Lecture Notes in Artificial Intelligence

Subseries of Lecture Notes in Computer Science

Edited by J. Siekmann

345

Rolf T. Nossum (Ed.)

Advanced Topics in Artificial Intelligence

2nd Advanced Course, ACAI '87
Oslo, Norway, July/August 1987

Springer-Verlag

Lecture Notes in Artificial Intelligence

Subseries of Lecture Notes in Computer Science
Edited by J. Siekmann

345

Rolf T. Nossum (Ed.)

Advanced Topics in Artificial Intelligence

2nd Advanced Course, ACAI '87
Oslo, Norway, July 28 – August 7, 1987

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo

Editor

Rolf T. Nossum
Oslo College of Engineering
Cort Adelers gate 30, N-0254 Oslo 2, Norway

2nd Advanced Course in Artificial Intelligence, ACAI '87

Organizer

European Coordinating Committee for AI (ECCAI)

CR Subject Classification (1987): I.2.1, I.2.3–4, I.2.6–7

ISBN 3-540-50676-4 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-50676-4 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 345: R.T. Nossum (Ed.), Advanced Topics in Artificial Intelligence. VII, 233 pages. 1988.

Vol. 346: M. Reinfrank, J. de Kleer, M.L. Ginsberg, E. Sandewall (Eds.), Non-Monotonic Reasoning. Proceedings, 1988. XIV, 237 pages. 1989.

Vol. 347: K. Morik (Ed.), Knowledge Representation and Organization in Machine Learning. XV, 319 pages. 1989.

Other volumes of the Lecture Notes in Computer Science relevant to Artificial Intelligence:

Vol. 203: B. Buchberger (Ed.), EUROCAL '85. Proceedings Vol. 1, 1985. V, 233 pages. 1985.

Vol. 204: B.F. Caviness (Ed.), EUROCAL '85. Proceedings Vol. 2, 1985. XVI, 650 pages. 1985.

Vol. 221: E. Wada (Ed.), Logic Programming '85. Proceedings, 1985. IX, 311 pages. 1986.

Vol. 225: E. Shapiro (Ed.), Third International Conference on Logic Programming. Proceedings, 1986. IX, 720 pages. 1986.

Vol. 230: J.H. Siekmann (Ed.), 8th International Conference on Automated Deduction. Proceedings, 1986. X, 708 pages. 1986.

Vol. 231: R. Hausser, NEWCAT: Parsing Natural Language Using Left-Associative Grammar. II, 540 pages. 1986.

Vol. 238: L. Naish, Negation and Control in Prolog. IX, 119 pages. 1986.

Vol. 256: P. Lescanne (Ed.), Rewriting Techniques and Applications. Proceedings, 1987. VI, 285 pages. 1987.

Vol. 264: E. Wada (Ed.), Logic Programming '86. Proceedings, 1986. VI, 179 pages. 1987.

Vol. 271: D. Snyers, A. Thayse, From Logic Design to Logic Programming. IV, 125 pages. 1987.

Vol. 286: B. Bouchon, R.R. Yager (Eds.), Uncertainty in Knowledge-Based Systems. Proceedings, 1986. VII, 405 pages. 1987.

Vol. 301: J. Kittler (Ed.), Pattern Recognition. Proceedings, 1988. VII, 668 pages. 1988.

Vol. 306: M. Boscarol, L. Carlucci Aiello, G. Levi (Eds.), Foundations of Logic and Functional Programming. Proceedings, 1986. V, 218 pages. 1988.

Vol. 308: S. Kaplan, J.-P. Jouannaud (Eds.), Conditional Term Rewriting Systems. Proceedings, 1987. VI, 278 pages. 1988.

Vol. 310: E. Lusk, R. Overbeek (Eds.), 9th International Conference on Automated Deduction. Proceedings, 1988. X, 775 pages. 1988.

Vol. 313: B. Bouchon, L. Saitta, R.R. Yager (Eds.), Uncertainty and Intelligent Systems. IPMU '88. Proceedings, 1988. VIII, 408 pages. 1988.

