma. Defining $\alpha(a):=\bigcup\sigma(a)$ we obtain [α].

2.2 Definition.

1. The infinite game $\Sigma(a)=\Sigma(a,\infty)$ ($a\in U$) has two players CL and AT (for: *clause* and *atom*). A play of the game proceeds as follows. First, CL chooses some $C_0\in\sigma(a)$, then AT chooses an $a_1\in C_0$, after which CL chooses some $C_1\in\sigma(a_1)$, then AT chooses $a_2\in C_1$ again, and so on.

The play stops when one of the players has no move anymore. This can happen either when CL chooses $C_n=\emptyset$ (then CL **wins** AT **loses**) or AT chooses a_n with $\sigma(a_n)=\emptyset$ (AT **wins**, CL **loses**). So, the player with the last move wins. A play which goes on forever is a draw.

2. Let α be an ordinal. $\Sigma(a,\alpha)$ is a game similar to $\Sigma(a,\infty)$, but now, AT, together with each of his ordinary $\Sigma(a,\infty)$-moves a_n, has to choose - on penalty of losing - an ordinal $\alpha_n<\alpha$ as well, in such a way that $\alpha_1>\alpha_2>\alpha_3>\ldots$.

3. If $\alpha<\omega$, the optimal thing to do for AT is choosing $\alpha-n$ as the ordinal-part of his n-th move; and so the ordinal-part of his moves is irrelevant here if we agree that each play shall have length n at the most. (Here, n is the length of a play $a=a_0,C_0,a_1,C_1,\ldots,a_{n-1},C_{n-1}$.)

2.3 Theorem.

1. $\Sigma(a,\infty)$ is a win for CL iff $a\in T\uparrow$;
2. $\Sigma(a,\alpha)$ is a win for CL iff $a\in T\uparrow\alpha$;
3. $\Sigma(a,\infty)$ is a win for AT iff $a\notin T\downarrow$;
4. $\Sigma(a,\alpha)$ is a win for AT iff $a\notin T\downarrow\alpha$.

Proof.

$1\Rightarrow$. Suppose that $a\notin T\uparrow$. Let AT pick his moves - as far as possible - outside T↑. In fact, AT always can play outside T↑: if $a_n\notin T\uparrow$ and $C_n\in\sigma(a_n)$ than $C_n\subset T\uparrow$ is impossible as otherwise $a_n\in T(C_n)\subset T(T\uparrow)=T\uparrow$; and hence

$a_{n+1} \in C_n \setminus T\uparrow$ exists. Playing this way, AT cannot lose. But then CL cannot possibly have a winning strategy.

$1 \Leftarrow$. Suppose that $a \in T\uparrow$; say, $a \in T\uparrow\alpha$, where $\alpha = \rho(a) = \min\{\xi | a \in T\uparrow\xi\}$. So, α is a successor. Now CL can pick $C \in \sigma(a)$ such that $C \subset T\uparrow\alpha - 1$; and hence every $b \in C$ has $\rho(b) < \rho(a)$. This describes a strategy for CL forcing finiteness of the resulting play; and hence, the strategy is winning for CL.

2. Induction w.r.t. α.

$\Sigma(a,\alpha)$ is a win for CL $\Leftrightarrow$ $\exists C \in \sigma(a)\, \forall b \in C\, \forall \beta < \alpha\, [\Sigma(b,\beta)$ is a win for CL$]$

$\Leftrightarrow$ $\exists C \in \sigma(a)\, \forall b \in C\, \forall \beta < \alpha\, [b \in T\downarrow\beta]$

by induction hypothesis

$\Leftrightarrow$ $a \in T\uparrow\alpha$.

$3 \Rightarrow$. Suppose that $a \in T\downarrow$. CL plays - as far as possible - $\subset T\downarrow$. In fact, he can always do so: if $C_n \subset T\downarrow$ then $a_{n+1} \in T\downarrow = T(T\downarrow)$ and hence for some $C \in \sigma(a_{n+1})$: $C \subset T\downarrow$. So, CL can't lose. So, AT cannot have a winning strategy.

$3 \Leftarrow$. Suppose that $a \notin T\downarrow\alpha$ where $\alpha = \rho(a) = \min\{\xi | a \notin T\downarrow\xi\}$. AT now takes care that he plays a sequence of elements of descending ρ-rank.

4. Induction w.r.t. α.

$\Sigma(a,\alpha)$ is a win for AT $\Leftrightarrow$ $\forall C \in \sigma(a)\, \exists b \in C\, \exists \beta < \alpha\, [\Sigma(b,\beta)$ is a win for AT$]$

$\Leftrightarrow$ $\forall C \in \sigma(a)\, \exists b \in C\, \exists \beta < \alpha\, [b \notin T\downarrow\beta]$

$\Leftrightarrow$ $a \notin T\downarrow\alpha$. $\boxtimes$

3. Weak recurrency.

If $a \notin T\uparrow$, then $\Sigma(a,\infty)$ is not a win for CL according to 2.3.1.

In fact, AT has a strategy with which he cannot lose, since we have

[*] $\forall a \notin T\uparrow\, \forall C \in \sigma(A)\, \exists b \in C [\, b \notin T\uparrow\,]$

and so all AT has to do in order not to lose is staying outside $T\uparrow$.

Unfortunately, this strategy cannot guarantee him a win since a play may go on forever (and in fact, if $a \in T\downarrow$, then CL can see to that by always playing $\subset T\downarrow$).

Fortunately though, the only thing that is needed for AT to win $\Sigma(a,\infty)$ (with $a \notin T\uparrow$) by just playing outside $T\uparrow$ is that each play should always terminate.

For then, since the last move will be AT's by [*], CL must lose.

Now, termination can be guaranteed if a map $l: U \to OR$ (OR the collection of all ordinals) exists such that we have the following stronger alternative to [*]:

[*,l] $\forall a \notin T\uparrow\, \forall C \in \sigma(a)\, \exists b \in C [b \notin T\uparrow \wedge\, l(b) < l(a)]$

since then AT simply plays a sequence that is l-descending.

This motivates the following

3.1 Definition.

1. σ is ∞-**weakly recurrent** if l:U→OR exists such that [*,l] holds.
2. σ is α-**weakly recurrent** if l:U→α exists for which [*,l].
3. (Bezem [199?])
   σ is **weakly recurrent** iff it is ω-weakly recurrent.
4. A program is (∞/α-) weakly recurrent iff the induced σ is.

3.2 Lemma. If σ is ∞-weakly recurrent w.r.t. l then, if k⩾l(a):

$$a\epsilon T{\downarrow}k \Rightarrow a\epsilon T{\uparrow}.$$

Proof. Immediate from the discussion above.
If $a\notin T{\uparrow}$ then, by ∞-weak recurrency, clearly Σ(a,l(a)) is a win for AT, whence $a\notin T{\downarrow}l(a)$ by 2.3.4.
As $k\geqslant l(a)$, $T{\downarrow}k\subset T{\downarrow}l(a)$. Hence, $a\notin T{\downarrow}k$. ⊠

3.3 Corollary. ([Bezem 199?] for α=ω.)
If σ is α-weakly recurrent w.r.t. l:U→α then $T{\uparrow}=T{\downarrow}\alpha=T{\downarrow}$.

3.4 Theorem. The following two statements are equivalent:

1. $T{\downarrow}\alpha=T{\uparrow}$
2. σ is α-weakly recurrent.

In particular ([Bezem 199?]) a program is determinate iff it is weakly recurrent.
Proof. 1⇒2: let l be the rank-function associated with the T↓-hierarchy, i.e., define l on $U\backslash T{\uparrow}\subset U\backslash T{\downarrow}\alpha$ by $l(a):=\min\{\xi|a\notin T{\downarrow}\xi\}$. ⊠

4. Computation of "recursively defined" operations.

Although 3.2/3/4 may appear to be too obvious to be interesting, in fact they can be quite helpful, as the results below indicate.
We show that the natural programs for recursively defined operations are weakly recurrent with respect to an obvious map and hence determinate. [Blair 1986] shows that every total recursive function can be determinately computed. [Bezem 199?] has the even better result that every total recursive function can be recurrently computed. Below, we reprove these results, using a simpler implementation for μ, showing left-rule termination of the resulting programs and termination for their variants under a simple transformation.
To be able to discuss computation of relations and functions over some domain, some representation in the Herbrand universe HU of closed terms of the objects of the domain is needed. In the case of ℕ, the usual way to achieve this is by means of an individual constant 0 (for zero) and a unary

function symbol S (for successor). In the sequel however, we don't distinguish between an object and its representation in the Herbrand universe and we only discuss computation of relations and functions over HU, part of which may be identified with $\mathbb{N}$, for instance, using 0 and S. The usual definition now reads as follows.

4.1 Definition.

1. The program P **computes** the relation $r \subset HU^k$ **in** the symbol R iff for all $t_1,\dots,t_k \in HU$: $r(t_1,\dots,t_k) \Leftrightarrow R(t_1,\dots,t_k)\in T\uparrow$.
2. P **computes** $f:HU^k \to HU$ **in** F iff for all $t_1,\dots,t_k,s \in HU$:
   $f(t_1,\dots,t_k)=s \Leftrightarrow F(t_1,\dots,t_k,s)\in T\uparrow$,
   i.e., iff P computes the *graph* of f in F.

Instead of 4.1.2, a somewhat different notion could be considered as well:

4.2 Definition.

P **strongly computes** f in F iff for all $t_1,\dots,t_k \in HU$:
P refutes the goal $F(t_1,\dots,t_k,y)\to$ with computed answer $y=f(t_1,\dots,t_k)$.

However

4.3 Lemma. If 0 is the only individual constant and S the only function symbol then P computes f in F iff it does so strongly.

Proof. First, suppose that P strongly computes f in F.
First, if $f(n)=m$, then $y=m$ is a computed answer for the goal $F(n,y)\to$ by strong computability and hence, by soundness, $F(n,m)\in T\uparrow$.
Second, if $F(n,m)\in T\uparrow$ then $\lambda:=\{y/m\}$ is correct for the goal $F(n,y)\to$.
Therefore, by strong completeness, P refutes $F(n,y)\to$ with an answer $\theta:=\{y/t\}$ more general than λ. By assumption however, $t=f(n)$. Since $\lambda=\theta\delta$ for some δ, $m=y\lambda=y\theta\delta=f(n)\delta=f(n)$.
Conversely, assume that P computes f in F.
Assume $m=f(n)$. We have to show that P refutes $F(n,y)\to$ with the answer $y=m$. Since $m=f(n)$, $F(n,m)\in T\uparrow$ by hypothesis and hence P refutes $F(n,m)\to$ by completeness. By strong completeness however, P refutes $F(n,y)\to$ with an answer $\theta=\{y/t\}$ more general than $\lambda:=\{y/m\}$. Since 0 and S are the only constant/function symbol, there are only two possibilities for t.
(i) $t=S^k y$ for some k. But then we would have $F(n,k+m+1)\in T\uparrow$ as well, entailing the impossibility $f(n)=k+m+1>m=f(n)$.
(ii) $t=S^k 0$. But then, since θ is more general than λ, $k=m$ and $t=m$. ⊠

4.4 Definition.

(1). $f:\mathbb{N}^k\to\mathbb{N}$ is **primitive recursive** if there is a list $f_0,\dots,f_m=f$ (the **p.r. definition** of f) such that for each $k\leqslant m$ one of 1-5 hold:

1. $f_k=\lambda n.n+1$
2. $f_k=\lambda n_0,\dots,n_{i-1}.n_j$ for some $j<i$
3. $f_k=\lambda n.0$

4. $f_k = \lambda \underline{n}.f_j(f_{j(0)}(\underline{n}),\ldots,f_{j(p-1)}(\underline{n}))$ where $j,j(0),\ldots,j(p-1)<k$ ("composition")

5. f_k is the unique function satisfying

$f_k(\underline{n},0)=f_j(\underline{n})$

$f_k(\underline{n},p+1)=f_i(\underline{n},p,f_k(\underline{n},p))$ where $j,i<k$ ("primitive recursion").

(2). If $f_0,\ldots,f_m=f$ is a p.r. definition of the primitive recursive function f, the **natural program** associated with it is obtained (cf. Apt [1988]) by transforming equations 1-5 above into rules in the following, straightforward, way:

1. $\rightarrow F_k(x,x+1)$
2. $\rightarrow F_k(x_0,\ldots,x_{i-1},x_j)$
3. $\rightarrow F_k(x,0)$
4. $F_{j(0)}(\underline{x},y_0),\ldots,F_{j(p-1)}(\underline{x},y_{p-1}),F_j(y_0,\ldots,y_{p-1},z) \rightarrow F_k(\underline{x},z)$
5. $F_j(\underline{x},y) \rightarrow F_k(\underline{x},0,y)$

$F_k(\underline{x},y,z),F_i(\underline{x},y,z,w) \rightarrow F_k(\underline{x},y+1,w).$

Obviously, the canonical program computes f_j in F_j $(j \leqslant m)$.

4.5 Theorem. The natural program associated with a p.r. definition is determinate.

Proof. Let P be the program constructed from the p.r. definition $f_0,\ldots,f_m$. We want to use 3.3, i.e., we have to define a map $l:HB\rightarrow\mathbb{N}$ w.r.t. which P is weakly recurrent.

For $j \leqslant m$, define $l(F_j(\underline{n},m))$ to be *the length of the calculation* of $f_j(\underline{n})$, according to the p.r. definition $f_0,\ldots,f_j$ of f_j.

Notice that $l(F_j(\underline{n},m))$ does not depend on the "value" m but only on the "arguments" $\underline{n}$.

By 3.3, it suffices to prove the

Claim. P is weakly recurrent w.r.t. l.

So, let $A \notin T\uparrow$ be a ground atom and $C \in \sigma(A)$; i.e., $C \rightarrow A$ is ground-instance of a P-rule. We have to find $B \in C$ such that $l(B)<l(A)$.

Now, the only rules with non-empty bodies are the ones associated with composition and recursion. In the following we simplify.

1. $A=F(n,m)$; $C \rightarrow A$ is: $G(n,u),H(u,m)\rightarrow F(n,m)$. (Composition)

Assume that P computes g,h,f in resp. G, H and F.

By hypothesis, $F(n,m) \notin T\uparrow$; hence, $f(n) \neq m$.

(i). $G(n,u) \notin T\uparrow$. We can take $B:=G(n,u)$, since, in order to calculate f(n), one has - according to the p.r. definition of f - to calculate g(n) first; i.e., $l(G(n,u))<l(F(n,m))$.

(ii). $G(n,u) \in T\uparrow$. Hence, $g(n)=u$.

Now if $H(u,m) \in T\uparrow$, then $h(u)=m$ and hence $f(n)=h(g(n))=h(u)=m$.

But, $f(n) \neq m$. So, $H(u,m) \notin T\uparrow$. We now can take $B:=H(u,m)$ as $l(H(u,m))<l(F(n,m))$: in order to calculate f(n) according to the p.r.

definition, one has to calculate h(u) first (u being g(n)).

2. $C \rightarrow A$ is

$G(n,m) \rightarrow F(n,0,m)$. Then $G(n,m) \notin T\uparrow$ and $l(G(n,m)) < l(F(n,0,m))$.

3. $C \rightarrow A$ is

$F(n,p,u), H(u,m) \rightarrow F(n,p+1,m)$.

Again, there are two cases.

(i). $F(n,p,u) \notin T\uparrow$. Then $l(F(n,p,u)) < l(F(n,p+1,m))$.

(ii). $F(n,p,u) \in T\uparrow$. Then $H(u,m) \notin T\uparrow$ and $l(H(u,m)) < l(F(n,p+1,m))$. ⊠

4.6 Remarks.

1. Of course, the argument of 4.5 can be used for every recursively defined function, where we can take this term in a very broad sense.
For an example, cf. the proof of 4.10 below.

2. For another example, Bezem [199?] notes that the natural program computing Ackermann's function is not ω-recurrent; however, our choice of l easily establishes its weak recurrency.

3. Suppose that P computes the function f in the relation symbol F acording to 4.1.2. We call $t_1,\ldots,t_k$ the **arguments** of $F(t_1,\ldots,t_k,s)$ and s its **value**.
Now, it is easily verified that the natural programs for p.r. functions terminate on goals with arguments ground under the Prolog-selection-rule which always selects the left-most atom.

General recursive functions are obtained in a way entirely similar to the primitive recursive ones, now admitting the minimalisation-schema which obtains the (possibly, partial) function f from g by the definition $f(\underline{n}) := \mu m[g(\underline{n},m)=0]$.
However, minimalisation spoils the case for the "natural" programs.

4.7 Definition.

Suppose that $f(n) = \mu m[g(n,m)=0]$. Suppose that P computes g in G.
The **natural rules** ([Apt 1988]) computing f in F are the following three.
First, define the relation o by $o(n,m) :\equiv \forall m' < m[g(n,m') > 0]$.
The following two rules compute o in O:

$$\rightarrow O(x,0)$$
$$O(x,y), G(x,y,Sz) \rightarrow O(x,y+1).$$

Now, add

$$O(x,y), G(x,y,0) \rightarrow F(x,y).$$

Clearly, these rules compute f in F.

4.8 Example.

$g:\mathbb{N} \rightarrow \mathbb{N}$ is defined by: $g(k)=0$ if k is a square, $g(k)=1$ otherwise. So, between the zeros n^2 and $(n+1)^2$ of g are $(n+1)^2 - n^2 - 1 = 2n$ values 1.

Suppose that P determinately computes g in G.
Consider the extended program, canonically computing the argument-less function $f:=\mu y[gy=0]$ using the rules from 4.7. So, $F(n)\in T\uparrow \Longleftrightarrow n=0$.
Notice:
1. $F((n+1)^2)\in T\downarrow(2n)$; 2. $F(k)\in T\downarrow\omega \Longleftrightarrow k=0$; 3. $T\downarrow=T\downarrow\omega$.
Add the rule

$F(Sx)\rightarrow Q$

which is a canonical composition-rule computing the argument-less Q.
Now, for the associated T': $Q\in T'\downarrow\omega\setminus T'\downarrow(\omega+1)$; $T'\downarrow=T'\downarrow(\omega+1)$.
Determinacy is lost.

The *total* recursive functions are obtained in the same way as the general recursive ones by restricting application of the μ-schema:
$f(n):=\mu m[g(n,m)=0]$, to functions g satisfying the existence-condition $\forall m\exists n[g(n,m)=0]$.

4.9 Theorem. For the natural programs (4.4/7) computing total recursive functions: $T\downarrow=T\uparrow$.

Proof. Theorem 1.4. ⊠

Here is a proof of Blair's [1986]

4.10 Theorem.

Every total recursive function can be computed by a determinate program.

Proof. This is an immediate consequence of the following theorem which shows that there is an alternative to the "natural rules" of 4.7 which preserves determinacy.

4.11 Theorem. Suppose that

1. $g:\mathbb{N}\rightarrow\mathbb{N}$ satisfies $\forall n\exists m[g(n,m)=0]$
2. $f:\mathbb{N}\rightarrow\mathbb{N}$ is defined by the minimalisation $f(n):=\mu m[g(n,m)=0]$
3. the program P computes g in G; $T=T_P$ is its immediate consequence operator
4. $l:HB\rightarrow\mathbb{N}$ is such that
   (i) for all n,m: $l(G(n,m))=l(G(n,0))$
   (i.e., $l(G(n,m))$ does not depend on the "value" m of $G(n,m)$)
   (ii) $G(n,m)\in T\downarrow l(G(n,m))$ implies $G(n,m)\in T\uparrow$
   (i.e., l "witnesses weak recurrency of P w.r.t. G-atoms").

Then P can be extended to a program computing f in a new symbol F in such a way that an extension l' of l to F-atoms exists with,
for all n,m:

(i) $l'(F(n,m))\in\mathbb{N}$ (l' is finite)
(ii) $l'(F(n,m))=l'(F(n,0))$ (independence of m)
(iii) $F(n,m)\in T\downarrow l'(F(n,m))$ implies $F(n,m)\in T\uparrow$ (weak recurrency).

(In fact, l' will witness *recurrency* of the *new* rules.)

Proof.

Define the relation zero$\subset \mathbb{N}^3$ by:

$$\text{zero}(n,i,j) :\equiv \forall j' < j\,[\,g(n,i+j') \neq 0\,] \wedge g(n,i+j) = 0.$$

I.e., zero(n,i,j) holds iff i+j is the first zero $\geqslant$ i of the function $\lambda m.g(n,m)$.

Here is a recursion defining zero (the recursion is on the last argument):

$$\text{zero}(n,i,0) \iff g(n,i) = 0$$
$$\text{zero}(n,i,j+1) \iff g(n,i) \neq 0 \wedge \text{zero}(n,i+1,j).$$

Of course, this is *not* a *primitive* recursion since the second argument has not been kept fixed.

Nevertheless, here are program-rules which, upon addition to the program P, obviously *compute* zero in the symbol ZERO

(n,i,u,j are meant to be variables here):

$$G(n,i,0) \rightarrow ZERO(n,i,0)$$
$$G(n,i,u+1),\ ZERO(n,i+1,j) \rightarrow ZERO(n,i,j+1).$$

Extend l to l* on ZERO-atoms by the recursion (n,i,j *numbers* this time):

$$l^*[ZERO(n,i,0)] = 1 + l[G(n,i,0)]$$
$$l^*[ZERO(n,i,j+1)] = 1 + \max\{l[G(n,i,0)],\ l^*[ZERO(n,i+1,j)]\}.$$

It is easily seen that l* witnesses weak recurrency of the rules w.r.t. ZERO-atoms in the sense of the theorem.

Finally add the following rule which can, without difficulty, be seen to compute f in F:

$$ZERO(n,0,m) \rightarrow F(n,m).$$

It remains to construct a map l' on F-atoms witnessing weak recurrency and not depending on F-"values".

(Note that the obvious map $l'(F(n,m)) := 1 + l^*(ZERO(n,0,m))$ *does* depend on m.)

Define l' by

$$l'(F(n,m)) := 1 + \max[\{1 + f(n) + l[G(n,f(n),0)]\} \cup \{l^*[ZERO(n,0,m))] \mid m < f(n)\}].$$

By hypothesis 1., f(n) is a well-defined number, and hence so is l'(F(n,m)).

Also, notice that l'(F(n,m)) does not depend on m.

Therefore, our theorem has been proved if we can show:

If $k+1 \geqslant l'(F(n,m))$ and $F(n,m) \in T{\downarrow}k+1$ then $F(n,m) \in T{\uparrow}$.

Proof. Assume that $k+1 \geqslant l'(F(n,m))$ and $F(n,m) \in T{\downarrow}k+1$.

Then, by the F-rule, $ZERO(n,0,m) \in T{\downarrow}k$.

Now, $ZERO(n,0,m) \in T{\uparrow}$ clearly amounts to: $m = f(n)$.

Instead of showing $m = f(n)$, we show that $m < f(n)$ and $m > f(n)$ are impossible.

a. $m < f(n)$.

Then, by definition of l', $k \geqslant l^*(ZERO(n,0,m))$.

Since l* witnesses weak recurrency of the ZERO-computing rules, $ZERO(n,0,m) \in T{\uparrow}$, i.e., $m = f(n)$; contradicting $m < f(n)$.

Thus, $m<f(n)$ is impossible.

b. $m>f(n)$.

Since $ZERO(n,0,m)\epsilon T\downarrow k$, by the second ZERO-rule, there exists u_0 such that

$$G(n,0,Su_0),\ ZERO(n,1,m-1)\epsilon T\downarrow k-1.$$

Assuming $m-1>0$, by the second ZERO-rule again, there exists u_1 such that

$$G(n,1,Su_1),\ ZERO(n,2,m-2)\epsilon T\downarrow k-2.$$

A third application of the second ZERO-rule (as long as $m-2>0$) provides us with a u_2 such that

$$G(n,2,u_2),\ ZERO(n,3,m-3)\epsilon T\downarrow k-3;$$

We proceed, until we obtain at the $f(n)$-th step a number u such that

$$G(n,f(n)-1,Su),\ ZERO(n,f(n),m-f(n))\epsilon T\downarrow k-f(n);$$

By assumption $m>f(n)$, hence $m-f(n)$ is still positive; and so we apply the second ZERO-rule a last time, obtaining v such that

$$G(n,f(n),Sv),\ ZERO(n,f(n)+1,m-f(n)-1)\epsilon T\downarrow k-f(n)-1.$$

Now, by definition of 1', $k-f(n)-1\geqslant 1(G(n,f(n),Sv))$; and so $G(n,f(n),Sv)\epsilon T\uparrow$, i.e., $g(n,f(n))=v+1$ - a contradiction, since $g(n,f(n))=0$.

So, $m>f(n)$ is impossible and our proof is complete. ⊠

4.12 Remark on the proof of 4.11.

From the definition of 1* it follows, by induction on j, that

$$1^*(ZERO(n,i,j))\leqslant\Sigma_{0\leqslant k\leqslant j}[k+1(G(n,i+k,0))].$$

In particular,

$$1^*(ZERO(n,0,m))\leqslant\Sigma_{0\leqslant k\leqslant m}[k+1(G(n,k,0))].$$

From the definition of 1' it now follows, that

$$1'(F(n,m))\leqslant\Sigma_{0\leqslant k\leqslant f(n)}[k+1(G(n,k,0))],$$

and so we could have defined

$$1'(F(n,m))=\Sigma_{0\leqslant k\leqslant f(n)}[k+1(G(n,k,0))] \text{ outright.}$$

Of course, this just expresses that 1' here has the same *meaning* as the map defined in the proof of 4.5.

Bezem [199?] proves that every total recursive function can even be recurrently computed. His proof uses a translation from the register-machine model to logic programs due to Shepherdson.

Below we present a more direct proof of this theorem using a translation *from logic programs* to logic programs, based on the proofs of theorems 4.5 (weak recurrency of the canonical p.r.-computing programs) and 4.11 (weak recurrency of our implementation of the μ-schema for total recursiveness) above.

Note that the rules proposed in 4.5/11 are such that, when applied with the Prolog "left-most"-selection-rule to a goal in whch the arguments are *variable-free,* they *never will produce an infinite SLD-derivation,* i.e.,

the "left-most"-rule will, when applied to such goals, always lead to termination of the SLD-resolution-process: remark 4.6.3 extends to the zero-computing rules.
The reason that this is so is simply, that (i) left-most selection will never introduce variables on argument-places (just inspect the rules - variables introduced on argument-places in bodies of rules - which only happens with composition and recursion - always occur in atoms which are *not* left-most) and (ii) the rules are weakly recurrent w.r.t. a map which only depends on arguments.
(N.B.: Compare the μ-rules given in 4.7. Notice that they behave bad in this respect: the rule $O(x,y),G(x,y,O)\rightarrow F(x,y)$ transfers the value y in $F(x,y)$ to an argument-place in $G(x,y,O)$.)
To be able to circumvent the problem of a variable argument in a body transferred from a value-place in a head (alternatively, the problem that an unfortunate selection may not lead to termination), we shall transform the program for a total recursive function in such a way that "goals will become single atoms and atoms will become terms".
This needs a little coding.

4.13. Coding sequences and numbers.

Our transformation translates an arbitrary logic program P in a language L into a program P* of which the language is $L\cup\{nil, cons, GOAL\}$ where nil is a new individual constant, cons a new two-place function symbol and GOAL a new one-place relation symbol.
In connection with nil and cons, we need the following standard abbreviations.

4.13.1. Think of nil as representing the empty sequence.
Writing cons(t,s), think of s as denoting a (finite) sequence; cons(t,s) then will denote the sequence obtained from s by the addition of t as a new element in front of it.
So, we have the following notations:

1. $[t|s]:=cons(t,s)$.
2. $[t]:=[t|nil]$. (Sequence of length one.)

And of course

3. $[t_1,\ldots,t_n|s]:= [t_1,[t_2,\ldots,[t_n|s]\ldots]]$.

(Concatenation of sequences where the first one is completely specified.)
In particular:

4. $[t_1,\ldots,t_n]:=[t_1,[t_2,\ldots,[t_n]\ldots]]$.

(Completely specified sequences of given length.)
In order to avoid the cluttering-up of [and] and make for somewhat easier reading, below I shall use < and > for the same purpose as [and].

4.13.2 Coding numbers.

5. $\ulcorner 0 \urcorner := \text{nil}$
6. $\ulcorner n+1 \urcorner := [\ulcorner n \urcorner | \ulcorner n \urcorner]$ $(= [\ulcorner n \urcorner, \ulcorner n-1 \urcorner, \ulcorner n-2 \urcorner, \ldots, \ulcorner 0 \urcorner])$.

(Defining $\ulcorner n+1 \urcorner := [\ulcorner n \urcorner]$ would work as well.)

4.14 The transformation.

Suppose that P is an arbitrary logic program in a language L.
Assume that $R_1, \ldots, R_m$ is a complete list of all relation symbols of L.

4.14.1 Translating rules.

Suppose that

$$R_i(t_1,\ldots,t_n), \ldots, R_j(s_1,\ldots,s_k) \rightarrow R_k(u_1,\ldots,u_p)$$

is any rule of P.
The **translation** of this rule (using nil, cons and GOAL) now reads

$$\text{GOAL}(\langle[\ulcorner i \urcorner, t_1,\ldots,t_n], \ldots, [\ulcorner j \urcorner, s_1,\ldots,s_k] | w\rangle) \rightarrow \text{GOAL}(\langle[\ulcorner k \urcorner, u_1,\ldots,u_p] | w\rangle).$$

Notice that, in the translated rule, atoms and clauses of the old rule have been coded into terms in an obvious fashion; the R_i are identified by their indices. Note the new variable w.

4.14.2 Definition of the transformation.

The **transformed program** P* consists of all translations of the P-rules, together with the following m+1 rules:
for i=1,...,m, we have the following **start-rule**:

$$\text{GOAL}(\langle[\ulcorner i \urcorner, x_1,\ldots,x_n]\rangle) \rightarrow R_i(x_1,\ldots,x_n)$$

(assuming R_i to be n-place);
finally we have the one bodyless **end-rule**:

$$\rightarrow \text{GOAL}(\text{nil}).$$

4.15 Theorem.

Suppose that P is left-rule terminating on a class of atomic goals AG.
Then P* is terminating on AG with the same results as P.

I.e.:
P* produces a finite SLD-tree for every goal N in AG; the derivation fails iff P fails finitely on N under the left-rule and it succeeds iff P succeeds on N left-most, yielding the same answer-substitutions as does P.
Proof. Each derivation from P* starting with an initial goal in the language L corresponds in a unique fashion to a left-rule SLD-derivation from P. The point of the translation is forcing left-rule application, so to speak. ⊠

The application of this theorem to programs computing total recursive functions which are left-rule terminating on atomic goals with ground arguments yields Bezem's [199?]

4.16 Theorem. Every total recursive function can be computed by a recurrent program.

Proof. Apply * to a suitable left-rule terminating program P. Notice that P* will terminate on *all* atomic ground goals of the extended language. ⊠

4.17 Remark. The translation effortlessly transforms the natural weakly recurrent program for the Ackermann-function into a recurrent one. Cf. [Bezem 199?] example 3.2, where a different recurrent program is obtained by means of ingenuity, however, with additional niceties.

5. Proof of a theorem of Blair.

5.0. By its definition $T\downarrow=\bigcup\{X\subset HB|X\subset T(X)\}$, $T\downarrow$ is Σ^1_1 at the worst. It is a remarkable result of Blair [1982] that $T\downarrow$ actually *can* be as complicated as complete-Σ^1_1.
The argument in [Blair 1982] is hard to follow and the sketch of [Blair 1986] is rather incomplete. Below follows a short proof of Blair's theorem, the main lemma of which (cf. 5.5) may have some interest of its own.

5.1. The usual computability-notion for logic programs is defined in terms of the *least* fixed point: P **computes** the relation r on the Herbrand-universe HU of closed terms **in** the symbol R if: $r(t,\ldots)\Leftrightarrow R(t,\ldots)\in T\uparrow$.
We'll now say that P **co-computes** it if: $r(t,\ldots)\Leftrightarrow R(t,\ldots)\notin T\downarrow$.
Note that, if $T\uparrow=T\downarrow$ - or even, if $R(t,\ldots)\in T\uparrow\Leftrightarrow R(t,\ldots)\in T\downarrow$ - then P computes r in R iff it co-computes $\neg r$ in R. This holds in particular when P is determinate.

5.2. Since greatest fixed points - as opposed to their least counterparts - often seem to be difficult to grasp, note the well-known fact that they are *complements* of *least* fixed points: if $T:\mathcal{P}(U)\rightarrow\mathcal{P}(U)$ is monotone, define its **dual** $T^c:\mathcal{P}(U)\rightarrow\mathcal{P}(U)$ by $T^c(X):=U\setminus T(U\setminus X)$. T^c is monotone as well. It is easily checked, that for all ξ: $T\downarrow\xi=U\setminus T^c\uparrow\xi$ and hence $T\downarrow=U\setminus T^c\uparrow$. Finally, note that $T^{cc}=T$.