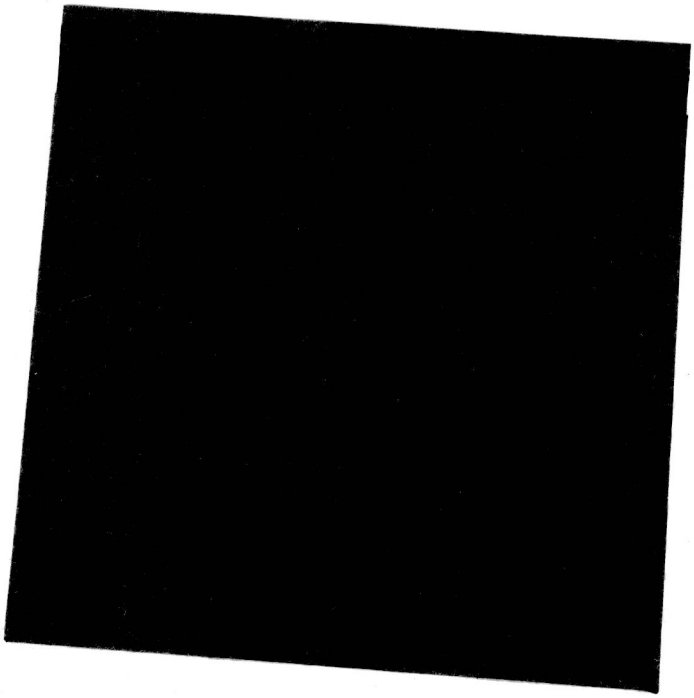
Vol. 315: K. Furukawa, H. Tanaka, T. Fujisaki (Eds.), Logic Programming '87. Proceedings, 1987. VI, 327 pages. 1988.

Vol. 320: A. Blaser (Ed.), Natural Language at the Computer. Proceedings, 1988. III, 176 pages. 1988.

Lecture Notes in Artificial Intelligence

Subseries of Lecture Notes in Computer Science

Edited by J. Siekmann



Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

Editorial

Artificial Intelligence has become a major discipline under the roof of Computer Science. This is also reflected by a growing number of titles devoted to this fast developing field to be published in our Lecture Notes in Computer Science. To make these volumes immediately visible we have decided to distinguish them by a special cover as Lecture Notes in Artificial Intelligence, constituting a subseries of the Lecture Notes in Computer Science. This subseries is edited by an Editorial Board of experts from all areas of AI, chaired by Jörg Siekmann, who are looking forward to consider further AI monographs and proceedings of high scientific quality for publication.

We hope that the constitution of this subseries will be well accepted by the audience of the Lecture Notes in Computer Science, and we feel confident that the subseries will be recognized as an outstanding opportunity for publication by authors and editors of the AI community.

Editors and publisher

PREFACE

Artificial Intelligence being a rapidly developing scientific field, there continually exists a need for rapidly disseminating its latest advances to students and practitioners.

Conferences and conference proceedings are a relatively rapid communication channel, but often the quality of the material presented in them is variable, and the style of presentation quite demanding.

The biennial Advanced Courses in Artifical Intelligence, organized under the auspices of ECCAI, the European Coordinating Committee for Artificial Intelligence, provide a forum for communicating the latest results of the field. World-class AI scientists are invited to lecture, and are asked to provide written accounts of their lectures for publication. The present volume draws on the material presented at the second Advanced Course, which was held in Oslo, Norway, in July 1987.

In contrast to the first Advanced Course, reported as Springer Lecture Notes in Computer Science vol. 232, which covered the foundations of AI extensively, the second Advanced Course emphasized in-depth treatment of a selection of special topics in AI. Inevitably, the course also contained reports on advances in some of the same areas as were covered in the first course.

Philippe Jorrand's chapter gives a unified view of computational mechanisms based on syntactic manipulation of algebraic terms. Dis- and anti-unification are introduced, and their significance explained by way of examples. Term rewriting systems, functional programming systems, and logic programming systems arise as instances of a common computational foundation.

The chapter by Wolfgang Bibel contains new material on enhancements of matrix-based automated deduction, and tells of recent research on how it extends to non-standard logics. A case is made that matrix-based deduction is particularly amenable to this kind of extension.

In the chapter on Qualitative Reasoning, Tony Cohn surveys the state of the art of this important and rapidly developing field as it appears in the summer of 1987.