5.3. This section will not be needed; however, the reader may wonder how the program-rules below were constructed. In that case, the following might be illuminating.
If T is the operator of a logic program P, we may ask a seemingly silly question: is there a "program" P^c (of some sort) of which T^c is the operator? The answer is: yes! And it can be obtained fairly easily from P as follows.
To make things easier, assume that P has, for each relation it defines, only *one* rule and that, moreover, variables are the *only* terms rule-heads contain.
Note, that the first restriction really is immaterial: we can always combine two rules $C\rightarrow R(x)$ and $D\rightarrow R(x)$ into the one rule $C\vee D\rightarrow R(x)$ (cf. the procedure for obtaining the *completion* of a program). So, we can satisfy the first assumption if we admit arbitrary *positive quantifier-free* bodies of rules, i.e., bodies made up of atoms using $\wedge$ and $\vee$.
The second restriction can be satisfied transforming a rule $C\rightarrow R(t)$ into $C\wedge y=t\rightarrow R(y)$. However, the discussion only serves as a guideline for finding additional rules without function symbols in the heads co-computing some new relations, and so we don't need equality.
Finally, if $C(x,y)\rightarrow R(x)$ is a rule, logically speaking it won't change if we

add the "hidden existential quantifier": $\exists y C(x,y) \to R(x)$.

Now we have the

Fact: the corresponding co-rule is obtained by

(i) simultaneously changing in C $\wedge$ to $\vee$ and $\vee$ to $\wedge$ everywhere, and

(ii) changing $\exists$ to $\forall$.

The result has the form $\forall y C^c(x,y) \to R(x)$ where C^c again is positive quantifier-free.

This is quite easy to prove taking into account the definition of T^c - but, since below we will use this only twice and check things there thoroughly, we leave this to the reader.

The main point behind Blair's result now appears to be that a *universal* quantifier in the body of a rule can be (very) much more powerful than an *existential* one.

Below, we will draw attention to this fact at the appropriate place.

5.4. We need the following well-known

Lemma. Let $\prec$ be any relation on a set W.

Define the monotone operator $\Phi: \mathcal{P}(W) \to \mathcal{P}(W)$ by

$\Phi(X) := \{\alpha \in W \mid \forall \beta \prec \alpha [\beta \in X]\}$.

Then $\Phi\uparrow = Wf(W,\prec)$, where $Wf(W,\prec)$ is the **well-founded part** of $\prec$, that is: the largest $\prec$-initial of W on which $\prec$ is well-founded.

Proof. Of course, $\prec$ is well-founded on V iff $\prec$-*induction on* V is valid:

if $\forall \alpha \in V[\forall \beta \prec \alpha(\beta \in V \Rightarrow \beta \in X) \Rightarrow \alpha \in X]$ then $\forall \alpha \in V[\alpha \in X]$.

(i) If V is an $\prec$-initial of W on which $\prec$ is well-founded then $V \subset \Phi\uparrow$, i.e., $\forall \alpha \in V[\alpha \in \Phi\uparrow]$ follows using $\prec$-induction on V (putting $X := \Phi\uparrow$): suppose that $\alpha \in V$ and $\forall \beta \prec \alpha(\beta \in V \Rightarrow \beta \in \Phi\uparrow)$ (induction hypothesis). Then $\forall \beta \prec \alpha[\beta \in \Phi\uparrow]$ since V is an $\prec$-initial of W and $\alpha \in V$; i.e., $\alpha \in \Phi(\Phi\uparrow) = \Phi\uparrow$.

(ii) $\Phi\uparrow$ is an $\prec$-initial of W: if $\alpha \in \Phi\uparrow$ then $\alpha \in \Phi(\Phi\uparrow)$, hence $\alpha \in W$ and $\forall \beta \prec \alpha(\beta \in \Phi\uparrow)$.

(iii) Finally, $\prec$ is well-founded on $\Phi\uparrow$:

Assume that $\forall \alpha \in \Phi\uparrow[\forall \beta \prec \alpha(\beta \in \Phi\uparrow \Rightarrow \beta \in X) \Rightarrow \alpha \in X]$.

I.e., $\forall \alpha \in W[\forall \beta \prec \alpha(\beta \in Y) \Rightarrow \alpha \in Y]$, where $Y := (W \setminus \Phi\uparrow) \cup X$.

Then $\Phi(Y) \subset Y$ by definition of Φ, hence $\Phi\uparrow \subset Y$ since $\Phi\uparrow$ is the least such Y, whence $\Phi\uparrow \subset X$. $\boxtimes$

Relation to the discussion in 5.3: Note that the following "program rule" completely corresponds to the inductive definition of $\Phi\uparrow = Wf(W,\prec)$:

$$\forall \beta[\beta \prec \alpha \to \beta \in \Phi\uparrow] \wedge \alpha \in W \to \alpha \in \Phi\uparrow.$$

This amounts to $\forall \beta[\neg \beta \prec \alpha \vee \beta \in \Phi\uparrow] \wedge \alpha \in W \to \alpha \in \Phi\uparrow$, which has the co-rule-form of 5.3, *provided* we can read $\ulcorner \neg \beta \prec \alpha \urcorner$ *positively*. (Cf. the hypothesis of theorem 5.5.) This is the motivating guideline behind the following section. The corresponding, ordinary program-rule then reads (cf. the fact of 5.3)

$$\exists\beta[\neg\beta\prec\alpha\wedge\beta\in\Phi\uparrow]\vee\alpha\in W\rightarrow\alpha\in\Phi\uparrow.$$

The disjunction here is responsible for the splitting of this rule into wf1 and wf2 in the next subsection.

As a consequence, we have the following

Remark. The *dual* Φ^c of the canonical operator Φ used in showing that $Wf(W,\prec)$ is *positive elementary inductively definable* in W and the complement of $\prec$ (cf. Moschovakis [1974] for this terminology, the content of which however is obvious here) is continuous and, hence, has closure ordinal ω.

5.5. We are going to co-compute well-founded parts.

Theorem. Suppose that $W\subset HU$ and that $\prec$ is a relation on W.

Assume that the program P *co*-computes (!*) W and *the complement* (!!**) of $\prec$ in the symbols nw and pr, respectively.

Then the addition of the following two rules

wf 1: $pr(\beta,\alpha)\wedge wf(\beta)\rightarrow wf(\alpha)$

wf 2: $nw(\alpha)\rightarrow wf(\alpha)$

to P produces a program Q which co-computes $Wf(W,\prec)$ in wf.

Proof. Let T and S be the operators associated with P resp. Q.

It suffices to prove the

Claim. $wf(\alpha)\in S^c\uparrow\Leftrightarrow\alpha\in Wf(W,\prec)$,

since then $wf(\alpha)\notin S\downarrow\Leftrightarrow wf(\alpha)\in S^c\uparrow\Leftrightarrow\alpha\in Wf(W,\prec)$.

Remember, that, for $\Phi(X):=\{\alpha\in W|\forall\beta\prec\alpha[\beta\in X]\}$, we have $\Phi\uparrow=Wf(W,\prec)$.

Define, for $X\subset HB$: $\Omega(X):=\{\alpha|wf(\alpha)\in X\}$.

The claim now amounts to: $\Omega(S^c\uparrow)=Wf(W,\prec)$ $(=\Phi\uparrow)$.

The "easy" part of this is the inclusion $\supset$.

$\alpha\in\Omega(S^c\uparrow)$

$\Leftrightarrow$ $wf(\alpha)\in S^c\uparrow$ (by def. of Ω)

$\Leftrightarrow$ $wf(\alpha)\notin S\downarrow$ (by 5.2)

$\Leftrightarrow$ $\neg\exists\beta[\,pr(\beta,\alpha),wf(\beta)\in S\downarrow]\wedge nw(\alpha)\notin S\downarrow$
(by the rules wf, since $S\downarrow$ is a fixed point)

$\Leftrightarrow$ $\neg\exists\beta[pr(\beta,\alpha)\in T\downarrow\wedge wf(\beta)\in S\downarrow]\wedge nw(\alpha)\notin T\downarrow$
(on the P-language, Q and P behave similarly)

$\Leftrightarrow$ $\neg\exists\beta[\beta\prec\alpha\wedge wf(\beta)\in S\downarrow]\wedge\alpha\in W$ (by !* and !!*)

$\Leftrightarrow$ $\forall\beta[\beta\prec\alpha\rightarrow wf(\beta)\notin S\downarrow]\wedge\alpha\in W$ (by predicate logic)

$\Leftrightarrow$ $\forall\beta[\beta\prec\alpha\rightarrow wf(\beta)\in S^c\uparrow]\wedge\alpha\in W$ (by 5.3)

$\Leftrightarrow$ $\forall\beta[\beta\prec\alpha\rightarrow\beta\in\Omega(S^c\uparrow)]\wedge\alpha\in W$ (by def. of Ω)

$\Leftrightarrow$ $\alpha\in\Phi(\Omega(S^c\uparrow))$. (by def. of Φ)

So, $\Omega(S^c\uparrow)$ is a fixed-point of Φ, whence it follows, that $Wf(W,\prec)=\Phi\uparrow\subset\Omega(S^c\uparrow)$.

Finally, we have to show that $\Omega(S^c\uparrow)\subset Wf(W,\prec)$.

Here, we cannot come by without delving into the details of the

S-hierarchies.
So instead, we show that, for all ξ, $\Omega(S^c\uparrow\xi)\subset Wf(W,\prec)$: and this clearly suffices.
Use induction w.r.t. ξ. For the parts $\xi=0$ and ξ a limit, the induction presents no problems. As to the successor-step, note the following, somewhat longer list, the individual steps of which *still* are simple and accounted for:

$\alpha\in\Omega(S^c\uparrow\xi+1)$
$\iff$ $wf(\alpha)\in S^c\uparrow\xi+1$ (by def. of Ω)
$\iff$ $wf(\alpha)\in S^c(S^c\uparrow\xi)$ (def. of the $S^c\uparrow$-hierarchy)
$\iff$ $wf(\alpha)\notin S(HU\setminus S^c\uparrow\xi)$ (def. of S^c)
$\iff$ $\neg\exists\beta[pr(\beta,\alpha), wf(\beta)\in HU\setminus S^c\uparrow\xi]\wedge nw(\alpha)\notin HU\setminus S^c\uparrow\xi$ (by the rules wf1/2)
$\iff$ $\neg\exists\beta[pr(\beta,\alpha), wf(\beta)\in S\downarrow\xi]\wedge nw(\alpha)\notin S\downarrow\xi$ (by 5.3)
$\iff$ $\neg\exists\beta[pr(\beta,\alpha)\in T\downarrow\xi, wf(\beta)\in S\downarrow\xi]\wedge nw(\alpha)\notin T\downarrow\xi$ (since pr and nw belong to the P-language)
$\iff$ $\forall\beta[pr(\beta,\alpha)\notin T\downarrow\xi$ or $wf(\beta)\notin S\downarrow\xi]\wedge nw(\alpha)\notin T\downarrow\xi$ (by logic)
$\Rightarrow$ $\forall\beta[\neg\beta\prec\alpha$ or $wf(\beta)\notin S\downarrow\xi]\wedge\alpha\in W$

(by assumptions !* and !!** we have for any ξ (since $T\downarrow\subset T\downarrow\xi$)
$pr(\beta,\alpha)\notin T\downarrow\xi\Rightarrow pr(\beta,\alpha)\notin T\downarrow\iff\neg\beta\prec\alpha$,
and similarly for nw and W - note we *don't* get $\iff$ here!)
$\iff$ $\forall\beta[\beta\prec\alpha\rightarrow wf(\beta)\in S^c\uparrow\xi]\wedge\alpha\in W$ (by 5.2)
$\iff$ $\forall\beta[\beta\prec\alpha\rightarrow\beta\in\Omega(S^c\uparrow\xi)]\wedge\alpha\in W$ (def. of Ω)
$\iff$ $\alpha\in\Phi(\Omega(S^c\uparrow\xi))$. (def. of Φ)

Due to the one $\Rightarrow$ here, we have obtained no more than
$\Omega(S^c\uparrow\xi+1)\subset\Phi(\Omega(S^c\uparrow\xi))$.
Nevertheless, this suffices. For, assuming $\Omega(S^c\uparrow\xi)\subset Wf(W,\prec)=\Phi\uparrow$ by way of inductive hypothesis, we then have
$\Omega(S^c\uparrow\xi+1)\subset\Phi(\Omega(S^c\uparrow\xi))\subset\Phi(\Phi\uparrow)=\Phi\uparrow=Wf(W,\prec)$, and we have completed our proof. ⊠

Let P be any program. Note that $T\downarrow\xi$ is hyperarithmetical as long as $\xi<\omega_1^{CK}$ and $T\downarrow$ is hyperarithmetical iff its downward closure ordinal is $<\omega_1^{CK}$; cf., e.g., [Shoenfield 1967].
Here, ω_1^{CK} is the least ordinal which is not the height of a recursive well-founded relation.
Below we show that every Π^1_1-set A can be co-computed. Now, take any non-hyperarithmetical $A\in\Pi^1_1$ and let P co-compute A. Clearly then, T must have downward closure ordinal ω_1^{CK}, since A is computable in the complement of $T\downarrow$.
This proves the $\alpha=\omega_1^{CK}$ - instance of the following

Theorem ([Blair 1982]).

Every ordinal $\leqslant\omega_1^{CK}$ is the downward closure ordinal of an immediate consequence operator.

Proof. Every $\alpha<\omega_1^{CK}$ is the height of a recursive well-founded structure $(W,\prec)$. The program co-computing $Wf(W,\prec)$ $(=W$, since $(W,\prec)$ is well-founded) described in the previous theorem (since W and $\prec$ are recursive, we can construct a base-program co-computing W and the complement of $\prec$) has this ordinal as its downward closure ordinal. ⊠
Similar programs for the $<\omega_1^{CK}$-part of the theorem were used in Apt [1988].

Theorem. If $W\in\Pi^1_1$ and $\prec\in\Sigma^1_1$ then $Wf(W,\prec)$ is Π^1_1, hence co-computable, and has ordinal $\leqslant\omega_1^{CK}$.

Proof. Immediate from the theorem and the fact (to be shown below) that Π^1_1-sets are exactly the co-computable ones. ⊠

5.6. In order to finally prove Blair's result, it clearly suffices to show that we can *co*-compute an arbitrary Π^1_1-set.
For this, we use the Kleene-Spector normal-form for Π^1_1-sets. It says that, when $A\in\Pi^1_1$, there is a recursive $R\subset\mathbb{N}^{<\omega}\times\mathbb{N}$ ($\mathbb{N}^{<\omega}$ is the set of codes of number-sequences of finite length - of course, a *nice* coding makes it *equal* to $\mathbb{N}$) such that, for $\prec$ defined by: $\beta\prec\alpha:\equiv\beta$ properly extends α:

5.6.1 $\quad n\in A \iff \prec$ is well-founded on $W_n:=\{\alpha|R(\alpha,n)\}$.

(Cf., for instance, Shoenfield [1967], p. 180, the "Tree Theorem" and the first six lines of the proof given there.)
Now, we first parametrize the fore-going.
For each n, define the (recursive) monotone operator $\Phi_n:\mathcal{P}(W_n)\to\mathcal{P}(W_n)$ by

5.6.2 $\quad \Phi_n(X):=\{\alpha\in W_n|\forall\beta\prec\alpha[\beta\in W_n\to\beta\in X]\}$.

Again, $\Phi_n\uparrow$ will be the well-founded part of $(W_n,\prec)$. Therefore,

5.6.3 $\quad \prec$ is well-founded on $W_n \iff W_n\subset\Phi_n\uparrow$.

So, combining things, we have

5.6.4 $\quad n\in A \iff W_n\subset\Phi_n\uparrow$.

Relation with 5.3: note that 5.6.4 corresponds to the A-computing "program-rule"

$$\forall\alpha[\alpha\in W_n\to\alpha\in\Phi_n\uparrow]\to\alpha\in A$$

which takes the form of a co-rule

$$\forall\alpha[\alpha\notin W_n\vee\alpha\in\Phi_n\uparrow]\to\alpha\in A$$

provided we can read $\ulcorner\alpha\notin W_n\urcorner$ *positively.* (Cf. the new assumption in the next section.) This motivates the rule [A] below.

5.7. To transform definition 5.6.4 into a co-computation of A, we first

co-compute $Wf(\prec|W_n)=\Phi_n\uparrow$ uniformly in n. By the discussion above, this is accomplished by the rules

wf 1: $pr(\beta,\alpha,n)\wedge wf(\beta,n) \rightarrow wf(\alpha,n)$

wf 2: $nw(\alpha,n) \rightarrow wf(\alpha,n)$

where we now **assume:** a base-program P co-computing W_n and the complement of $\prec$ in $nw(\alpha,n)$ resp. $pr(\beta,\alpha,n)$. Of course, this can be done since both are recursive (for instance, either by 4.9 or by Blair's 4.10).
In order to be able to co-compute A, we have to **assume furthermore** that our base-program P also co-computes the *complement* of W_n; say, in $w(\alpha,n)$. Again, this is unproblematic since the complement of a recursive relation is recursive.
These rules, together with P, form the program Q.
Let S be its associated operator.

5.8. Finally, 5.6.4 can now be transformed straightforwardly into the A-co-computing rule

[A] $w(\alpha,n)\wedge wf(\alpha,n)\rightarrow \underline{A}(n)$.

Add this to Q. Let R be the resulting program.
U is the operator associated with R.
Claim: R co-computes A in $\underline{A}$.
proof:
$\underline{A}(n)\notin U\downarrow$

$\Leftrightarrow \neg\exists\alpha[w(\alpha,n),wf(\alpha,n)\in U\downarrow]$ (by [A], since $U\downarrow$ is a fixedpoint)

$\Leftrightarrow \forall\alpha[w(\alpha,n)\notin U\downarrow$ or $wf(\alpha,n)\notin U\downarrow]$ (by predicate logic)

$\Leftrightarrow \forall\alpha[w(\alpha,n)\notin T\downarrow$ or $wf(\alpha,n)\notin S\downarrow]$
(since P,Q,R agree on atoms of the P-language such as w;
similarly: Q and R agree w.r.t. wf)

$\Leftrightarrow \forall\alpha[\alpha\notin W_n$ or $\alpha\in\Phi_n\uparrow]$
(P co-computes the *complement* of W_n in w;
Q co-computes $\Phi_n\uparrow$ in wf)

$\Leftrightarrow \forall\alpha[\alpha\in W_n\Rightarrow\alpha\in\Phi_n\uparrow]$ (by propositional logic)

$\Leftrightarrow W_n\subset\Phi_n\uparrow$

$\Leftrightarrow n\in A.$ (by 5.6.4) ⊠

This ends our proof. However, note the

5.9. Corollary. More liberal rules of the form $C\rightarrow R(x)$ where C is arbitrary positive (*any* number of ∃ and ∀ allowed!) can always be replaced by *ordinary* (i.e., quantifier-free) rules yielding the *same greatest* fixed point.

Proof. The greatest fixed point of such rules always is Σ^1_1. ⊠

6. Extending the finite-failure characterization: canonical mgu-trees.

6.1. Introduction.

Let P be a (definite) logic program and A a ground atom.
There are *two* (well-known) theorems which draw conclusions about the status of A relative to P from the form of an SLD-search tree for the goal $\leftarrow A$. They are the following two equivalences:

(i) $A \in T\uparrow \Leftrightarrow$ every SLD-tree for $(\leftarrow)A$ is succesful

("strong completeness of SLD-resolution w.r.t. ground-atoms"),

and, whenever ϱ is a *fair* selection-rule:

(ii) $A \notin T\downarrow\omega \Leftrightarrow$ the SLD-tree for A generated by ϱ fails finitely

(characterization of "finite failure").

This section generalizes (ii), showing that from an unsuccesful fair SLD-tree for A, even if it is infinite, interesting information concerning the position of A in the $T\downarrow$-hierarchy can be obtained.
We define the *canonical mgu-tree* corresponding to a (fair) SLD-tree. Our basic result then reads as follows:

(iii) $A \notin T\downarrow \Leftrightarrow$ the canonical mgu-tree corresponding to a fair SLD-tree for A is well-founded.

This result is then refined by putting in ordinals at both sides: on the left, measuring where A drops out of the $T\downarrow$-hierarchy, and on the right, measuring the height of the mgu-tree. These results contain (ii) as a special case.
By nature of its impredicative definition $T\downarrow = \bigcup\{X \subset HB | X \subset T(X)\}$, $T\downarrow$ is Σ^1_1.
Therefore, by Kleene-Spector, there is a recursive R such that

$A \notin T\downarrow \Leftrightarrow \prec$ is well-founded on $\{\alpha | R(\alpha, A)\}$.

The canonical mgu-tree is just one proposal for such an R which is closely associated with a (fair) SLD-tree for A.
Before coming to its definition, we need to say a few things on how to attain fairness, on mgu's and on standardization-apart in sections 2, 3 and 4 below.

6.2. Attaining fairness.

Notice, that (primitive) recursive fair selection rules exist. A natural one is obtained when, generating an SLD-tree, at each node a finite stack of literals to be selected is kept; at each resolution-step the, in the stack lowest, literal (or its descendant) is selected and the new literals introduced by the next step - if any - are added on top of the stack (so, the stack is used first-in first-out). With this rule - which we call the **stack-rule** - , it is clear, at each moment of the resolution-process, after how many steps a certain literal (or its descendant) will be resolved (if it can be resolved, and if the derivation does not end with failure

before).
A recursive selection rule produces a recursively enumerable SLD-tree. And if we regard the SLD-tree as a *prefix-tree* - which we shall - it will even be recursive.
Similarly, a recursive fair rule such as the above will produce a recursive canonical mgu-tree.

6.3. Mgu's.
The SLD-resolution process uses *most general unifiers*. We take these to be produced by the Montanari-Martelli (MM-) algorithm (cf. Apt [1988]). Mgu's obtained this way satisfy two well-known properties. To formulate these, we introduce some terminology first. Notice that a mgu first is a *substitution*, that is: a function θ from variables to terms such that $Dom(\theta):=\{x\in Var|\theta(x)\neq x\}$ is finite. We denote by $Ran(\theta)$ the set $\bigcup\{Var(\theta(x))|x\in Dom(\theta)\}$ of all variables occurring in terms $\theta(x)$ for $x\in Dom(\theta)$. Finally, $Var(\theta):=Dom(\theta)\cup Ran(\theta)$.
The two crucial properties of MM-mgu's now are

(i) idempotency: $Dom(\theta)\cap Ran(\theta)=\emptyset$

(ii) relevancy: if θ is constructed to most generally unify expressions A and B then $Var(\theta)\subset Var(A)\cup Var(B)$.

In order to construct a mgu θ for which $t_i\theta=s_i\theta$ $(i<k)$, the MM-algorithm transforms the set $\{t_i=s_i|i<k\}$ of equations - in case an mgu exists - into a "solved set" of equations $E(\theta)=\{x_i=u_i|i<m\}$ which is - *modulo* free equality axioms - logically equivalent with the original one and has the properties that (i)' no x_i occurs in a u_j and (ii)' every variable in $E(\theta)$ occurs in some equation $t_i=s_i$ - i.e., the transformation does not introduce new variables. Then $\theta:=\{x_i/u_i|i<m\}$ is the desired mgu.
HU is the **Herbrand universe** of closed terms.
A substitution λ is **ground** if $\lambda(x)\in HU$ for each $x\in Dom(\lambda)$.
In the following lemma, the notation $HU\models E(\theta)[\lambda]$ - where θ is an arbitrary substitution and λ is ground - signifies that λ - considered as an HU-ssignment of variables - *satisfies* the equations in $E(\theta)$ (in the usual logical sense of this term).
We could relax this by not requiring λ to be ground, but this won't be needed.
6.3.1 Lemma. Suppose that we have two substitutions θ and λ where θ is idempotent and λ is ground. Then the following conditions are pairwise equivalent:

1. $HU\models E(\theta)[\lambda]$ (λ **satisfies** $E(\theta)$)
2. $\lambda=\theta\lambda$ (λ **matches** θ)
3. for some δ, $\lambda=\theta\delta$ (λ **refines** or **specializes** θ).

Proof.

$1 \Rightarrow 2$. If $x \in \mathrm{Dom}\,\theta$, $\theta(x)=t$, then $HU \models x=t[\lambda]$, i.e., $\lambda x = t\lambda = x\theta\lambda$. If $x \notin \mathrm{Dom}\,\theta$ then $\theta(x)=x$ hence $x\lambda = x\theta\lambda$.

$2 \Rightarrow 1$. If $x \in \mathrm{Dom}\,\theta$, $\theta(x)=t$, then $\lambda(x)=t\lambda$ hence $HU \models x=t[\lambda]$.

$2 \Rightarrow 3$. Take $\delta := \lambda$.

$3 \Rightarrow 2$. $\lambda = \theta\delta = \theta\theta\delta = \theta\lambda$ by idempotency. ⊠

Remark. The lemma makes sense also if we replace HU by an arbitrary algebra C and let $\lambda: Var \to C$ be a C-assignment of the variables; the proof remains unchanged.

The same is true for the following:

6.3.2 Lemma. Suppose that θ is idempotent.

Every ground substitution λ with $\mathrm{Dom}(\lambda)=\mathrm{Ran}(\theta)$ uniquely extends to a $\lambda' \supseteq \lambda$ on $\mathrm{Var}(\theta)$ refining θ.

Proof. Define $\lambda' := \theta\lambda$.

It remains to check that $\mathrm{Dom}(\lambda')=\mathrm{Var}(\theta)$:

If $x \in \mathrm{Ran}(\theta)$ then $x \notin \mathrm{Dom}(\theta)$, hence $x\lambda' = x\theta\lambda = x\lambda$ and $x \in \mathrm{Dom}(\lambda')$.

If $x \in \mathrm{Dom}(\theta)$ then $\mathrm{Var}(x\theta) \subset \mathrm{Ran}(\theta)$, $x\lambda' = x\theta\lambda \neq x$ $x\theta \neq \theta$, and $x \in \mathrm{Dom}(\lambda')$ as well.

Hence, $\mathrm{Var}(\theta) \subset \mathrm{Dom}(\lambda') \subset \mathrm{Ran}(\theta)$, i.e., $\mathrm{Dom}(\lambda') \subset \mathrm{Var}(\theta)$.

Finally, if $x \in \mathrm{Dom}(\lambda')$ then $x \neq x\lambda'$ and so either $x \in \mathrm{Dom}(\theta)$ or $x \in \mathrm{Dom}(\lambda) \setminus \mathrm{Dom}(\theta)$.

Finally, uniqueness of λ': suppose that $\lambda'' = \theta\lambda''$, $\lambda'' \supseteq \lambda$.

Let $x \in \mathrm{Var}(\theta)$. We need to show, that $x\lambda' = x\lambda''$.

(i) $x \in \mathrm{Ran}(\theta)$. Then $x\lambda'' = x\lambda = x\lambda'$.

(ii) $x \in \mathrm{Dom}(\theta)$. Then $x\lambda'' = x\theta\lambda''$ and $x\lambda' = x\theta\lambda'$. However, $x\theta\lambda'' = x\theta\lambda'$ since λ'' and λ' agree on $\mathrm{Ran}(\theta)$ by (i). ⊠

6.4. Standardization-apart.

To effectuate standardization-apart in the application of program-rules, assume first of all that all variables of an initial goal N_0 shall have superscripts 0 attached.

Suppose that $N_0, N_1, N_2, N_3, \ldots$ is a derivation using (variants of) program-rules $P_1, P_2, P_3, \ldots$ and mgu's $\theta_1, \theta_2, \theta_3, \ldots$.

We'll always assume that all variables in the (variant of the) rule P_i used to deduce N_i via θ_i shall have the superscript i attached to them.

It then follows that variables in N_i shall always have one of the superscripts $0, \ldots, i$.

Furthermore, in the set-up so arranged we'll have, by idempotency and relevancy of the mgu's, for each sequence $\theta_1, \ldots, \theta_n$ of (in a derivation) consecutively used mgu's, the following facts:

6.4.1 Lemma. $\mathrm{Dom}(\theta_i) \cap \mathrm{Var}(\theta_{i+1}) = \emptyset$.

Proof. Suppose that $x \in \mathrm{Dom}(\theta_i)$.

Now, $N_i = \Leftarrow(C,M)\theta_i$ where, for some A,B: $P_i = (C \to B)$, $N_{i-1} = \Leftarrow(A,M)$ and

$B\theta_i = A\theta_i$.
By idempotency, $x \notin Var(N_i)$. (If $x \in Var(C,M)$ then, since $x \in Dom(\theta_i)$, θ_i pulls it out; if $x \notin Var(C,M)$ then, since $x \notin Ran(\theta_i)$, θ_i won't put it back in.)
By our standardization-apart convention, $x \notin Var(P_{i+1})$.
But, $Var(\theta_{i+1}) \subset Var(N_i) \cup Var(P_{i+1})$ (relevancy).
Hence, $x \notin Var(\theta_{i+1})$. ⊠
6.4.2 Corollary. If $i<j$ then $Dom(\theta_i) \cap Var(\theta_j) = \emptyset$ and all compositions $\theta_1 \ldots \theta_n$ of (in a derivation) consecutive mgu's $\theta_1, \ldots, \theta_n$ are idempotent.
Proof. As above. If $x \in Dom(\theta_1 \ldots \theta_n)$, there is a *first* θ_i such that $x \in Dom(\theta_i)$. Then $x \notin Ran(\theta_i)$, and, by our standardization-apart convention, no θ_j $(j>i)$ will introduce x back again. So, for $j>i$, $x \notin Var(\theta_j)$; $x \notin Ran(\theta_1 \ldots \theta_n)$, and $\theta_1 \ldots \theta_n$ will be idempotent. ⊠
6.4.3 Lemma. If for all i s.t. $1 \leqslant i \leqslant n$: $\lambda = \theta_i\lambda$, then $\lambda = \theta_1 \ldots \theta_n\lambda$.
Proof. Trivial. ⊠

6.5. Canonical mgu-trees associated with failing SLD-trees.
Fix a program P. Let A be a ground atom not in $T\uparrow$.
B is the (recursive) SLD-prefix-search-tree for A constructed by some selection rule.
To describe the canonical mgu tree $M = M(B)$ associated with B, it is useful to identify the elements of B - which really are finite, unsuccesful derivations from the initial goal A - with the sequence $\tau = (\theta_1, \ldots, \theta_k)$ of consecutive mgu's used in constructing such a derivation.
$HU^{<\omega}$ denotes the set of all substitutions which are ground.
For $\tau = (\theta_1, \ldots, \theta_k) \in B$, $Var(\tau) := \bigcup_{1 \leqslant i \leqslant k} Var(\theta_i)$.

6.5.1 Definition. If $\lambda \in HU^{<\omega}$, $\tau = (\theta_1, \ldots, \theta_k) \in B$, $Dom(\lambda) = Var(\tau)$ and for all i s.t. $1 \leqslant i \leqslant k$, $\lambda = \theta_i\lambda$, then we'll say that λ **matches** τ.

6.5.2 Definition.
$$M = M(B) := \{(\tau,\lambda) \in B \times HU^{<\omega} \mid \lambda \text{ matches } \tau\}.$$
The ordering $\prec$ on M(B) is the natural one:
$$(\sigma,\chi) \prec (\tau,\lambda) :\equiv \sigma <_B \tau \wedge \chi \supset \lambda.$$
Here, $<_B$ is the prefix-ordering of B: $\sigma <_B \tau$ means that σ properly extends τ in B.

6.5.3 Example. $P := \{px \rightarrow pSx;\ px \rightarrow A\}$ is the usual example illustrating that $T\downarrow$ may be different from $T\downarrow\omega$. In fact, $T\downarrow\omega = \{A\}$ and $T\downarrow = T\downarrow(\omega+1) = \emptyset$.
By the way, it also is a simple example showing that satisfiability of equations in *Herbrand*-models (opposed to *arbitrary* models) is incompact.
There is but one selection rule; it produces, when confronted with the initial goal A an SLD-tree B consisting of one infinite branch. The clauses

on this branch are $A, px^1, px^2, px^3, \ldots$; the (equations corresponding to the) mgu's used are $\emptyset$, $x^1 = Sx^2$, $x^2 = Sx^3$, $x^3 = Sx^4, \ldots$.
B is not well-founded.
However, note that M(B) *is* well-founded. In fact, the node A has $\prec$-height $\omega+1$.

The reader may question the non-emptiness of M(B). However:

6.5.4 Lemma. $\forall \tau \in B\ \exists \lambda \in HU^{<\omega}[(\tau,\lambda) \in M(B)]$.

Proof.
Suppose that $\tau = (\theta_1, \ldots, \theta_k) \in B$.
Define $Basis(\tau) \subset Var(\tau)$ by recursion on k as follows:

$Basis((\theta)) = Ran(\theta)$

$Basis(\tau^\frown(\theta)) = Ran(\theta) \cup [Basis(\tau) \setminus Dom(\theta)]$

($^\frown$ denoting concatenation).