The chapter on Knowledge Acquisition, by Bob Wielinga and his co-workers, focuses on a major bottleneck in the practical deployment of AI in the form of Expert Systems. It contains a comprehensive overview of elicitation techniques in use today, and stresses those that result in formal specification of knowledge bases.

Alan Biermann's chapter on Learning Systems presents a unified framework for classifying such systems. The reader may wish to refer to his chapter in vol. 232 of the Springer Lecture Notes in Computer Science for background material.

Sam Steel gives an up-to-date and in-depth treatment of the essential Topics in Planning, in the chapter so named.

The chapter on Natural Language Systems by Jens Erik Fenstad gives insight into the situational semantics approach. The reader will appreciate the relationship with other branches of AI, for instance, with AI planning, and the plan-based theory of action, both of which, like situational semantics, appear to owe much to a logical framework laid in the 60s by McCarthy.

Springer-Verlag deserve a special word of acknowledgement for recognizing the value of the material contained in this volume, and offering their very efficient publishing services. This adds to the feeling of satisfaction that comes from organizing an event like the Advanced Course in Artificial Intelligence.

Rolf Nossum, Oslo.

Lecture Notes in Computer Science

Vol. 296: R. Janßen (Ed.), Trends in Computer Algebra. Proceedings, 1987. V, 197 pages. 1988.

Vol. 297: E.N. Houstis, T.S. Papatheodorou, C.D. Polychronopoulos (Eds.), Supercomputing. Proceedings, 1987. X, 1093 pages. 1988.

Vol. 298: M. Main, A. Melton, M. Mislove, D. Schmidt (Eds.), Mathematical Foundations of Programming Language Semantics. Proceedings, 1987. VIII, 637 pages. 1988.

Vol. 299: M. Dauchet, M. Nivat (Eds.), CAAP '88. Proceedings, 1988. VI, 304 pages. 1988.

Vol. 300: H. Ganzinger (Ed.), ESOP '88. Proceedings, 1988. VI, 381 pages. 1988.

Vol. 301: J. Kittler (Ed.), Pattern Recognition. Proceedings, 1988. VII, 668 pages. 1988.

Vol. 302: D.M. Yellin, Attribute Grammar Inversion and Source-to-source Translation. VIII, 176 pages. 1988.

Vol. 303: J.W. Schmidt, S. Ceri, M. Missikoff (Eds.), Advances in Database Technology – EDBT '88. X, 620 pages. 1988.

Vol. 304: W.L. Price, D. Chaum (Eds.), Advances in Cryptology – EUROCRYPT '87. Proceedings, 1987. VII, 314 pages. 1988.

Vol. 305: J. Biskup, J. Demetrovics, J. Paredaens, B. Thalheim (Eds.), MFDBS 87. Proceedings, 1987. V, 247 pages. 1988.

Vol. 306: M. Boscarol, L. Carlucci Aiello, G. Levi (Eds.), Foundations of Logic and Functional Programming. Proceedings, 1986. V, 218 pages. 1988.

Vol. 307: Th. Beth, M. Clausen (Eds.), Applicable Algebra, Error-Correcting Codes, Combinatorics and Computer Algebra. Proceedings, 1986. VI, 215 pages. 1988.

Vol. 308: S. Kaplan, J.-P. Jouannaud (Eds.), Conditional Term Rewriting Systems. Proceedings, 1987. VI, 278 pages. 1988.

Vol. 309: J. Nehmer (Ed.), Experiences with Distributed Systems. Proceedings, 1987. VI, 292 pages. 1988.

Vol. 310: E. Lusk, R. Overbeek (Eds.), 9th International Conference on Automated Deduction. Proceedings, 1988. X, 775 pages. 1988.

Vol. 311: G. Cohen, P. Godlewski (Eds.), Coding Theory and Applications 1986. Proceedings, 1986. XIV, 196 pages. 1988.

Vol. 312: J. van Leeuwen (Ed.), Distributed Algorithms 1987. Proceedings, 1987. VII, 430 pages. 1988.

Vol. 313: B. Bouchon, L. Saitta, R.R. Yager (Eds.), Uncertainty and Intelligent Systems. IPMU '88. Proceedings, 1988. VIII, 408 pages. 1988.