Now 6.5.4 immediately follows from the

6.5.5 Claim. Each λ:Basis(τ)$\rightarrow$HU uniquely extends to a λ':Var(τ)$\rightarrow$HU matching τ.

Proof. For $\tau = (\theta)$, this is 6.3.2.
Next, suppose that λ:Basis($\tau^\frown(\theta)$)$\rightarrow$HU.
$\gamma := \lambda | Ran(\theta)$ extends to γ' matching θ by 6.3.2.
$\xi := \lambda | [Basis(\tau) \setminus Dom(\theta)] \cup \gamma' | [Basis(\tau) \cap Dom(\theta)]$ extends to ξ' matching τ by induction hypothesis.
Now define $\lambda' := \gamma' \cup \xi'$. ⊠

6.6. Relating M(B) to the T↓-hierarchy.
First, we have the following "Kleene-Spector normal-form" for HB\T↓:

6.6.1 Theorem.

1. If $\prec$ is well-founded on M(B) then $A \notin T\downarrow$.

Conversely:

2. If B is fair and $A \notin T\downarrow$ then $\prec$ is well-founded on M(B).

6.6.1.2 immediately follows from a lemma which deserves to be mentioned separately:

6.6.2 Lemma. Let β be an infinitely descending branch through M(B) - B a fair SLD-tree.
Let λ be the union of second components γ in tuples $(\sigma,\gamma) \in \beta$.
Let α be the infinite branch through B of which all σ with $(\sigma, \lambda | Var(\sigma)) \in \beta$ are initials.
Then if C is any goal on α, $T\downarrow \models C\lambda$.

Proof.
Suppose this is not the case.
Then a *least* ordinal ξ exists such that, for some C on α, and some literal

$R \in C$, $R\lambda \notin T{\downarrow}\xi$.
Now, α is infinite.
Therefore, since B was constructed using a fair rule, somewhere below C in B a descendant $R\theta$ of R must be selected, θ being the composition of mgu's used along α going from C to the place where the selection takes place. (θ may be empty if R was selected immediately.)
Since α is infinite, $R\theta$ can be unified with the head of a program-rule (for if not, α would stop here and fail).
Suppose that α uses the rule $D' \rightarrow R'$ to resolve $R\theta$ using a next mgu θ'.
Then $D'\theta'$ is part of the clause on α immediately below the occurrence of $R\theta$.
Now, $(D' \rightarrow R')\theta'\lambda = (D'\theta'\lambda \rightarrow R'\theta'\lambda) = (D'\theta'\lambda \rightarrow R\theta\,\theta'\lambda) = D'\theta'\lambda \rightarrow R\lambda$, since λ matches all compositions of consecutive mgu's along β.
As $R\lambda \notin T{\downarrow}\xi$, there must be a literal $Q\theta'\lambda$ of $D'\theta'\lambda$ and an ordinal $\delta < \xi$ such that $Q\theta'\lambda \notin T{\downarrow}\delta$ - contradicting the minimality of ξ. ⊠

Proof of 6.6.1.1.
Suppose that $\prec$ is well-founded on M(B).
We claim that for every $(\tau,\lambda) \in M(B)$:

for some literal Q in the goal of τ, $Q\lambda \notin T{\downarrow}$.

In particular, this will be true of the top-node of M(B) where A is the *only* literal, and we are through.
The claim is proved using induction along $\prec$.
So, suppose that $(\tau,\lambda) \in M(B)$, where C is the goal of τ.
Supose that, for every literal $Q \in C$, we have $Q\lambda \in T{\downarrow}$. [*]
Then in particular for the Q in C selected in B, $Q\lambda \in T{\downarrow}$.
Since $T(T{\downarrow}) = T{\downarrow}$, for some ground-instance $D \rightarrow Q\lambda$ of a rule we have $T{\downarrow} \models D$. [**]
Let $E \rightarrow R$ be the rule of which $D \rightarrow Q\lambda$ is an instance; say, $(D \rightarrow Q\lambda) = (E \rightarrow R)\gamma$ where γ is ground.
So, Q unifies with R: by standardization-apart, $\lambda' := \lambda \cup \gamma$ is a unifier.
Let θ be a mgu of Q and R.
Since Q unifies with R, one of the resolvents F of C uses the rule $E \rightarrow R$ with the mgu θ.
As $\lambda' = \theta\delta$ for some δ, λ' matches θ, and so, since $\lambda' \supseteq \lambda$ matches every mgu occurring before θ, $(\sigma,\lambda') \in M(B)$, where σ (corresponding to F) is the successor of τ in B.
By induction-hypothesis, for some $S \in F$, $S\lambda' \notin T{\downarrow}$.
If $S = S'\theta$ with $S' \in C \setminus \{Q\}$, then $S\lambda' = S'\theta\lambda' = S'\lambda' = S'\lambda$, contradicting the assumption [*] above.
So, $S = S'\theta$ with $S' \in E$. However, $S\lambda' = S'\theta\lambda' = S'\lambda' = S'\gamma \in D$, contradicting [**].
⊠

Now for a sharpened version of 6.6.1.
In order to obtain a slightly nicer match, *redefine* the T↓-hierarchy by the one recursion-equation

[1] $\quad T{\downarrow}\xi = T(\bigcap_{\delta<\xi} T{\downarrow}\delta)$.

(Cf. for instance, Moschovakis [1974] and Barwise [1975].)
The difference with the usual definition - which can succinctly be given by the recursion

[2] $\quad T{\downarrow}\xi = \bigcap_{\delta<\xi} T(T{\downarrow}\delta)$

- is that [1] only proceeds through the successor-stages of the [2]-hierarchy, *skipping* the 0-th- and all limit-stages.
Let $\tau(A)$ be the usual rank of $A\notin T{\downarrow}$ in the prewellordering on the *complement* of T↓ corresponding to either one of these hierarchies.
N.B.: the induced prewellorderings under either [1] or [2] of course are the same, and so this redefinition does not affect (the definition of) τ.
Then for $A\notin T{\downarrow}$,

$\tau(A)=\xi$ iff $A\in\bigcap_{\delta<\xi} T{\downarrow}\delta \setminus T{\downarrow}\xi$

under the new definition [1],

iff $A\in T{\downarrow}\xi \setminus T{\downarrow}(\xi+1)$

under the old one [2].
For the sequel, let $\rho(\tau,\lambda)$ be the $\prec$-rank of (τ,λ) in the well-founded part of M(B).

6.6.1.1, refined.

If $\prec$ is well-founded on M(B) and the top-node $(A,\emptyset)$ has $\rho(A,\emptyset)=\mu$, then $A\notin T{\downarrow}\mu$.
I.e., $\tau(A)\leqslant\rho(A,\emptyset)$.

Proof. Almost as before, now with bounds added.
Assume the hypothesis.
We claim that for every $(\tau,\lambda)\in M(B)$ with $\prec$-rank ξ
for some literal Q in the goal of τ, $Q\lambda\notin T{\downarrow}\xi$.
In particular, this will be true of the top-node of M(B) where A is the *only* literal and $\xi:=\mu$, and we are through.
The claim is proved using induction along $\prec$.
So, suppose that $(\tau,\lambda)\in M(B)$, where C is the goal of τ, has $\prec$-rank ξ.
Supose that, for every literal $Q\in C$, we have $Q\lambda\in T{\downarrow}\xi$. [*]
Then in particular for the Q in C selected in B, $Q\lambda\in T{\downarrow}\xi$.
Since $T{\downarrow}\xi=T(\bigcap_{\delta<\xi} T{\downarrow}\delta)$, for some ground-instance $D\rightarrow Q\lambda$ of a rule we have $\bigcap_{\delta<\xi} T{\downarrow}\delta\models D$. [**]
Let $E\rightarrow R$ be the rule of which $D\rightarrow Q\lambda$ is an instance; say, $(D\rightarrow Q\lambda)=(E\rightarrow R)\gamma$ where γ is ground.
So, Q unifies with R: by standardization-apart, $\lambda':=\lambda\cup\gamma$ is a unifier.
Let θ be a mgu of Q and R.

Since Q unifies with R, one of the resolvents F of C uses the rule $E \to R$ with the mgu θ.
As $\lambda' = \theta\delta$ for some δ, λ' matches θ, and so, since $\lambda' \supseteq \lambda$ matches every mgu occurring before θ, $(\sigma,\lambda') \in M(B)$, where σ (corresponding to F) is the successor of τ in B.
Let $\delta < \xi$ be the $\prec$-rank of (σ,λ').
By induction-hypothesis, for some $S \in F$, $S\lambda' \notin T{\downarrow}\delta$.
If $S = S'\theta$ with $S' \in C\setminus\{Q\}$, then $S\lambda' = S'\theta\lambda' = S'\lambda' = S'\lambda$, contradicting the assumption [*] above, as $T{\downarrow}\xi \subset T{\downarrow}\delta$.
So, $S = S'\theta$ with $S' \in E$. However, $S\lambda' = S'\theta\lambda' = S'\lambda' = S'\gamma \in D$, contradicting [**].
⊠

Due to the fact that the SLD-resolution process handles but one atom at the time while many others must remain waiting for resolution - if we use the stack-rule to attain fairness, these make up the stack employed - , the next result probably cannot be made much sharper.

6.6.1.2, refined.

Suppose that fairness of B is accomplished by the stack-rule.
If $A \notin T{\downarrow}\mu$ then $\prec$ is well-founded on M(B) and $\varrho(A,\emptyset) \leqslant \omega \times \mu$, i.e., $\varrho(A,\emptyset) \leqslant \omega \times \tau(A)$.
Moreover, the sharper bound $\varrho(A,\emptyset) \leqslant \alpha + n(s+1)$ can be obtained - where α is the limit ordinal and n the natural number such that $\mu = \alpha + n$ - if the stack employed never (i.e., nowhere in B) will contain more than s literals.

Proof.
Well-foundedness of $\prec$ follows from 6.6.1.2.
Now, since B *must* select the only literal A of its top-goal A (i.e., the stack going with A in B must be empty), the first part of this result is a special case of the following lemma, where k=0:

6.6.3 Lemma. If 1. $(C,\lambda) \in M(B)$, $T{\downarrow}\xi \not\models C\lambda$
2. the stack going with C in B contains k literals
then $\varrho(C,\lambda) \leqslant \omega \times \xi + k$.

Proof. Induction w.r.t. ξ.
By 1., choose $Q \in C$ such that $Q\lambda \notin T{\downarrow}\xi$.
Suppose that $(D,\gamma) \prec (C,\lambda)$ and the descendant of Q is selected in D.
By 2., $\varrho(D,\gamma) + k' = \varrho(C,\lambda)$ for some $k' \leqslant k$.
Let (D',γ') be an immediate predecessor of (D,γ) in M(B).
Then in D', the descendant of Q in D has been resolved,
whence, by 1., $\bigcap_{\delta<\xi} T{\downarrow}\delta \not\models D'\gamma'$.
So, for some $\delta < \xi$, $T{\downarrow}\delta \not\models D'\gamma'$.
By induction hypothesis, $\varrho(D',\gamma') \leqslant \omega \times \delta + p(D')$ - p(D') being the size of the

stack going with D' in B.
Hence, $\varrho(D',\gamma') < \omega\times\delta+\omega \leqslant \omega\times\xi$.
Therefore, $\varrho(D,\gamma) = \sup\{\varrho(D',\gamma')+1 \mid (D',\gamma') \prec (D,\gamma)\} \leqslant \omega\times\xi$.
And so, $\varrho(C,\lambda) \leqslant \omega\times\xi+k' \leqslant \omega\times\xi+k$. ⊠

The second part of the refined version of 6.6.1.2 similarly follows from the

6.6.4 Lemma. If 1. $(C,\lambda)\in M(B)$, $T{\downarrow}\xi \not\models C\lambda$
2. the stack never gets size $>s$ in B
then $\varrho(C,\lambda) \leqslant \alpha+n(s+1)$ - where α is the limit ordinal and n the natural number such that $\xi=\alpha+n$.

Proof. Similar to 6.6.3. ⊠

6.6.5 Corollary. If B is constructed fairly using the stack-rule and $\omega\times\tau(A)=\tau(A)$, then $\varrho(A,\emptyset)=\tau(A)$.

6.6.6 Fact. $\omega\times\mu=\mu$ iff $\exists\alpha[\mu=\omega^{\omega}\times\alpha]$.

6.6.7 Theorem.

1. If ϱ is a selection-rule such that, for $A\in HB\backslash T{\uparrow}$, $(A,\emptyset)$ has rank $<\alpha$ in the associated mgu-tree then P is α-weakly recurrent.
2. In particular, if ϱ is such that for each $A\in HB\backslash T{\uparrow}$ the SLD-search-tree for the goal $A\rightarrow$ is finite then P is weakly recurrent.

Proof.
1. By 6.6.1.1, if $(A,\emptyset)$ has rank $<\alpha$ ($A\notin T{\uparrow}$) then A has rank $<\alpha$ in the $T{\downarrow}$-hierarchy. Hence, $HB\backslash T{\uparrow} \subset HB\backslash T{\downarrow}\alpha$, i.e., $T{\downarrow}\alpha \subset T{\uparrow}$.
2. If SLD-trees are finite then the associated mgu-trees are finite as well and the result follows from 1. by taking $\alpha=\omega$. ⊠

7. Non-standard Herbrand-universes.

7.0. Contents.

Sections 1-3 of this part discuss quotients and limits of free term algebras, convergence of sequences of substitutions, the structure of limits and several ways to turn limits into models.

Finally, in section 4, we have a couple of remarks in the margin of [Blair/Brown 199?]. This paper constructs, by iterating in a sense the model-construction in the proof of the theorem on "completeness of negation as failure", a remarkable countable algebra over which "every program is canonical". We show that the conclusions of that paper follow if we take the algebra to be *countable recursively saturated*.

Subsequently, we relax recursive saturation to properties more appropriate for the algebraic setting.

7.1 Quotients and Limits.

Substitutions can be used in several ways to produce new algebras out of the free term algebras TERM=TERM(VAR) of all terms over a set VAR of variables:

1. As *identifications*. Used as such, they induce a quotient of the free term algebra in which a term is identified with the variable for which it is substituted. See 7.1.4.
2. As *homomorphisms* between free term algebras. The induced quotient now identifies terms being unified. (7.1.5)
3. Finally, a sequence of substitutions-as-homomorphisms can be used to form a direct limit. (7.1.6; in 7.1.7 we prove that every model of the free equality axioms can be represented in this fashion.)

The proper settings for the subject at hand are 2 and 3. Nevertheless, 1 is not without interest: first, this offers a way to produce algebras in which not every free equality axiom is preserved; second, we'll note that 1 and 2 produce isomorphs when the substitution is idempotent; and hence we then harvest the merits of both approaches.

The following discussion is parametrized by a finite set FUNC of function symbols which contains at least one individual constant (that is, a zero-argument function symbol). We start with the preliminary 7.1.1-3.

7.1.1 Equality axioms.

The **Free Equality Axioms** consist of the following:

FEQ1. $f\underline{x}=f\underline{y} \rightarrow \underline{x}=\underline{y}$ $(f\in FUNC)$

FEQ2. $f\underline{x}\neq g\underline{y}$ $(f,g\in FUNC, f\neq g)$

FEQ3. $x\neq t$ for every term t which has x as a proper subterm.

By FEQ we denote the theory axiomatized by these axioms.

The **Domain Closure Axiom** is

DCA. $\forall x \bigvee_{f \in FUNC} \exists \underline{y}: x = f\underline{y}$.

7.1.2 Congruences.

Let **C** be a FUNC-model, that is: an algebra interpreting the symbols of FUNC.

An equivalence relation $\sim$ on **C** is a **congruence** if it respects the functions of **C**: $a_1 \sim b_1, \ldots, a_n \sim b_n \Rightarrow f(a_1, \ldots, b_n) \sim f(b_1, \ldots, b_n)$.

Clearly, for every relation R on **C** there is a *least* congruence extending R.

Every congruence $\sim$ on **C** induces a quotient-algebra $\mathbf{C}/\sim$ over the set of $\sim$-equivalence classes $|c|$ ($c \in \mathbf{C}$). The quotient-map $\varepsilon: c \mapsto |c|$ then is a (surjective) homomorphism: $\mathbf{C} \to \mathbf{C}/\sim$; hence

Lemma. $\sim$ is a congruence over **C** iff there exist an algebra **B** and a homomorphism $h: \mathbf{C} \to \mathbf{B}$ such that $\sim$ is the *kernel* $\{(a,b) \in \mathbf{C}^2 | ha = hb\}$ of h.

7.1.3 Equation-induced congruences over the free algebra.

Let VAR be a set of variables and TERM=TERM(VAR) the free algebra of all FUNC-terms over VAR.

Each set E of equations $t = s$ with $t, s \in$ TERM corresponds in a 1-1-fashion to a relation $R(E) := \{(t,s) | t = s \in E\}$ over TERM.

C(E) is the least congruence containing R(E).

TERM/E is the quotient of TERM modulo C(E).

Note that, for $t \in$ TERM, the *value* $t\varepsilon$ of t under the quotient-map $\varepsilon: c \mapsto |c|$ in TERM/E (in the model-theoretic sense, where ε is considered as an assignment) equals $|t|$. (Induction on t.)

Lemma. ε satisfies every $e \in E$ in TERM/E: $\text{TERM/E} \models E[\varepsilon]$.

Representation of arbitrary algebras using equation-induced quotients:

If **C** is a (countable) algebra then $\mathbf{C} \cong \text{TERM/E}$ for some set E of equations.

Proof. Fix a surjection $\alpha: \text{VAR} \to \mathbf{C}$ and let E be the set of all equations satisfied by α in **C**. ⊠

7.1.4 Substitutions as Identifications.

Suppose that $\theta: \text{VAR} \to \text{TERM} = \text{TERM(VAR)}$ is a substitution, i.e., $\text{Dom}(\theta) := \{x \in \text{VAR} | x \neq \theta(x)\}$ is finite.

$E(\theta) := \{x = \theta(x) | x \in \text{Dom}(\theta)\}$ is the set of equations **associated** with θ.

Lemma. For each algebra **C** and $\alpha: \text{VAR} \to \mathbf{C}$, the following are equivalent:

1. $\mathbf{C} \models E(\theta)[\alpha]$
2. $\theta\alpha = \alpha$ (α **matches** θ, θ is **invariant over** α).

N.B.: $\theta\alpha$ associates with x the value $(\theta(x))\alpha$ of the term $\theta(x)$ under the assignment α in **C**.

$\text{Id}(\theta)$, the **identification** induced by θ, is the least congruence

containing the graph $\{(x,\theta(x))|x\in Dom(\theta)\}$ of θ.
Hence, $Id(\theta)=C(E(\theta))$, and $TERM/Id(\theta)=TERM/E(\theta)$.

Lemma. 1. $TERM/Id(\theta)\models E(\theta)[\varepsilon]$
2. $\theta\varepsilon=\varepsilon$
3. $TERM/Id(\theta)\models FEQ1/2$.

In 7.1.5 we note that more holds when θ is idempotent.

Example. If $\theta=\{x/Sx\}$, $|x|$ is a solution in $TERM/Id(\theta)$ of $x=Sx$, and so FEQ3 cannot be satisfied in this quotient.

7.1.5 Substitutions as unifiers.

Let θ be a substitution.
Identify θ with the unique homomorphism between the free term algebras to which it recursively extends.
$Un(\theta)$, the **unification** corresponding to θ, is the kernel of θ considered-as-a-homomorphism, i.e., $Un(\theta)$ is defined by the condition: $t\theta=s\theta$.
$TERM/\theta$ is the induced quotient.

Lemma. 1. $TERM/\theta\models FEQ$
2. $Un(\theta)\subset Id(\theta)$
3. If θ is *idempotent* then $Un(\theta)=Id(\theta)$.

Proof.
2. If $t\theta=s\theta$ then $t\sim t\theta\sim s\theta\sim s$ under $Id(\theta)$.
3. In order that $Id(\theta)\subset Un(\theta)$, since $Id(\theta)$ is the least congruence $\supset graph(\theta)$, it suffices that $graph(\theta)\subset Un(\theta)$. So, assume $\theta(x)=t$. By idempotency $x\theta=x\theta\theta=t\theta$, and hence $(x,t)\in Un(\theta)$. ⊠

N.B.:

Lemma. For every substitution θ there is an *idempotent* substitution Φ such that $\forall s,t[s\theta=t\theta\Leftrightarrow s\Phi=t\Phi]$, i.e., $Un(\theta)=Un(\Phi)$, and $TERM/\theta\cong TERM/\Phi$.

Proof. Let $V=Var(\theta)$ be the set of all variables occurring in $Dom(\theta)$ and in terms $\theta(x)$ for $x\in Dom(\theta)$. $E:=\{s=t|s,t\in TERM(V), s\theta=t\theta\}$. E is infinite. However, modulo FEQ^-, E amounts to $E^-:=\{s=t\in E|s\in V\}$.
Claim: E^- is finite.
For, otherwise $x\in V$ would exist such that $x=t\in E$ for infinitely many t. For all those t, $x\theta=t\theta$. However, $x\theta$ has a fixed complexity (=number of function symbols) r, the complexity of $t\theta$ is $\geqslant$ that of t; and there are only finitely many terms t over the finite set V of bounded complexity.
Now, feed E^- to the Martelli-Montanari-algorithm. This produces an idempotent Φ such that $\forall s,t\in TERM(V)[s\theta=t\theta\Leftrightarrow s\Phi=t\Phi]$ and hence $\forall s,t[s\theta=t\theta\Leftrightarrow s\Phi=t\Phi]$ as well. ⊠

7.1.6 Direct limits.

Suppose that V(i) is a set of variables ($i\in\mathbb{N}$).
TM(i) is the set of all FUNC-terms over V(i).

Suppose, furthermore, that $\boldsymbol{\theta}=\langle\theta_{i,i+1}|i\in\mathbb{N}\rangle$ is a sequence of substitutions $\theta_{i,i+1}:V(i)\to TM(i+1)$.
Again, identify $\theta_{i,i+1}$ with the unique free-algebra homomorphism:$TM(i)\to TM(i+1)$ to which it recursively extends.
Finally, put $\theta_{i,j}:=\theta_{i,i+1}\dots\theta_{j-1,j}$ $(i\leqslant j)$.

Definition. The **direct limit** $\mathrm{Lim}(\boldsymbol{\theta})$ is the algebra obtained as follows.
First, define $\approx$ on the set $\{(t,i)|t\in TM(i)\}$ by

[$\approx$] $\quad (t,i)\approx(s,j) :\equiv \exists k\geqslant i,j:\ t\theta_{i,k}=s\theta_{j,k}$.

The universe of $\mathrm{Lim}(\boldsymbol{\theta})$ is the quotient $\{(t,i)|t\in TM(i)\}/\approx$.
(The idea here being that (t,i) heads the sequence

$$(t,i),\ (t\theta_{i,i+1},i+1),\ (t\theta_{i,i+2},i+2),\dots;$$

(t,i) and (s,j) are identified iff their sequences eventually coincide.)
Each $F\in FUNC$ is interpreted as a function F^* over this universe defined by

$$F^*(|(t_1,i(1))|,\dots,|(t_n,i(n))|):=|(F(t_1\theta_{i(1),k},\dots,t_n\theta_{i(n),k}),k)|,$$

where $k=\max\{i(1),\dots,i(n)\}$.

Consider the following condition:

[*] $\quad t\in TM(i)$ and $j<i$ imply $t\theta_{0,j}=t$.

Lemma. [*] will be satisfied always when we obtain $\boldsymbol{\theta}$ as a sequence of mgu's along an infinite SLD-derivation and $V(i)$ is the set of variables in the i-th goal of this derivation.

For, if $\boldsymbol{\theta}$ is constructed in such a way, by idempotency and standardization-apart we will have for all $j>0$ that $\mathrm{Dom}(\theta_{0,j})\cap V(j)=\emptyset$
(if $x\in\mathrm{Dom}(\theta_{0,j})$, there is a *first* $j'<j$ s.t. $x\in\mathrm{Dom}(\theta_{j',j'+1})$, whence $x\notin V(j)$ for all $j\geqslant j'+1$)
and so $\theta_{0,j}$ does not act on $t\in TM(i)$ when $i>j$. ⊠

Lemma. If [*] holds then $\mathrm{Lim}(\boldsymbol{\theta})$ is isomorphic to the quotient $TERM/\boldsymbol{\theta}$ of $TERM:=\bigcup_i TM(i)$ modulo the congruence $\sim$ defined by:

$$t\sim s:\equiv \exists k:\ t\theta_{0,k}=s\theta_{0,k}.$$

Proof. Note first that

$$\exists k\geqslant i,j:\ t\theta_{i,k}=s\theta_{j,k} \quad \text{iff} \quad \exists k:\ t\theta_{0,k}=s\theta_{0,k}.$$

Also, by [*] the $\theta_{0,k}$ are idempotent. It then follows that the map $h:|(t,i)|\mapsto|t|$ (where $\approx$-equivalence classes are meant on the left and $\sim$-classes on the right), is an isomorphism, as for $k=\max(i,j)$, F^* resp. F° corresponding to the (say, binary) symbol F in $\mathrm{Lim}(\boldsymbol{\theta})$ resp. $TERM/\boldsymbol{\theta}$:
$h(F^*(|(t,i)|,|(s,j)|))=h|(F(t\theta_{0,k},s_{0,k}),k)|=|F(t\theta_{0,k},s\theta_{0,k})|=F^\circ(|t\theta_{0,k}|,|s\theta_{0,k}|)=F^\circ(|t|,|s|)=F^\circ(h|(t,i)|,h|(s,j)|)$, since by idempotency $t\theta_{0,k}=t\theta_{0,k}\theta_{0,k}$ and, hence, $|t|=|t\theta_{0,k}|$. ⊠

7.1.7 Representation of algebras by limits.

Definition. For substitutions λ and θ we say that

1. λ **refines** θ, notation: $\lambda \leqslant \theta$, if $\exists \delta: \lambda = \theta\delta$.
2. λ **neatly refines** θ, notation: $\lambda \leqslant^* \theta$,
   if $\lambda \leqslant \theta$ and $Dom(\lambda) \supseteq Dom(\theta)$.

Note that, if $\lambda \leqslant \theta$, then $Dom(\theta) \subset Dom(\lambda)$ iff there are no two different variables $x \in Dom(\theta)$ and $y \in Dom(\lambda)$ such that $\theta(x)=y$ and $\lambda(y)=x$.
If $\lambda \leqslant \theta$, i.e., $\lambda = \theta\delta$ for some δ, and θ is idempotent,
then $\lambda = \theta\delta = \theta\theta\delta = \theta\lambda$.

Lemma. Suppose that

1. **C** satisfies FEQ,
2. $\alpha: VAR \to \mathbf{C}$ is an assignment,
3. t,s are terms for which $t\alpha = s\alpha$,
4. θ is an idempotent substitution for which $\theta\alpha = \alpha$.

Then an idempotent λ exists, neatly refining θ,
and such that $\lambda\alpha = \alpha$ and $t\lambda = s\lambda$.

Proof.
Consider the set of equations $E := E(\theta) \cup \{t\theta = s\theta\}$.
By 3/4, α satisfies E in **C**.
Consider the Martelli-Montanari-algorithm for producing idempotent most general unifiers. Put it to work on E. Since α satisfies E in **C** and **C** models FEQ1, it is easily checked that each set of equations produced by the algorithm starting from E is satisfied by α in **C** as well. Since also $\mathbf{C} \models$ FEQ2/3, the algorithm cannot fail to produce a most general unifier λ for E for which, hence, $\lambda\alpha = \alpha$. Since λ unifies $E(\theta)$ and θ most generally unifies $E(\theta)$, we obtain $\lambda \leqslant \theta$ and, as θ is idempotent, $\lambda = \theta\lambda$. Also, $t\lambda = t\theta\lambda = s\theta\lambda = s\lambda$. Finally, $Dom(\lambda) \supseteq Dom(\theta)$: if $x \in Dom(\theta)$ then $x \notin Var(t\theta = s\theta)$. So, the algorithm never replaces a variable x on the left-hand side of an equation $x = u$ in $E(\theta)$ (or its transforms), and neither does it delete such an equation (or a transform). ⊠

Theorem. If **C** is a countable algebra satisfying the free equality axioms, then $\mathbf{C} \cong TERM/\boldsymbol{\theta}$ for some neatly descending sequence $\boldsymbol{\theta}$: $\theta_0 \geqslant^* \theta_1 \geqslant^* \theta_2 \geqslant^* \ldots$ of idempotent substitutions θ_i.

Proof.
Fix a surjection $\alpha: VAR \to \mathbf{C}$.
Fix an enumeration of all equations $s = t$ satisfied by α in **C**.
By the lemma, construct a sequence $\boldsymbol{\theta}$: $\theta_0 \geqslant^* \theta_1 \geqslant^* \theta_2 \geqslant^* \ldots$ of idempotent substitutions matched by α, the n-th substitution unifying the n-th equation.
Clearly, $s\alpha = t\alpha$ iff $\exists i: s\theta_i = t\theta_i$ (if $s\theta_i = t\theta_i$ then $s\alpha = s\theta_i\alpha = t\theta_i\alpha = t\alpha$).
Hence, the map $|t| \mapsto t\alpha$ is an isomorphism: $TERM/\boldsymbol{\theta} \to \mathbf{C}$. ⊠

7.2 Convergence.

Contents: we characterize the standard part of a direct limit produced by an infinite SLD-derivation (under suitable conditions on standardization-apart and the mgu-producing algorithm) in terms of convergence of the substitutions involved.

Sections 6.3/4 on conventions regarding mgu's and standardization-apart of program-rules in SLD-derivations are still in force.
Note in particular that we effectuated standardization-apart by means of superscripting variables: 0 to superscript variables of the initial goal, and $j>0$ to superscript variables in the (variant of the) program-rule used at the j-th step.
For the rest of this section, fix a sequence $\theta=(\theta_1,\theta_2,\theta_3,\ldots)$ of mgu's consecutively occurring in an infinite SLD-derivation.

7.2.1 Definition.

1. θ **converges on** the term t **at** i iff $\forall j\geqslant i\ [t\theta_1\ldots\theta_j=t\theta_1\ldots\theta_i]$.
2. θ **converges on** t **at** i **exactly** iff it converges at i but not at some $j<i$.
3. θ **converges on** t iff it converges on t at some i.
4. θ **converges** iff it converges on every term.

Clearly θ converges on t iff it converges on every $x\in Var(t)$; and θ converges iff it converges on every variable.

HU is the Herbrand universe of all closed terms.
An assignment $\lambda:Var(\theta)\to HU$ ($Var(\theta):=\bigcup\{Var(\theta_i)|i>0\}$) is said to **match** θ if $\theta_1\ldots\theta_i\lambda=\lambda$ for all i.
7.2.2 Lemma. If θ has a matching assigment λ in HU, then θ converges.
Proof. Suppose that λ matches θ. Fix $x\in Var$. We have to show that θ converges on x.
Define $V(i):=Var(x\theta_1\ldots\theta_i)$ and $V:=\bigcup_i V(i)$.
Claim. V is finite.
Proof. λ matches θ and has a finite domain.
Fix some k for which $V=\bigcup_{i\leqslant k}V(i)$.
Now it is easy to see that θ must converge on x (somewhere): subsequent mgu's θ_{k+1}, θ_{k+2},... can only take away variables: every variable which will be introduced eventually has been introduced by the previous mgu's already by choice of k; and, once substituted away, a variable won't come back anymore by the conventions of 6.4. Since there are only $|V(k)|$ variables in $x\theta_1\ldots\theta_i$, subsequent mgu's can only change $x\theta_1\ldots\theta_i$ and its descendants $x\theta_1\ldots\theta_j$ ($j\geqslant k$) $|V(k)|$-many times at the most, after which convergence obtains. ⊠

Conversely:

7.2.3 Lemma. Suppose that θ converges.
Define k(x) (x∈Var) by the stipulation that θ converges on x at k(x) exactly.
Define $Ran(\theta) := \bigcup\{Var(x\theta_1 \dots \theta_{k(x)}) | x \in Var\}$.
Then each $\lambda: Ran(\theta) \to HU$ uniquely extends to a $\lambda': Var(\theta) \to HU$ matching θ.

7.2.4 Corollary. θ converges iff it has a matching assignment in HU.

7.3 Structure of limits and models.

Contents. We make some observations concerning the structure of direct limits/quotients. We discuss three ways of extending direct limits to models. Finally, the theorem on completeness of negation as failure is slightly extended.