Vol. 314: H. Göttler, H.J. Schneider (Eds.), Graph-Theoretic Concepts in Computer Science. Proceedings, 1987. VI, 254 pages. 1988.

Vol. 315: K. Furukawa, H. Tanaka, T. Fujisaki (Eds.), Logic Programming '87. Proceedings, 1987. VI, 327 pages. 1988.

Vol. 316: C. Choffrut (Ed.), Automata Networks. Proceedings, 1986. VII, 125 pages. 1988.

Vol. 317: T. Lepistö, A. Salomaa (Eds.), Automata, Languages and Programming. Proceedings, 1988. XI, 741 pages. 1988.

Vol. 318: R. Karlsson, A. Lingas (Eds.), SWAT 88. Proceedings, 1988. VI, 262 pages. 1988.

Vol. 319: J.H. Reif (Ed.), VLSI Algorithms and Architectures – AWOC 88. Proceedings, 1988. X, 476 pages. 1988.

Vol. 320: A. Blaser (Ed.), Natural Language at the Computer. Proceedings, 1988. III, 176 pages. 1988.

Vol. 322: S. Gjessing, K. Nygaard (Eds.), ECOOP '88. European Conference on Object-Oriented Programming. Proceedings, 1988. VI, 410 pages. 1988.

Vol. 323: P. Deransart, M. Jourdan, B. Lorho, Attribute Grammars. IX, 232 pages. 1988.

Vol. 324: M.P. Chytil, L. Janiga, V. Koubek (Eds.), Mathematical Foundations of Computer Science 1988. Proceedings. IX, 562 pages. 1988.

Vol. 325: G. Brassard, Modern Cryptology. VI, 107 pages. 1988.

Vol. 326: M. Gyssens, J. Paredaens, D. Van Gucht (Eds.), ICDT '88. 2nd International Conference on Database Theory. Proceedings, 1988. VI, 409 pages. 1988.

Vol. 327: G.A. Ford (Ed.), Software Engineering Education. Proceedings, 1988. V, 207 pages. 1988.

Vol. 328: R. Bloomfield, L. Marshall, R. Jones (Eds.), VDM '88. VDM – The Way Ahead. Proceedings, 1988. IX, 499 pages. 1988.

Vol. 329: E. Börger, H. Kleine Büning, M.M. Richter (Eds.), CSL '87. 1st Workshop on Computer Science Logic. Proceedings, 1987. VI, 346 pages. 1988.

Vol. 330: C.G. Günther (Ed.), Advances in Cryptology – EUROCRYPT '88. Proceedings, 1988. XI, 473 pages. 1988.

Vol. 331: M. Joseph (Ed.), Formal Techniques in Real-Time and Fault-Tolerant Systems. Proceedings, 1988. VI, 229 pages. 1988.

Vol. 332: D. Sannella, A. Tarlecki (Eds.), Recent Trends in Data Type Specification. V, 259 pages. 1988.

Vol. 333: H. Noltemeier (Ed.), Computational Geometry and its Applications. Proceedings, 1988. VI, 252 pages. 1988.

Vol. 334: K.R. Dittrich (Ed.), Advances in Object-Oriented Database Systems. Proceedings, 1988. VII, 373 pages. 1988.

Vol. 335: F.A. Vogt (Ed.), CONCURRENCY 88. Proceedings, 1988. VI, 401 pages. 1988.

Vol. 337: O. Günther, Efficient Structures for Geometric Data Management. XI, 135 pages. 1988.

Vol. 338: K.V. Nori, S. Kumar (Eds.), Foundations of Software Technology and Theoretical Computer Science. Proceedings, 1988. IX, 520 pages. 1988.

Vol. 339: M. Rafanelli, J.C. Klensin, P. Svensson (Eds.), Statistical and Scientific Database Management. Proceedings, 1988. IX, 454 pages. 1988.

Vol. 340: G. Rozenberg (Ed.), Advances in Petri Nets 1988. VI, 439 pages. 1988.

Vol. 341: S. Bittanti (Ed.), Software Reliability Modelling and Identification. VII, 209 pages. 1988.

Vol. 342: G. Wolf, T. Legendi, U. Schendel (Eds.), Parcella '88. Proceedings, 1988. 380 pages. 1989.