Assume that V(i), TM(i), $\theta = (\theta_{0,1}, \theta_{1,2}, \theta_{2,3} \dots)$ are as in 7.1.6 and that [*] of that section holds; so we may consider $Lim(\theta)$ as well to be the quotient $TERM/\theta$ of $TERM := \bigcup_i TM(i)$.

7.3.1 Lemma. $|.| : HU \to Lim(\theta)$ embeds the Herbrand-algebra into $Lim(\theta)$.

Proof. $\approx$ reduces to the identity on HU. ⊠

Identify HU with its image in $Lim(\theta)$ under |.|.
We need a slightly modified notion of convergence.

7.3.2 Definitions.

1. θ **converges** on t at k iff $\forall j \geqslant k$: $t\theta_{0,j}$ is a variant of $t\theta_{0,k}$.
2. $Con(\theta) := \{t \in TM | \theta \text{ converges on } t\}$
3. For $t \in Con(\theta)$, $k(t) \in \mathbb{N}$ is the number at which θ converges on t exactly
4. $St(\theta) := \{t\theta_{0,k(t)} | t \in Con(\theta)\}$
5. $ST(\theta) := \{|t| | t \in Con(\theta)\}$ is called the **standard part** of $Lim(\theta)$.

7.3.3 Lemma. $|.| : St(\theta) \to ST(\theta)$ is a bijection.
Identify $St(\theta)$ with its image $ST(\theta)$ under |.|.

7.3.4 Lemma. $|.| : Con(\theta) \to ST(\theta)$ is a surjection.

The direct limit-construction opens up two possibilities standard Herbrand universes do not have:

(i) $ST(\theta) \setminus HU$ may be non-empty. These are new elements compared to the standard situation of HU; nevertheless they are also standard in the sense that, in $Lim(\theta)$, they can be identified with the object $t\theta_{0,k(t)}$

in $St(\theta)$.

(ii) More importantly, θ may not converge on some t and hence $Lim(\theta)\backslash ST(\theta)$, the **non-standard part** of $Lim(\theta)$, may be non-empty.

7.3.5 Lemma. $Lim(\theta)$ satisfies the free equality axioms.

7.3.6 Lemma. $ST(\theta)$ is freely generated in $Lim(\theta)$ from the set
$$\{|x|\,|\,x\in Con(\theta)\}.$$

7.3.7 Lemma. $Lim(\theta)$ satisfies the domain closure axiom iff $Con(\theta)=HU$.

Direct limits turned into models.

Suppose now that REL is a set of relation symbols. A **model** will be a FUNC-algebra equipped with interpretations for all symbols in REL.
Assume that $\mathbf{A}=\langle \mathbf{A}_i | i\in\mathbb{N}\rangle$ is a sequence such that $\mathbf{A}_i$ is a model over the free algebra TM(i) ($i\in\mathbb{N}$).
Assume that $\theta_{i,j}:\mathbf{A}_i\to\mathbf{A}_j$ is a homomorphism whenever $i\leqslant j$.
Since we know already that each $\theta_{i,j}$ is a homomorphism w.r.t. the algebraic structure, this only puts a restriction on the interpretation of relation symbols in the $\mathbf{A}_i$, namely, for $t,\ldots\in TM(i)$ and $R\in REL$:
$$\mathbf{A}_i\models R(t,\ldots) \Rightarrow \mathbf{A}_j\models R(t\theta_{i,j},\ldots).$$
Identifying models with subsets of their Herbrand-base, this condition can be given more succinctly as:

for atomic Q: $\quad Q\in\mathbf{A}_i \Rightarrow Q\theta_{i,j}\in\mathbf{A}_j$.

If this is satisfied, we can turn $Lim(\theta)$ into **the model** $Lim(\mathbf{A},\theta)$ **over** $Lim(\theta)$ **induced by A** by defining interpretations for the relation symbols according to the following

7.3.8 Definition. For $t,s,\ldots\in TM(i)$:
$$Lim(\mathbf{A},\theta)\models R(|t|,|s|,\ldots) :\equiv \exists j\geqslant i:\ \mathbf{A}_j\models R(t\theta_{i,j},s\theta_{i,j},\ldots).$$
Of course, this definition requires some careful handling.
If t,s,... are in diferent TM(i); say, $t\in TM(i(t))$, let $m:=\max\{i(t),i(s),\ldots\}$; now $Lim(\theta)\models R(|t|,|s|,\ldots)$ just means $\exists k\geqslant m:\ \mathbf{A}_k\models R(t\theta_{i(t),k},s\theta_{i(s),k},\ldots)$.

For each i, let T(i) be the monotone operator over the Herbrand-base HB(i) corresponding to TM(i) which is associated with the program P.
Again, $T(i)\uparrow$ and $T(i)\downarrow$ are its least resp. greatest fixed point: they both are identified with models over TM(i).

For these choices of the $\mathbf{A}_i$, the homomorphism-condition is satisfied:

7.3.9 Lemma. $\theta_{i,j}: T(i)\uparrow\to T(j)\uparrow$ is a homomorphism.

Proof. Suppose that A is an atomic sentence over TM(i) and $\theta=\theta_{i,j}$.
We have to show, that $A\in T(i)\uparrow$ implies $A\theta\in T(j)\uparrow$.

Consider $X:=\{A\in FM(i)|A\theta\in T(j)\uparrow\}$.
It suffices to show, that $X\models P$, since then, $T(i)\uparrow\subset X$ and we are done.
So, suppose that $C(x,y)\rightarrow R(\sigma(x))$ is some P-rule.
Assume that $C(t,s)\subset X$. We have to show, that $R(\sigma(t))\in X$.
By definition of X, $C(t,s)\theta\subset T(j)\uparrow$.
Since $T(j)\uparrow\models P$, we obtain $R(\sigma(t))\theta\in T(j)\uparrow$. Hence, $R(\sigma(t))\in X$. ⊠
This result however may seem nicer than it really is. For, notice that for sensible programs, the rules are such that every variable in a rule-head also will occur in the corresponding body. But then we will have $T(i)\uparrow=T\uparrow\subset HU$, trivializing the result.

7.3.10 Lemma. $\theta_{i,j}: T(i)\downarrow\rightarrow T(j)\downarrow$ is a homomorphism.
Proof. Suppose again that A is an atomic sentence over TM(i) and $\theta=\theta_{i,j}$. We have to show, that $A\in T(i)\downarrow$ implies $A\theta\in T(j)\downarrow$. But this follows immediately from the
7.3.11 Lemma. $A\in T(i)\downarrow\xi \Rightarrow A\theta\in T(j)\downarrow\xi$.
Proof. Induction w.r.t. ξ. As usual, the cases $\xi=0$ and ξ a limit are trivial. So, assume $A\in T(i)\downarrow\xi+1$. Then for some rule-instance $C\rightarrow A$, $C\subset T(i)\downarrow\xi$. By induction hypothesis, $C\theta\subset T(j)\downarrow$. But then $C\theta\rightarrow A\theta$ is a rule-instance as well, whence, $A\theta\in T(j)\downarrow\xi+1$. ⊠

We can get some more information on the greatest-fixed-point case. The next result is obvious, once one views the *variables* $x\in V(i)$ as *constants* of TM(i) about which the program does not provide any specific information.
7.3.12 Lemma. If $x\in V(i)$ and $A(x)\in T(i)\downarrow\xi$ then $A(y)\in T(i)\downarrow\xi$ for any other $y\in V(i)$ and $A(c)\in T(i)\downarrow\xi$ for any constant c.
Proof. Easy induction w.r.t. ξ. ⊠
7.3.13 Lemma. If $t,s,\ldots\in TM(i)\cap TM(j)$ and $A=R(t,s,\ldots)$, then
$A\in TM(i)\downarrow\xi \Leftrightarrow A\in TM(j)\downarrow\xi$.

7.3.14 Lemma. Suppose that $\varphi(x,\ldots)$ is a positive quantifier-free formula.
Then the following are equivalent for $t,\ldots\in TM(i)$:
1. $Lim(\mathbf{A},\theta)\models\varphi(|t|,\ldots)$
2. $\exists j\geqslant i$: $\mathbf{A}_j\models\varphi(t\theta_{i,j},\ldots)$.

Proof. Immediate from the definition. ⊠

The following result, telling which sentences are *preserved* by the construction, is known.
7.3.15 Corollary. Supose that φ and ψ are as in 7.3.14.
If for every i: $\mathbf{A}_i\models\forall x[\varphi(x)\rightarrow\exists y\psi(x,y)]$

then $\mathrm{Lim}(\mathbf{A},\theta)\models\forall x(\varphi(x)\rightarrow\exists y\psi(x,y))$.

Proof. Suppose that for every i: $\mathbf{A}_i\models\forall x[\varphi(x)\rightarrow\exists y\psi(x,y)]$.
Take any $|t|\in\mathrm{Lim}(\theta)$ such that $\mathrm{Lim}(\mathbf{A},\theta)\models\varphi(|t|)$.
Say, $t\in TM(i)$.
Then $j\geqslant i$ exists such that $\mathbf{A}_j\models\varphi(t\theta_{i,j})$.
By hypothesis, $s\in TM(j)$ exists such that $\mathbf{A}_j\models\varphi(t\theta_{i.j},s)$
But then, $\mathrm{Lim}(\mathbf{A},\theta)\models\psi(|t|,|s|)$; i.e., $\mathrm{Lim}(\mathbf{A},\theta)\models\exists y\psi(|t|,y)$. ⊠

7.3.16 Remark. [Keisler1960] proves that each theory which is preserved by direct limits (of type ω) has a set of axioms of the form $\forall x(\varphi\rightarrow\exists y\psi)$ with φ,ψ positive quantifier-free.
Cf. also [Chang/Keisler 1990] p.322, exercise 5.2.24.

7.3.17 Corollary. If each $\mathbf{A}_i$ is a fixed point of T(i), then

$$\mathrm{Lim}(\mathbf{A},\theta)\models\mathrm{Comp}(P).$$

Proof. Except some of the FEQ-axioms, the sentences in Comp(P) - the completion of P - are of the form required and are satisfied by every A_i. ⊠

7.3.18. "Completeness of negation as failure" is the statement that for every $A\in T{\downarrow}\omega$ there is a model of A+Comp(P).
The usual proof employs the limit of mgu's along an infinite fair derivation of A, turned into a model in still another way. We strengthen this slightly, obtaining a model satisfying the domain closure axiom as well. Of course, this also follows from [Blair/Brown 199?], which is discussed in 7.4.
So, assume θ is the sequence of mgu's along an infinite *fair* derivation of the ground atom $A\in T{\downarrow}\omega$ (such a derivation exists by the finite failure theorem).
Define T* to be the monotone operator over the Herbrand-base associated with $\mathrm{Lim}(\theta)$ induced by P.
To describe T* a little more precisely, we admit all variables of $V=\bigcup_i V(i)$ as individual constants, $x\in V$ interpreted as $|x|$ in $\mathrm{Lim}(\theta)$.
Let $HB(\theta)$ be the Herbrand base over $\mathrm{Lim}(\theta)$, i.e., the set of all atomic formulas A s.t. $\mathrm{Var}(A)\subset V$. Of course, we identify R(t,...) and R(s,...) in case $t\approx s$,....
Now, for $X\subset HB(\theta)$, $T(X)\subset HB(\theta)$ is defined by: $A\in T(X)$ iff for some instance $C\rightarrow A$ of a P-rule: $C\subset X$.
$T^*{\uparrow}$ and $T^*{\downarrow}$ are the least, resp. greatest fixed points of T*. Obviously, these are models of Comp(P).
$\mathrm{Lim}(\uparrow,\theta)$ is the model produced by the $TM(i){\uparrow}$, $\mathrm{Lim}(\downarrow,\theta)$ the one produced by the $TM(i){\downarrow}$.

7.3.19 Lemma. $T^*{\uparrow}=\mathrm{Lim}(\uparrow,\theta)\subset\mathrm{Lim}(\downarrow,\theta)\subset T^*{\downarrow}$.

Proof of last inclusion: suppose that $Lim(\downarrow,\theta)\models R(t,s)$. Then for some i $R(t,s)\theta_{0,i}\in T(i)\downarrow$. The result now is immediate from the

7.3.20 Lemma. If $A\in HB(i)$ then for all ξ

$$A\in T(i)\downarrow\xi \Rightarrow A\in T^*\downarrow\xi.$$

The converse of this fails, even for $A\in HB$. This is illustrated by the program $\{px\rightarrow pSx;\ px\rightarrow pO\}$: $T(i)\downarrow=T(i)\downarrow\omega+1=\emptyset$ for all i; however, $p(x^1)\in T^*\downarrow\omega$, and hence $p(O)\in T^*\downarrow\omega+1$.

7.3.21 Completeness of negation as failure.

Let τ be the infinite fair derivation of $A\in T\downarrow\omega$ from which θ is taken. Each literal in each clause from each goal on τ is satisfied by the assignment $|.|: V\rightarrow Lim(\theta)$ in the model $T^*\downarrow$.

Proof. If not, there is a least ξ such that for some atom A occurring in the derivation: $A\notin T^*\downarrow\xi+1$.

From A, follow the derivation downwards until you find the selected descendant $A\theta_{i,j}$ of A. Suppose the P-rule $D\rightarrow R$ is applied there using $\theta_{j,j+1}$. Then $D\theta_{j,j+1}\rightarrow R\theta_{j,j+1}$ is a rule-instance; $R\theta_{j,j+1}=A\theta_{i,j+1}$ is identified with A. We must have $T^*\downarrow\xi\not\models D\theta_{j,j+1}$, contradicting the choice of ξ. ⊠

We know that $T^*\downarrow$ will be a model of Comp(P). It is not necessarily a model of the domain closure axiom; however, taking one more quotient remedies this: by 7.3.7 the only thing needed is to identify each variable in $Con(\theta)$ (if any) with some individual constant. Therefore

7.3.22. Theorem. Every $A\in T\downarrow\omega$ is satisfied in a model of Comp(P) and the domain closure axiom.

Varying the construction suitably gives:

7.3.23 Theorem ([Blair/Brown 199?], with a different proof.)

We can construct θ so, that $T^*\downarrow$ is a model of Comp(P), the domain closure axiom, and *all of* $T\downarrow\omega$.

Proof. Note that 7.3.21 produces a model of one atom in $T\downarrow\omega$ only. Fix an enumeration of $T\downarrow\omega$. Start a fair derivation using the first element of this enumeration for an initial goal. Add the i-th element of the enumeration both as a last literal to the i-th goal obtained and to the stack regulating fairness. ⊠

7.4. Marginal remarks to a theorem of Blair and Brown.

Contents: we note that the results of [Blair/Brown 199?] can be obtained immediately from certain results in [Barwise 1975] on recursive saturation and inductive definability, in particular, Gandy's theorem. Subsequently, the recursive saturation-condition is relaxed.

7.4.1. [Blair/Brown 199?] constructs a countable FUNC-algebra **M** such that the following conditions are satisfied.

1. **M** is a quotient of the free algebra over the set of all terms induced by a collection of (idempotent) substitutions; in particular, $HU \subset \mathbf{M}$.

2. **M** satisfies all free equality axioms.

3. The immediate consequence operator T^* induced over **M** by an arbitrary logic program always has downward closure ordinal $\leqslant \omega$ (i.e., $T^*{\downarrow} = T^*{\downarrow}\omega$).

N.B. A program P is called *canonical* by [Jaffar/Stuckey 1986] whenever $T{\downarrow} = T{\downarrow}\omega$, where $T = T_P$. This explains the title of [Blair/Brown 199?].
Finally, assuming an individual constant 0, a unary function symbol S and a binary function symbol J to be present in FUNC:

4. Every total recursive function is represented by an element of **M**.

Here, we say that $a \in \mathbf{M}$ **represents** $f:\mathbb{N}\to\mathbb{N}$ iff for some (by the free equality axioms, *unique*) sequence $a_0 = a, a_1, a_2, \ldots$ exists in **M** such that $a_i = J(f(i), a_{i+1})$ $(i \in \mathbb{N})$, identifying each $k \in \mathbb{N}$ with the term S^k0.
(The definition of [Blair/Brown 199?] is, in an inessential way, slightly more complicated.)

7.4.2. In [Blair/Brown 199?], **M** is constructed as a limit of quotients of which the model from 7.4.23 is the first step. It is easy to see, that, in this model, the restricted hierarchy $T^*{\downarrow} \cap HB$, where HB is the Herbrand base over the standard algebra HU, closes in $\leqslant \omega$ steps; in fact: $T^*{\downarrow} \cap HB = T{\downarrow}\omega = T^*{\downarrow}\omega \cap HB$.
Iterating the construction (this needs relativizing the unification-procedure and SLD-resolution to an arbitrary algebra) eventually produces the desired algebra.

7.4.3. Here is a shortcut to such an algebra.
In order to obtain 7.4.1.1-4 above, it suffices for **M** to be a *countable recursively saturated elementary extension* of the free algebra HU.
For elementary results on recursive saturation, cf. [Chang/Keisler 1990]. In particular, it is proved there that every countable model has a countable recursively saturated elementary extension.
That **M** satisfies the free equality axioms is now clear. It even satisfies the domain closure axiom.
That **M** is a quotient of the free algebra of all terms modulo a sequence of idempotent substitutions is immediate from our representation result 7.1.7 above.
Representation of recursive functions follows from recursive saturation. If $f:\mathbb{N}\to\mathbb{N}$ is recursive, consider the recursive type $\tau = \tau(x_0)$ consisting of the formulas

$$\exists x_1 [\, x_0 = J(f(0), x_1)\,]$$

$$\exists x_1 \exists x_2 [\, x_0 = J(f(0), x_1) \wedge x_1 = J(f(1), x_2)\,]$$

$$\exists x_1 \exists x_2 \exists x_3 [x_0 = J(f(0),x_1) \wedge x_1 = J(f(1),x_2) \wedge x_2 = J(f(2),x_3)]$$

etc.

Obviously, τ is finitely satisfiable in HU; hence it is satisfied in **M** by recursive saturation; equally obviously, an element satisfies τ iff it represents f.

7.4.4. It remains to show that the operator T^* induced over the Herbrand-base corresponding to **M** by an arbitrary program P has downward closure ordinal ω.

This follows from two quotable results in Barwise [1975]. Unfortunately, these results involve the ordinal $O(\mathbf{M})$, the analogue of ω_1^{CK} when $\mathbf{M} = (\mathbb{N},0,S)$.

$O(\mathbf{M})$ is the height of $HYP(\mathbf{M})$, the least transitive model of the weak set theory KPU^+ which has **M** as an element. For the precise definition, cf. [Barwise 1975]. *Least* models exist here, they are initials of Gödel's constructible hierarchy starting on **M** instead of the empty set.

We have the

7.4.5 Theorem ([Barwise 1975] Thm. 5.3 p.139)

M is recursively saturated iff $O(\mathbf{M}) = \omega$.

Of this, we only need (a special case of) the $\Rightarrow$-part, which has a rather complicated proof in [Barwise 1975]. A more easy one can be obtained as follows. We know that **M** is countably infinite. By the strong form of *resplendency* of **M** (i.e., [Chang/Keisler 1990] Thm.2.4.10 p.120), we can expand **M** into a model of KPU^+ in which **M** is represented as an element; and, using compactness along the way, we can take care that the only well-founded ordinals of the model are its natural numbers. It follows that the *standard part* of this model has height ω. By the *Truncation Lemma*, cf. Barwise l.c., it is a model of KPU^+ as well, and the result folows.

7.4.6. Now, by [Barwise 1975] Thm.3.11 p.211 ("Gandy's theorem"), every *extended inductive* definition over a model **A** has closure ordinal $\leqslant O(\mathbf{A})$; in particular, extended inductive definitions over **M** close at ω at the latest.

The extended inductive definitions include those in which the monotone operator is defined by a first-order positive formula on the structure under consideration.

It is not difficult to see that this includes the *duals* of program-induced operators. Though the set-up in logic programming differs from that in first-order positive inductive definability, it is straightforward to translate (the complement of) $T^*\downarrow$ into a first-order positive inductive definition with the same closure ordinal: if the program defines, say, relations $R_1,\ldots,R_k$ with resp. $r_1,\ldots,r_k$ arguments, the inductive definition now introduces *one* relation of $r_1+\ldots+r_k$ arguments, the first r_1 arguments

of which are used to represent R_1, the next r_2 to represent R_2, etc. In fact, the formula needed only uses *universal* quantifiers (cf. the discussion in part 5 on co-computability).

In order to obtain algebras satisfying the Blair/Brown conditions, the formulation of Gandy's theorem suggests that recursive saturation is probably much more than is needed here.

7.4.7. Consider the following

Definitions.

1. **C** is **equationally weakly recursively saturated** iff every recursive set E of *parameter-free* equations with possibly infinitely many variables finitely satisfied in **C** is realized in **C**.

2. **C** is **weakly recursively complete** iff each recursive neatly descending sequence $\theta_0 \geqslant^* \theta_1 \geqslant^* \theta_2 \geqslant^* \ldots$ of idempotent substitutions is matched in **C**
(i.e., an assignment α:VAR$\rightarrow$**C** exists s.t. $\alpha = \theta_i\alpha$ for all i).

3. **C** is **weakly canonical** iff for each logic program P: $T{\downarrow}\omega \subset T^*{\downarrow}$
(here, $T=T_P$ is the operator induced by P over HU; T^* is the operator induced by P over **C**).

Theorem. For countable algebras satisfying the free equality-axioms:

1. recursive saturation implies equational weak recursive saturation;
2. equational weak recursive saturation amounts to weak recursive completeness;
3. weak recursive completeness implies weak cononicity.

Proof. Fix an algebra **C** satisfying all free equality axioms.

1. Immediate from strong resplendency of **C** (i.e., [Chang/Keisler 1990] Thm.2.4.10 p.120). (This uses countability of **C**.)

2. $\Leftarrow$. Suppose that $E=\{e_0, e_1, e_2, \ldots\}$ is a recursive set of equations finitely satisfied in **C**.
Consider the Martelli-Montanari algorithm for obtaining idempotent mgu's. If E' is a finite set of equations, MM(E') denotes the (a) solved set of equations equivalent with E' (modulo free equality-axioms) which is produced by the algorithm - that is, if unification is possible at all. Note that, if E' is satisfiable in **C**, since **C** satisfies the free equality axioms, the MM-algorithm cannot fail to succeed on E' and MM(E') is defined.
Now define

$E_0 = MM\{e_0\}$: this produces an idempotent θ_0

$E_1 = MM(E(\theta_0) \cup \{e_1\theta_0\})$: this produces an idempotent $\theta_1 \leqslant^* \theta_0$
(cf. the proof of lemma 7.1.7)

$E_2 = MM(E(\theta_1) \cup \{e_2\theta_1\})$: this produces $\theta_2 \leqslant^* \theta_1$

etc.;

Now, apply weak recursive completeness.

2.$\Rightarrow$. Suppose that $\theta_0 \geqslant^* \theta_1 \geqslant^* \theta_2 \geqslant^* \ldots$ is a recursive descending sequence of idempotent substitutions; let E_i be the set of equations associated with θ_i. Apply equational weak recursive saturation to $\bigcup_i E_i$.

3. Just copy the proof of "completeness of negation as failure". If θ is the (by standardization-apart, neatly descending) sequence of successive compositions of mgu's along an infinite fair derivation of $A \in T{\downarrow}\omega$, by weak recursive completeness θ has a matching assignment α in **C**. It now follows that α satisfies in $T^*{\downarrow}$ every atom in every goal of the derivation: if not, a *least* ξ exists such that for some such atom B, $T^*{\downarrow}\xi \not\models B[\alpha]$; now, go down the derivation until you find the selected descendant of B and argue as in the proof of 7.3.21. ⊠

The following looks as a - not so easy - exercise in programming:

Conjecture. For **C** satisfying the free equality axioms:

weak canonicity implies weak recursive completeness.

7.4.8 Relativizing the discussion of 7.4.7:

Definitions.

1'. **C** is **equationally recursively saturated** iff every recursive set E of equations with finitely many parameters from **C** and with possibly infinitely many variables finitely satisfied in **C** is realized in **C**.

2'. **C** is **recursively complete** iff each recursive neatly descending sequence $\theta_0 \geqslant^* \theta_1 \geqslant^* \theta_2 \geqslant^* \ldots$ of idempotent substitutions using finitely many parameters from **C** is matched in **C**.

3'. **C** is **canonical** iff for each logic program P: $T^*{\downarrow}\omega \subset T^*{\downarrow}$.

Theorem. For countable algebras satisfying the free equality-axioms:

1. recursive saturation implies equational recursive saturation;
2. equational recursive saturation amounts to recursive completeness;
3. recursive completeness implies canonicity.

Proof. 1. As before. 2/3 Also as before, but now using the relativized machinery of [Blair/Brown 199?] on **M**-unification (needed for 2,3), -resolution and -finite failure (needed for 3), i.e., section 4 of that paper. Note that we can simplify the exposition given there slightly by skipping the taking of the first quotient corresponding to $\sim$ resp. $^\circ$ by *not* expanding the algebra **M** with *all* its elements as additional constants but just expanding with elements of **M** *not in* HU. ⊠

Again, we have a

Conjecture. For **C** satisfying the free equality axioms:

canonicity implies recursive completeness.

Example. Consider the simplest non-trivial case where FUNC={0,S}.

The Herbrand universe here is nothing but the set $\mathbb{N}$ with 0 and successor. Let $\mathbb{N}+\mathbb{Z}$ denote the model of FEQ consisting of $\mathbb{N}$ plus one copy of the integers $\mathbb{Z}$ with successor.

This model is far from being resursively saturated (the countable recursively saturated equivalent $\mathbb{N}+\mathbb{Z}\times\omega$ of $\mathbb{N}$ must contain a countably infinite number of copies of $\mathbb{Z}$).

However, it is *equationally* saturated:

Lemma.

Every set of equations finitely satisfied in $\mathbb{N}$ can be satisfied in $\mathbb{N}+\mathbb{Z}$.

Proof. The homomorphism:$\mathbb{N}+\mathbb{Z}\times\omega\rightarrow\mathbb{N}+\mathbb{Z}$ which identifies all $\mathbb{Z}$-copies preserves satisfaction of equations. ⊠

Remark. Clearly, this is the *only* case where we have equational saturation (instead of mere *recursive* saturation) of a *countable* algebra: as soon as there are two unary functions, or only one function with $\geqslant 2$ arguments, there is a notion of representation of number-theoretic functions by elements in the sense of [Blair/Brown 199?] - cf. 7.4.1.4 - and countable equationally saturated algebras no longer exist.

ACKNOWLEDGEMENTS.
I owe Marc Bezem, Johan van Benthem and Krzysztof Apt for listening to me patiently, and providing me with unpublished manuscripts and criticism on previous versions of this material.

References.

[Apt 1988]:
Apt, K.R.: **Introduction to Logic Programming.** (Revised and Extended Version.) Report CS-R8826 C.W.I. Amsterdam. To appear in: **Handbook of Theoretical Computer Science** (J.van Leeuwen, ed.) North-Holland, Amsterdam.

[Barwise 1975]
Barwise, J.: **Admissible Sets and Structures.** Springer, Berlin etc.

[Bezem 199?]:
Bezem, M. : *Strong Termination of Logic Programs.* Preprint. (A preliminary version is Bezem [1989].)

[Bezem 1989]:
Bezem, M. : *Characterizing Termination of Logic Programs with Level Mappings,* in: Proceedings of the North American Conference on Logic Programming, Cleveland, Ohio, pp. 69-80.

[Blair 1982]:
Blair, H.A. : *The Recursion-Theoretic Complexity of Predicate Logic as a Programming Language.* Inf. and Control 54 pp.25-47.

[Blair 1986]:
Blair, H.A. : *Decidability in the Herbrand Base,* manuscript (presented at the Workshop on Foundations of Deductive Databases and Logic Programming, Washington D.C., August 1986).

[Blair/Brown 199?]:
Blair, H.A. and A.L. Brown Jr.: *Definite Clause Programs* are *Canonical (Over a Suitable Domain).* Manuscript.

[Chang/Keisler 1990]:
Chang, C.C., and H.J. Keisler : **Model Theory.** North-Holland, Amsterdam etc.

[Jaffar/Stuckey 1986]:
Jaffar, J. and P.J. Stuckey: *Canonical Logic Programs.* J. of Logic Programming 3 (1986) pp.143-155.

[Keisler 1960]:

Keisler, H.J. : *Theory of models with generalized atomic formulas.* J.Symb.Logic 25 (1960), pp.1-26.

[Lloyd 1987]:

Lloyd, J.W. : **Foundations of Logic programming**, second, extended edition. Springer, Berlin etc.

[Moschovakis 1974]:

Moschovakis, Y.N. : **Elementary Induction on Abstract Structures.** North-Holland, Amsterdam etc. 1974.

[Shoenfield 1967]:

Shoenfield, J.R. : **Mathematical Logic.** Addison-Wesley, Reading, Mass.

The ITLI Prepublication Series

1990

Logic, Semantics and Philosophy of Language

LP-90-01 Jaap van der Does — A Generalized Quantifier Logic for Naked Infinitives
LP-90-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar
LP-90-03 Renate Bartsch — Concept Formation and Concept Composition
LP-90-04 Aarne Ranta — Intuitionistic Categorial Grammar
LP-90-05 Patrick Blackburn — Nominal Tense Logic
LP-90-06 Gennaro Chierchia — The Variablity of Impersonal Subjects
LP-90-07 Gennaro Chierchia — Anaphora and Dynamic Logic
LP-90-08 Herman Hendriks — Flexible Montague Grammar
LP-90-09 Paul Dekker — The Scope of Negation in Discourse, towards a flexible dynamic Montague grammar

LP-90-10 Theo M.V. Janssen — Models for Discourse Markers
LP-90-11 Johan van Benthem — General Dynamics
LP-90-12 Serge Lapierre — A Functional Partial Semantics for Intensional Logic
LP-90-13 Zisheng Huang — Logics for Belief Dependence

Mathematical Logic and Foundations

ML-90-01 Harold Schellinx — Isomorphisms and Non-Isomorphisms of Graph Models
ML-90-02 Jaap van Oosten — A Semantical Proof of De Jongh's Theorem
ML-90-03 Yde Venema — Relational Games
ML-90-04 Maarten de Rijke — Unary Interpretability Logic
ML-90-05 Domenico Zambella — Sequences with Simple Initial Segments
ML-90-06 Jaap van Oosten — Extension of Lifschitz' Realizability to Higher Order Arithmetic, and a Solution to a Problem of F. Richman

ML-90-07 Maarten de Rijke — A Note on the Interpretability Logic of Finitely Axiomatized Theories
ML-90-08 Harold Schellinx — Some Syntactical Observations on Linear Logic
ML-90-09 Dick de Jongh, Duccio Pianigiani — Solution of a Problem of David Guaspari
ML-90-10 Michiel van Lambalgen — Randomness in Set Theory

Computation and Complexity Theory

CT-90-01 John Tromp, Peter van Emde Boas — Associative Storage Modification Machines
CT-90-02 Sieger van Denneheuvel, Gerard R. Renardel de Lavalette — A Normal Form for PCSJ Expressions

CT-90-03 Ricard Gavaldà, Leen Torenvliet, Osamu Watanabe, José L. Balcázar — Generalized Kolmogorov Complexity in Relativized Separations
CT-90-04 Harry Buhrman, Leen Torenvliet — Bounded Reductions
CT-90-05 Sieger van Denneheuvel, Karen Kwast — Efficient Normalization of Database and Constraint Expressions
CT-90-06 Michiel Smid, Peter van Emde Boas — Dynamic Data Structures on Multiple Storage Media, a Tutorial
CT-90-07 Kees Doets — Greatest Fixed Points of Logic Programs

Other Prepublications

X-90-01 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics, Revised Version

X-90-02 Maarten de Rijke — Some Chapters on Interpretability Logic
X-90-03 L.D. Beklemishev — On the Complexity of Arithmetical Interpretations of Modal Formulae
X-90-04 — Annual Report 1989
X-90-05 Valentin Shehtman — Derived Sets in Euclidean Spaces and Modal Logic
X-90-06 Valentin Goranko, Solomon Passy — Using the Universal Modality: Gains and Questions
X-90-07 V.Yu. Shavrukov — The Lindenbaum Fixed Point Algebra is Undecidable
X-90-08 L.D. Beklemishev — Provability Logics for Natural Turing Progressions of Arithmetical Theories

X-90-09 V.Yu. Shavrukov — On Rosser's Provability Predicate
X-90-10 Sieger van Denneheuvel, Peter van Emde Boas — An Overview of the Rule Language RL/1

X-90-11 Alessandra Carbone — Provable Fixed points in $I\Delta_0+\Omega_1$, revised version
X-90-12 Maarten de Rijke — Bi-Unary Interpretability Logic
X-90-13 K.N. Ignatiev — Dzhaparidze's Polymodal Logic: Arithmetical Completeness, Fixed Point Property, Craig's Property

X-90-14 L.A. Chagrova — Undecidable Problems in Correspondence Theory