Vol. 343: J. Grabowski, P. Lescanne, W. Wechler (Eds.), Algebraic and Logic Programming. Proceedings, 1988. 278 pages. 1988.

Vol. 344: J. van Leeuwen, Graph-Theoretic Concepts in Computer Science. Proceedings, 1988. VII, 459 pages. 1989.

Vol. 345: see inside front cover (LNAI).

Vol. 346: see inside front cover (LNAI).

Vol. 347: see inside front cover (LNAI).

CONTENTS

FUNDAMENTAL MECHANISMS FOR ARTIFICIAL INTELLIGENCE PROGRAMMING LANGUAGES

- AN INTRODUCTION -

Philippe Jorrand

LIFIA
INPG-CNRS
46, avenue Félix Viallet
38000 GRENOBLE
FRANCE

I. INTRODUCTION

Artificial intelligence is not computer science. But computers are essential tools for research in artficial intelligence. Computers are the kind of machinery which is best fitted for mechanizing the various models for perception, reasoning, learning and control of action which are relevant to artificial intelligence.

Most artificial intelligence programs are quite complex objects and mastering the complexity of their design is a major research objective which lies at the intersection of computer science and artificial intelligence. Progress in that domain relies both on experience and on theory.

The material presented in this chapter is an introduction to the fundamental mechanisms of artificial intelligence languages and architectures. These mechanisms and the objects they involve are most of the time presented quite formally. The use of appropriate formalisms is indeed the price to pay for mastering complexity : without it, the relevant and useful properties cannot be clearly isolated nor properly used.

II. TERMS AND BASIC OPERATIONS ON TERMS

The notion of term for representing data objects, programs, computations and proofs constitutes the most primitive layer for the notions presented in this chapter. On top of it, all other mechanisms will be constructed. The material in this section is mostly drawn from [Huet 85].

II.1. Strings, trees and terms

II.1.1. Strings

Strings are well known objects. Let Σ be a countable set of objects called symbols : Σ is the alphabet with which strings will be constructed. A string u of length n on Σ is a function from $\{0,...,n-1\}$ into Σ : Σ^n is the set of all strings of length n on Σ. Let u_i denote the value of $u(i-1)$: $u_i \in \Sigma$.

Some classical notations for well understood notions :

- $\Sigma^* = \cup_{u \in N} \Sigma^n$ is the set of all strings
- Λ denotes the null string : $\Lambda \in \Sigma^0$

- $'a'$ denotes $u \in \Sigma^1$, when $u_1 = a$, for $a \in \Sigma$
- $u \hat{} v$ denotes the concatenation of u and v
- $'abc' = 'a' \hat{} 'b' \hat{} 'c'$, where $a, b, c \in \Sigma$
- $u.a = u \hat{} 'a'$, where $u \in \Sigma^*$, $a \in \Sigma$.
- $u \leq v$ (u "is a prefix of" v) iff $\exists w$ such that $v = u \hat{} w$

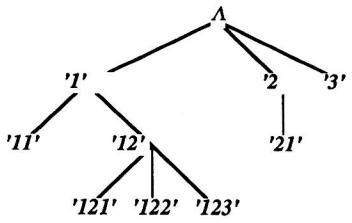
II.1.2. Trees

A tree has a structure and has pieces of information attached to its nodes and leaves. The structure of a tree will be represented by a set of strings on positive integers, called "positions", which enumerate in a "logical" way the nodes and leaves. Such a set of positions is called a tree domain. The information attached to nodes and leaves will be symbols from some alphabet Σ.

Thus, given N_+ , the set of positive integers, N_+^* is the set of all positions. Given a position $u = w.m$ and a position $v = w.n$, if $m<n$, u is said to be " left of" v : $u <_L v$. Then, a tree domain D is a subset of N_+^* which is closed under both the prefix and the "left of" orderings :

- $u \in D \wedge v < u \Rightarrow v \in D$
- $u \in D \wedge v <_L u \Rightarrow v \in D$

Clearly, Λ belongs to every non empty tree domain and is the position of the root of trees. For example, the set of positions $\{\Lambda, '1', '11', '12', '121', '122', '123', '2', '21', '3'\}$ is a tree domain which represents the following structure :



Finally, attaching information (i.e. symbols in Σ) to the nodes and leaves is done by defining a tree M as a function from its domain D into Σ. M is then called a Σ-tree : $M \in D \rightarrow \Sigma$. Given an arbitrary Σ-tree M, its domain is refered to by $D(M)$.

II.1.3. Operations on trees

Let M be a Σ-tree and u be a position in its domain $D(M)$. Two basic operations are defined on such trees : access to the subtree rooted at position u in M and grafting of a new subtree at position u in M.

The subtree rooted at position u in M is denoted by M/u. It is of course also a Σ-tree :

$$(M/u)(v) = M(u^{\wedge}v)$$

Given a Σ-tree N, the tree obtained by grafting N at position u in M is denoted by $M[u \leftarrow N]$. This tree is the same as M, except that the subtree which was rooted at position u in M is replaced by N in $M[u \leftarrow N]$. This tree is also a Σ-tree :

$$v \in D(M) \wedge \neg(u \leq v) \Rightarrow M[u \leftarrow N] = M(v)$$
$$w \in D(N) \wedge v = u^{\wedge}w \Rightarrow M[u \leftarrow N](v) = N(w)$$

II.1.4. Terms

Let M be a Σ-tree. The top-width of M is $//M// = max\{n \mid 'n' \in D(M)\}$, i.e. the number of subtrees directly descending under the root of M. On alphabet Σ, a function α into the integers N, called arity function, is defined : $\alpha \in \Sigma \rightarrow N$. An alphabet with an arity function is called a graded alphabet. A Σ-term is then a Σ-tree on a graded alphabet Σ, such that the number of subtrees directly descending from every node is equal to the arity of the symbol of Σ attached to that node :

$$M \text{ is a } \Sigma\text{-term} \Leftrightarrow \forall u \in D(M)\ //M/u// = \alpha(M(u))$$

The set of all Σ-terms is denoted by $T(\Sigma)$. The usual parenthesized syntax for terms is defined as follows: given $F \in \Sigma$, with $\alpha(F) = n$, and $M_1, ..., M_n \in T(\Sigma)$, then $F(M_1, ..., M_n) \in T(\Sigma)$.

II.2. Variables and substitutions

II.2.1. Terms with variables

Let V be a countable set of objects called variables, disjoint from Σ: $V \cap \Sigma = \varnothing$, and such that the arity of all variables is 0 : $\forall x \in V,\ \alpha(x) = 0$. Variables of V can be attached to positions of terms, as is already the case for symbols of Σ. Since their arity is 0, they will always appear on leaves of terms. The set of terms with variables is denoted by $T(\Sigma,V)$ and is equal to $T(\Sigma \cup V)$.

Let M be a term on Σ and V : $M \in T(\Sigma,V)$. The set of distinct variables which occur in M is :

$$V(M) = \{x \in V \mid \exists u \in D(M),\ M(u)=x\}$$

The number of distinct variables in M is denoted by $\nu(M) = |V(M)|$.

II.2.2. Substitutions

As is usual in the description of computations, the purpose of variables is to be replaced by values. Here, the considered values are terms : terms can be substituted for variables in $M \in T(\Sigma,V)$. Thus, some of the leaves of M to which variables are attached will be replaced by terms. Technically, this is achieved by grafting these terms at the corresponding leaf positions in M.

The elementary form of a substitution σ is a function from the set V of variables into the set $T(\Sigma,V)$ of terms, $\sigma \in V \rightarrow T(\Sigma,V)$, and is the identity almost everywhere. The domain of a substitution σ is the set of variables where σ is not the identity :

$$D(\sigma) = \{x \in V \mid \sigma(x) \neq x\}$$

In practice, substitutions are not simply applied to variables, but are applied to terms : for performing a substitution σ on the leaves of terms of the form $F(M_1, ..., M_n)$, σ is extended to a morphism over the set of terms $T(\Sigma,V)$:

$$\sigma(F(M_1, ..., M_n)) = F(\sigma(M_1), ..., \sigma(M_n))$$

If there are several occurrences of variable x, they will all be replaced by the same term $\sigma(x)$.

II.3. The domain of terms

II.3.1. Ordering among terms in $T(\Sigma,V)$

Let M and N be two terms in $T(\Sigma,V)$. M is said to be "less instanciated" than N iff N can be obtained from M by substituting terms for some of the variables in M :

$$M \leq N \Leftrightarrow \exists \sigma, N = \sigma(M)$$

This relation is also read "N is an instance of M".

For example, with $M = \boldsymbol{F(G(x,a),y)}$ and $N = \boldsymbol{F(G(H(b),a),y)}$, $M \leq N$ since $N = \sigma(M)$ with $\sigma(x)=\boldsymbol{H(b)}$. If $N = \boldsymbol{F(G(u,a),v)}$, it would still be the case that $M \leq N$. But, in this situation, N has exactly the same structure as M and differs from M only by a consistent renaming of its variables. As a consequence, with the inverse substitution, it would also be the case that $N \leq M$. The relation defined above is thus a quasi ordering, since its is not anti reflexive : both $M \leq N$ and $N \leq M$ can hold, while M and N are different terms. Such bijective substitutions (i.e. consistent renamings of variables) are called permutations. Two terms are said to be isomorphic if one of them can be obtained from the other by a permutation :

$$\forall M, N \in T(\Sigma,V), M \equiv N \Leftrightarrow M = \sigma(N) \text{ for some permutation } \sigma$$
$$\Leftrightarrow M \leq N \wedge N \leq M$$

II.3.2. Properties of the ordering in $T(\Sigma,V)$

The strict ordering ">", defined as

$$M > N \Leftrightarrow N \leq M \wedge \neg(M \leq N)$$

is a well founded ordering. This is easy to see, using the size $|M|$ of finite terms :

$$|M| = |F(M_1, ..., M_n)| = 1 + \sum_{i=1..n} |M_i|$$

Then, with $\mu(M) = |M| - v(M)$:

$$M > N \Rightarrow \mu(M) > \mu(N)$$

Thus, there is no infinite descending chain $M_1 > M_2 > \ldots$.

II.3.3. Terms form a complete lattice

Given two terms M and N in $T(\Sigma,V)$, it is always possible to find a term $M \cap N$ which is the most instanciated term such that both $M \cap N \leq M$ and $M \cap N \leq N$ hold. The term $M \cap N$ is then a greatest lower bound (g.l.b.) of M and N. This term is unique, up to the isomorphism $\equiv$.

For example, given $M = F(G(x,a),H(b))$ and $N = F(G(y,c),d)$, the term $M \cap N$ would be isomorphic to $F(G(u,v),w)$. A general definition of this operation "$\cap$" in $T(\Sigma,V)$ is as follows :

$$F(M_1, ..., M_n) \cap F(N_1, ..., N_n) = F(M_1 \cap N_1, ..., M_n \cap N_n)$$

$$M \cap N = \phi(M,N) \text{ in all other cases}$$

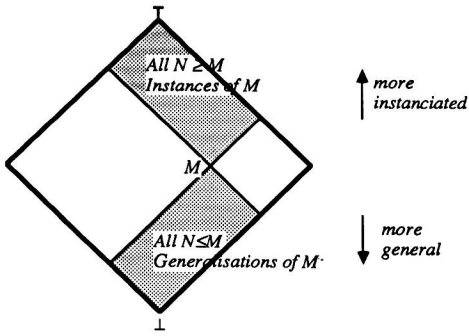
where ϕ is an arbitrary bijection which maps pairs of terms into variables :

$$\phi : T(\Sigma,V) \times T(\Sigma,V) \rightarrow V$$

For distinct ϕ, this definition of $M \cap N$ produces isomorphic results. The term $M \cap N$ is a g.l.b. of M and N under the ordering $\leq$. For identifying all isomorphic terms to a single object, the quotient of $T(\Sigma,V)$ by $\equiv$ is considered :

$$\mathbb{T}(\Sigma,V) = T(\Sigma,V) / \equiv$$

and this domain is completed by a top $\mathbf{T}$ (i.e. an element greater than all other elements). The domain $\mathbb{T}(\Sigma,V)$ is a complete lattice. It is easy to verify that the bottom of $\mathbb{T}(\Sigma,V)$ is $\perp = V$, the set of variables. Pictorially, $\mathbb{T}(\Sigma,V)$ can be represented as follows :



A term $\sigma(M)$ such that $V(\sigma(M)) = \varnothing$ is called a ground instance of M.